A Reaction Attack against Cryptosystems based on LRPC Codes

Simona Samardjiska[1], Paolo Santini[2], Edoardo Persichetti[3] and Gustavo Banegas[4,5]

[1] Radboud University
simonas@cs.ru.nl
[2] Universitá Politecnica delle Marche
p.santini@pm.univpm.it
[3] Florida Atlantic University
epersichetti@fau.edu
[4] Technische Universiteit Eindhoven, [5] Chalmers University of Technology
gustavo@cryptme.in

Abstract. Rank metric is a very promising research direction for code-based cryptography. In fact, thanks to the high complexity of generic decoding attacks against codes in this metric, it is possible to easily select parameters that yield very small data sizes. In this paper we analyze cryptosystems based on Low-Rank Parity-Check (LRPC) codes, one of the classes of codes that are efficiently decodable in the rank metric. We show how to exploit the decoding failure rate, which is an inherent feature of these codes, to devise a reaction attack aimed at recovering the private key. As a case study, we cryptanalyze the recent McNie submission to NIST's Post-Quantum Standardization process. Additionally, we provide details of a simple implementation to validate our approach.

1 Introduction

It is well known that, once quantum computers of an appropriate size will be available, traditional cryptographic schemes will not be secure anymore [43]. Code-based cryptosystems are among the most promising candidates for Post-Quantum Cryptography, the area concerned with designing cryptographic primitives which will be secure in this scenario. This is evident from the recent Call for Standardization issued by NIST [37], where the number of code-based submissions is second only to that of lattice-based ones [38]. In particular, code-based schemes seem to shine as solutions for encryption and key-exchange.

McEliece in 1978 [27] was the first to propose a code-based encryption scheme, based on the hardness of decoding random linear codes. This NP-hard problem

This work has been supported by the European Commission through the ERC Starting Grant 805031 (EPOQUE); through the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 643161; and by Sweden through the WASP expedition project Massive, Secure, and Low-Latency Connectivity for IoT Applications.

was exploited by selecting a binary Goppa code, for which a random-looking generator is released as public key, and encrypting a plaintext as a noisy codeword. While the private description allows for decoding (and hence decryption), the public one conveys no information about the code, and so the best attacks are generic decoding attacks such as Information-Set Decoding (ISD) [40], which are of exponential nature. This "code indistinguishability" assumption has been true for binary Goppa codes ever since, and the McEliece framework has now over 40 years of security history. However, this comes with the price of a fairly large public-key size, which can be as large as 1Mb with modern parameters [6].

The quest for obtaining compact key sizes started with investigating other families of codes (e.g. [35,44]), many of which have been shown to be insecure [45,30]. On top of that, a popular approach is to choose *structured* codes, such as Quasi-Cyclic [13,5] or Quasi-Dyadic [31,39], which allow for a dramatic reduction in the size of the public key. However, introducing algebraic structure is not always safe, as shown in [11]. As a result, algebraic codes with an algebraic structure do not seem to be an optimal choice for cryptographic schemes.

Currently, there are two major trends for obtaining code-based schemes with small keys. The first makes use of codes defined by very sparse parity-check matrices, such as LDPC and MDPC codes [4,32], while the second is based on *rank metric* codes [12]. In this paper, we focus on the latter, and in particular, we consider the case of LRPC codes, which are in a sense a point of contact between the two. In fact, LRPC stands for Low-Rank Parity-Check, and this class of codes is characterized by a "sparse" (in the rank metric sense) parity-check matrix, and therefore it can effectively be seen as a rank-metric equivalent of LDPC/MDPC codes. As we will see, and as it is often the case for rank-metric schemes, LRPC codes share many of the aspects of their Hamming metric counterpart, including vulnerabilities.

Our Contribution. In this paper, we show how to devise a reaction attack based on observing a collection of decryption failures, which are an inherent feature for all schemes based on probabilistic decoding algorithms. These attacks become feasible when the Decoding Failure Rate (DFR) of the scheme is non-negligible, as illustrated, for the Hamming case, in a famous paper by Guo, Johansson and Stankovski [21].

The scenario behind a reaction attack is that the attacker sends a large number of encrypted messages with small modifications on the messages and then observes the reaction of the decryption of those messages (the attacker does not take into account the results of the computation). For code-based cryptography, it is common to induce a decoding failure by selecting messages with a certain property, which is the case presented in [22]. It is important to mention that in this scenario the goal of the attacker is not to recover the message, but rather the private key (or an equivalent key for decryption). In our attack, we use an approach similar to [22], in that we are interested in collecting errors that produce decoding failures. However, in our case we only need to collect a small amount of error patterns to complete the attack. We will give a justification for this nice feature, and explain how it works in detail, in the main body of the paper.

While preparing the camera ready version of this paper, we became aware of an independent work proposing a reaction attack against LRPC cryptosystems [34]. In [34], the attack requires a significantly larger amount of decryption queries and is less general, since it assumes that the adversary is free to choose the error vectors which are used for encryption.

The paper is organized as follows. We begin with preliminary notions, and notation, in Section 2, including an overview of LRPC cryptosystems. Our attack is described in Section 3, with a detailed analysis of the success probability. In Section 4 we discuss equivalent keys and in Section 5 we give a description of an attack in this case, while also investigating the possibility of applying quantum techniques to speed up the attack. As a case study, in Section 6 we provide the results obtained when applying our attack to McNie, one of the first round NIST submissions.

2 Preliminaries

We use capital bold letters to denote matrices, and small bold letters to denote vectors. Given a matrix $\mathbf{A}$, its entry in the i-th row and j-th column is denoted as $a_{i,j}$; in analogous way, the i-th entry of a vector $\mathbf{a}$ is denoted as a_i. The rank of a matrix $\mathbf{A}$ is denoted by $|\mathbf{A}|$.

Let q be a prime power and m be an integer; we denote with $\mathbb{F}_q$ and $\mathbb{F}_{q^m}$ the finite fields of cardinality respectively equal to q and q^m; the set of all $n \times n$ matrices over $\mathbb{F}_q$ will be denoted by $\mathcal{M}_n(\mathbb{F}_q)$, and the set of all $n \times n$ invertible matrices by $GL_n(\mathbb{F}_q)$. We denote an index set $\{1, 2, \ldots, \tau\}$ by $[1; \tau]$.

When treating codes in rank metric, a useful description can be obtained by considering each vector as a matrix.

Let $B = \{B_1, \cdots, B_m\}$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, and define the function $\mathcal{F}_i : \mathbb{F}_{q^m} \rightarrow \mathbb{F}_q$ such that, for each $a \in \mathbb{F}_{q^m}$, $\mathcal{F}_i(a)$ corresponds to the i-th coefficient of a into the basis B. In other words, the following relation holds

$$a = \sum_{i=1}^{m} \mathcal{F}_i(a) B_i.$$

Let $V_n \subseteq \mathbb{F}_{q^m}^n$ be an n-dimensional subspace of $\mathbb{F}_{q^m}$; then, each vector $\mathbf{v} = \{v_i\} \in V_n$ can be represented as a matrix $\bar{\mathbf{V}} = \{v_{i,j}\} \in \mathbb{F}_q^{m \times n}$, whose entry in position (i, j) corresponds to $\mathcal{F}_i(v_j)$. In other words, the following two representations are equivalent

$$\mathbf{v} = [v_1, \cdots, v_n] \leftrightarrow \bar{\mathbf{V}} = \begin{bmatrix} \mathcal{F}_1(v_1) & \mathcal{F}_1(v_2) & \cdots & \mathcal{F}_1(v_n) \\ \mathcal{F}_2(v_1) & \mathcal{F}_2(v_2) & \cdots & \mathcal{F}_2(v_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{F}_m(v_1) & \mathcal{F}_m(v_2) & \cdots & \mathcal{F}_m(v_n) \end{bmatrix}.$$

Using this representation, rank metric codes can be defined in a very natural and intuitive way. In particular, given two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{q^m}^n$, the *rank distance* between $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$\mathrm{rd}(\mathbf{a}, \mathbf{b}) = \left|\bar{\mathbf{A}} - \bar{\mathbf{B}}\right| .$$

The *rank weight* of a vector $\mathbf{a}$ is then defined as $\mathrm{wt}(\mathbf{a}) = \mathrm{rd}(\mathbf{a}, \mathbf{0}_n) = |\bar{\mathbf{A}}|$, where $\mathbf{0}_n$ denotes the length-n null vector. The *support* of a vector $\mathbf{v} \in V_n$ is denoted as $\langle \mathbf{v} \rangle$ and corresponds to the subspace generated by its entries $v_1, \cdots, v_n$. With some abuse of notation, we use $\langle V_n \rangle$ to denote the subspace generated by the vectors in V_n.

2.1 Circulant Matrices and Quasi-Cyclic Codes

A *circulant matrix* is a matrix in which every row is obtained as a right cyclic shift of the previous. Eq. (1) shows a circulant matrix of size[5] p.

$$\mathbf{C}_p = \begin{bmatrix} t_0 & t_1 & \cdots & t_{p-1} \\ t_{p-1} & t_0 & \cdots & t_{p-2} \\ \vdots & & \ddots & \vdots \\ t_1 & t_2 & \cdots & t_0 \end{bmatrix} \tag{1}$$

Circulant $p \times p$ matrices over $\mathbb{F}_{q^m}$ form a ring that we will denote by $\mathcal{C}_p(\mathbb{F}_{q^m})$. Its cardinality is $|\mathcal{C}_p(\mathbb{F}_{q^m})| = q^{mp}$.

Proposition 1. *Let $x^p - 1 = p_1^{\alpha_1}(x) \cdot \cdots \cdot p_\tau^{\alpha_t}(x)$ be the factorization of $x^p - 1$ over $\mathbb{F}_{q^m}$ into powers of irreducible factors. The number of invertible circulant matrices in $\mathcal{C}_p(\mathbb{F}_{q^m})$ is equal to $\prod_{i=1}^{\tau} (q^{m \cdot d_i \alpha_i} - q^{m \cdot d_i(\alpha_i - 1)})$, where d_i is the degree of $p_i(x)$ in the factorization of $x^p - 1$.*

Proof. It is well known that $\mathcal{C}_p(\mathbb{F}_{q^m})$ is isomorphic to $\mathbb{F}_{q^m}[x]/\langle x^p - 1\rangle$. From the factorization $x^p - 1 = p_1^{\alpha_1}(x) \cdot \cdots \cdot p_\tau^{\alpha_\tau}(x)$ and the Chinese Remainder Theorem, $\mathbb{F}_{q^m}[x]/\langle x^p - 1\rangle$ is isomorphic to the direct product:

$$\mathbb{F}_{q^m}[x]/\langle x^p - 1\rangle \cong \mathbb{F}_{q^m}[x]/\langle p_1^{\alpha_1}(x)\rangle \times \cdots \times \mathbb{F}_{q^m}[x]/\langle p_\tau^{\alpha_t}(x)\rangle$$

The number of invertible elements in $\mathbb{F}_{q^m}[x]/\langle p_i^{\alpha_i}(x)\rangle$ is $q^{m \cdot d_i \alpha_i} - q^{m \cdot d_i(\alpha_i - 1)}$ where d_i is the degree of $p_i(x)$. Now it is easy to count the number of invertible elements in $\mathbb{F}_{q^m}[x]/\langle x^p - 1\rangle$. It is precisely the product of the invertible elements in each $\mathbb{F}_{q^m}[x]/\langle p_i^{\alpha_i}(x)\rangle$, i.e., $\mathcal{C}_p(\mathbb{F}_{q^m}) = \prod_{i=1}^{\tau} (q^{m \cdot d_i \alpha_i} - q^{m \cdot d_i(\alpha_i - 1)})$.

Note that when $\alpha_1 = \alpha_2 = \cdots = \alpha_\tau = 1$, $\mathbb{F}_{q^m}[x]/\langle x^p - 1\rangle$ factors into a direct product of fields, and our formula turns into $\prod_{i=1}^{\tau} (q^{m \cdot d_i} - 1)$. $\square$

[5] A circulant matrix can be defined as a special case of Toeplitz matrix; for more details about Toeplitz matrices see [19].

A *quasi-cyclic code* is a code with generator matrix of the form

$$\mathbf{G} = \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} & \cdots & \mathbf{C}_{1n_0} \\ \mathbf{C}_{21} & \mathbf{C}_{22} & \cdots & \mathbf{C}_{2n_0} \\ \vdots & & \ddots & \vdots \\ \mathbf{C}_{k_0 1} & \mathbf{C}_{k_0 2} & \cdots & \mathbf{C}_{k_0 n_0} \end{bmatrix} \tag{2}$$

where each matrix $\mathbf{C}_{ij}$ is a circulant matrix of the form (1).

2.2 LRPC Codes

A *Low-Rank Parity-Check (LRPC) code* $\mathcal{C}$ over $\mathbb{F}_{q^m}$ of length n, dimension k and rank d is described by an $(n-k)\times n$ parity-check matrix $\mathbf{H} = \{h_{i,j}\} \in \mathbb{F}_{q^m}^{(n-k)\times n}$, whose coefficients $h_{i,j}$ generate a subspace of $\mathbb{F}_{q^m}$ of dimension at most d. More precisely, each coefficient $h_{i,j}$ can be written as

$$h_{i,j} = \sum_{l=1}^{d} h_{i,j,l} F_l, \quad h_{i,j,l} \in \mathbb{F}_q, \tag{3}$$

where each $F_i \in \mathbb{F}_{q^m}$, and $F = \langle F_1, F_2, \cdots, F_d\rangle$ is a $\mathbb{F}_q$ subspace of $\mathbb{F}_{q^m}$ of dimension at most d generated by the basis $\{F_1, F_2, \cdots, F_d\}$.

Decoding of LRPC codes. Consider an LRPC code with parity-check matrix $\mathbf{H}$ of length n, dimension k and rank d, with basis $F = \{F_1, \cdots, F_d\}$. Let $\mathbf{e} = \{e_i\} \in \mathbb{F}_{q^m}^n$ be a vector of rank r, with basis $E = \{E_1, \cdots, E_r\}$. Recall that, considering the matrix representation, the vector $\mathbf{e}$ can be described as a matrix $\bar{\mathbf{E}} = \{e_{i,j}\}$, with $i \in [1; n]$, $j \in [1; r]$, such that

$$e_i = \sum_{j=1}^{r} e_{i,j} E_j, \quad e_{i,j} \in \mathbb{F}_q. \tag{4}$$

Let $\mathbf{s} \in \mathbb{F}_{q^m}^{n-k}$ be the *syndrome* of $\mathbf{e}$ with respect to $\mathbf{H}$, i.e. $\mathbf{He}^\top = \mathbf{s}$. Decoding consists in recovering $\mathbf{e}$, from the knowledge of $\mathbf{s}$. A decoding procedure, specific for the case of LRPC codes, has been proposed in [14], and is shown in Algorithm 1. In this section we briefly recall its main principles, in order to provide a basic understanding of the attack procedure we propose in this paper.

Under proper conditions (which we investigate in the following), the syndrome equation can be rewritten as a linear system whose unknowns are nr scalars in $\mathbb{F}_q$. Indeed, for the i-th coordinate of $\mathbf{s}$, we have

$$\begin{aligned} s_i = \sum_{j=1}^{n} h_{i,j} e_j &= \sum_{j=1}^{n} \left(\sum_{l=1}^{d} h_{i,j,l} F_l\right)\left(\sum_{u=1}^{r} e_{j,u} E_u\right) \\ &= \sum_{l=1}^{d}\sum_{u=1}^{r} F_l E_u \left(\sum_{j=1}^{n} h_{i,j,l} e_{j,u}\right). \end{aligned} \tag{5}$$

Then, by considering Eq. (5) for all $i \in [1; n-k]$, the syndrome equation can be rewritten as

$$\mathbf{s}' = \mathbf{A_H}\mathbf{e}'^{\top}, \tag{6}$$

where $\mathbf{s}' \in \mathbb{F}_q^{(n-k)rd}$, $\mathbf{A_H} \in \mathbb{F}_q^{(n-k)rd \times nr}$ and $\mathbf{e}' \in \mathbb{F}_q^{nr}$.

Essentially, the above equation corresponds to the writing of the syndrome equation in the base field $\mathbb{F}_q$. In particular, $\mathbf{s}'$ contains the coefficients of the syndrome in the basis $\{F_iE_j\}_{\substack{i \leq i \leq d \\ 1 \leq j \leq r}}$, while $\mathbf{A_H}$ and $\mathbf{e}'$ are obtained through a rewriting of $\mathbf{H}$ and $\mathbf{e}$. Then, decoding can be performed through Algorithm 1; essentially, what the decoder does is firstly recovering the support of the error vector (lines 1-5 in the algorithm), computing a basis for the found subspace (line 6) and then reconstructing the coefficients of the error vector in the selected basis (line 7).

Algorithm 1 Decoding of LRPC codes

```
    Input: $\mathbf{s} \in \mathbb{F}_{q^m}^{n-k}$, $\mathbf{s}' \in \mathbb{F}_q^{(n-k)rd}$, $\mathbf{A_H} \in \mathbb{F}_q^{(n-k)rd \times nr}$
    Output: $\mathbf{e}' \in \mathbb{F}_q$.
1: $S \leftarrow \langle s_1, s_2, \cdots, s_{n-k} \rangle$                    ▷ Syndrome space
2: for $i \leftarrow 1$ to $d$ do
3:     $S_i \leftarrow F_i^{-1} S$
4: end for
5: $E \leftarrow \bigcap_{j=1}^{d} S_j$                    ▷ Compute the error support
6: $\{E_1, \cdots, E_r\} \leftarrow$ basis for $E$
7: Solve $\mathbf{s}' = \mathbf{A_H}\mathbf{e}'^{\top}$          ▷ Find the coefficients of $\mathbf{e}$ in the basis $E$
8: return $\mathbf{e}'$
```

Note that Algorithm 1 is characterized by a certain failure probability, which can be estimated according to the system parameters. In particular, decoding failures can happen only because of the following three events [14].

1. Case of $\mathrm{Dim}(\langle EF \rangle) < rd$: this happens with probability $P_1 = \frac{d}{q^{m-rd}}$. (see [14, Sec. 3, Prop. 1]).
2. Case of $E \neq \bigcap_{i=1}^{d} S_i$: when $m > rd + 8$, this happens with probability $P_2 \ll 2^{-30}$. (see [14, Sec. 3, Remark 3]).
3. Case of $\mathrm{Dim}(S) < rd$ this happens with probability $P_3 = \frac{1}{q^{n-k+1-rd}}$. (see [14, Sec. 5, Prop. 4]).

For parameters of practical interest, we usually have $P_1, P_2 \ll P_3$: as we describe in the following sections, this fact is crucial for the success of our attack.

2.3 LRPC Cryptosystems

The key generation, encryption and decryption of the typical LRPC cryptosystem are summarized in Figure 1.

1. **Key generation:** Choose a random LRPC code over $\mathbb{F}_{q^m}$ of low rank d with support F and parity check $(n-k) \times n$ matrix $\mathbf{H}$, generator matrix $\mathbf{G}$ and decoding matrix $\mathbf{D}_H$ which can correct errors of rank r and a random invertible $(n-k) \times (n-k)$ matrix $\mathbf{R}$.
   Secret Key: the low rank matrix $\mathbf{H}$, the masking matrix $\mathbf{R}$.
   Public Key: the matrix $\mathbf{G}' = \mathbf{R}\mathbf{G}$.
2. **Encryption:** Translate the message m into a word $\mathbf{x}$, generate $\mathbf{e} \in \mathbb{F}_{q^m}$ randomly with rank r. Compute $\mathbf{c} = \mathbf{x}\mathbf{G}' + \mathbf{e}$.
3. **Decryption:** Compute syndrome $\mathbf{s} = \mathbf{H}\mathbf{c}^T$, recover the error vector $\mathbf{e}$ by decoding the LRPC code, then compute $\mathbf{x}\mathbf{G}' = \mathbf{c} - \mathbf{e}$ and $\mathbf{x}$.

Fig. 1. LRPC cryptosystem.

The LRPC cryptosystem described above was introduced in [14]. The authors first present the low-rank parity-check codes and their application in cryptography, and then describe a McEliece-like scheme; note that, in principle, a Niederreiter setting can be used as well.

While Figure 1 and the LRPC cryptosystem provide a general framework for schemes based on LRPC codes, the usual setting in practical schemes is to use specific types of LRPC codes that allow shorter keys. These include Quasi-cyclic codes, as shown in [15] (and later also used in McNie [17,24] and Ouroboros-R [29]), or ideal codes (which are a generalization of LRPC codes and used in LAKE [1], Locker [2] and Rollo [28]). All of the previous cryptosystems show clear advantage over cryptosystems in the Hamming metric - for the same level of security, the keys are orders of magnitude smaller. For instance, the public key in Classic McEliece is 132KB while Rollo-II has a public key of size 2.4KB. Furthermore, ideal codes (with additional assumptions) have been used in the construction of the signature scheme Durandal [3] - showing once more the advantage over the Hamming metric where the construction of efficient signature schemes is still a problem.

In what follows, we describe McNie [17,24] - a first round candidate [38] to the NIST PQ crypto standardization process [37]. We will use McNie to showcase our reaction attack in Section 6.

McNie follows a "hybrid" framework using both McEliece and Niederreiter in the encryption process. The scheme employs QC codes with low weight parity-check matrices of the form $[\mathbf{H}_1\ \mathbf{H}_2\ \mathbf{H}_3]$ and $\left[\begin{smallmatrix} \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{H}_4 \\ \mathbf{H}_5 & \mathbf{H}_6 & \mathbf{H}_7 & \mathbf{H}_8 \end{smallmatrix}\right]$ where $\mathbf{H}_i$ are circulant matrices. The authors refer to these codes as 3- and 4-Quasi-Cyclic codes. The key generation, encryption and decryption of McNie are summarized in Figure 2.

Remark 1. According to the protocol specifications [17], in the general description of the scheme, the authors suggest the possibility to further use a permutation matrix $\mathbf{P}$ used to form $\mathbf{F}$ as $\mathbf{F} = \mathbf{G}'\mathbf{P}^{-1}\mathbf{H}^\top\mathbf{S}$. However, in the actual proposal, this matrix is set to the identity matrix, so it is never used. Therefore, we do not see a reason to use it and burden the description.

1 **Key generation:** Choose a random 3 or 4 generator QC LRPC code over $\mathbb{F}_{q^m}$ of low rank d, parity check $(n-k)\times n$ matrix $\mathbf{H}$ and generator matrix $\mathbf{G}$. Further, choose a random invertible $(n-k)\times(n-k)$ matrix $\mathbf{S}$ and a random $l\times(n-k)$ matrix $\mathbf{G}'$.
Secret Key: The low rank matrix $\mathbf{H}$, and the masking matrix $\mathbf{S}$.
Public Key: The matrices $\mathbf{F}=\mathbf{G}'\mathbf{H}^\top\mathbf{S}$ and $\mathbf{G}'$.
2 **Encryption:** To encrypt a message $\mathbf{m}\in\mathbb{F}_{q^m}$, generate a random $\mathbf{e}\in\mathbb{F}_{q^m}$ of rank r. Compute $\mathbf{c}_1=\mathbf{m}\mathbf{G}'+\mathbf{e}$ and $\mathbf{c}_2=\mathbf{m}\mathbf{F}$. The ciphertext is $(\mathbf{c}_1,\mathbf{c}_2)$.
3 **Decryption:** Compute syndrome $\mathbf{s}'=\mathbf{c}_1\mathbf{H}^\top-\mathbf{c}_2\mathbf{S}^{-1}=\mathbf{e}\mathbf{H}^\top$. Recover the error vector $\mathbf{e}$ by decoding the LRPC code, then compute $\mathbf{m}\mathbf{G}'=\mathbf{c}_1-\mathbf{e}$ and obtain $\mathbf{m}$ by solving the obtained system.

Fig. 2. The McNie cryptosystem [17,24].

3 A Reaction Attack

We are now ready to describe the details of our attack. The main idea is to exploit decoding failures caused by the syndrome $\mathbf{s}$ not generating the whole space $\langle FE\rangle$. Thus, for ease of exposition, in this section we will assume that this is the case. Later we will show that the influence of other types of decoding failures to the success of our attack is negligible, thus justifying the current assumption.

Suppose that an adversary $\mathcal{A}$ interacts with a decryption oracle $\mathcal{D}$ of an LRPC cryptosystem. He continuously sends encrypted messages to $\mathcal{D}$ and waits for the reaction from the oracle. If $\mathcal{D}$ returns failure, $\mathcal{A}$ records the error $\mathbf{e}$ that he used in the encryption of the message. $\mathcal{A}$ collects a total of t error vectors, where t is chosen appropriately. We will discuss this choice later in this section.

Let $\mathbf{e}$ be an error vector that $\mathcal{A}$ collected during his interaction with $\mathcal{D}$. We now show to use this information to recover the secret matrix $\mathbf{H}$.

Recall from Section 2.2, Eq. (6), that we can express the syndrome equation over the base field as $\mathbf{s}'=\mathbf{A}_\mathbf{H}\mathbf{e}'^\top$, where $\mathbf{s}'$ contains the coefficients of the syndrome in the basis $\{F_iE_j\}_{\substack{1\le i\le d\\1\le j\le r}}$. Alternatively, directly from (5), the syndrome equation can be written in a matrix form: $\mathbf{s}$ can be written as the product between the basis $(F_1E_1,F_1E_2\dots,F_dE_r)$ and a matrix $\bar{\mathbf{A}}_{\mathbf{H},\mathbf{e}}\in\mathbb{F}_q^{rd\times(n-k)}$:

$$\mathbf{s}=(F_1E_1,F_1E_2\dots,F_dE_r)\cdot\bar{\mathbf{A}}_{\mathbf{H},\mathbf{e}} \tag{7}$$

The key observation in our attack is that a decoding failure occurs when the matrix $\bar{\mathbf{A}}_{\mathbf{H},\mathbf{e}}$ is not of full rank - in other words, the left kernel of the matrix $\bar{\mathbf{A}}_{\mathbf{H},\mathbf{e}}$ is non-trivial. This means that there must exist (at least) one nonzero vector $\mathbf{v}_\mathbf{e}\in\mathbb{F}_q^{rd}$, $\mathbf{v}_\mathbf{e}\neq 0_{1\times rd}$, such that

$$\mathbf{v}_\mathbf{e}\cdot\bar{\mathbf{A}}_{\mathbf{H},\mathbf{e}}=\mathbf{0}_{1\times n-k}. \tag{8}$$

Now consider our attack scenario. The adversary $\mathcal{A}$ knows the error $\mathbf{e}$ that caused the matrix $\bar{\mathbf{A}}_{\mathbf{H},\mathbf{e}}$ to be of non-full rank. He, however, does not know the coefficients $\mathbf{h} = \{h_{i,j,l}\}_{\substack{1\leq l\leq d\\ 1\leq i\leq n-k\\ 1\leq j\leq n}}$ of the matrix $\mathbf{H}$, and therefore he does not know the kernel vector $\mathbf{v_e}$. Setting $\bar{\mathbf{A}}_\mathbf{e}(\mathbf{h}) = \bar{\mathbf{A}}_{\mathbf{H},\mathbf{e}}$ to emphasize the unknown coefficients $\mathbf{h}$, we can rewrite (8) as

$$\mathbf{v_e} \cdot \bar{\mathbf{A}}_\mathbf{e}(\mathbf{h}) = \mathbf{0}_{1\times n-k}. \tag{9}$$

The main step of our attack now boils down to finding the solutions to Eq. (9) in the unknown coefficients $\mathbf{h}$ of $\mathbf{H}$ and the unknown kernel vector $\mathbf{v_e}$.

Observe that we can actually use several errors $\mathbf{e}_1, \dots, \mathbf{e}_t$ to form equations similar to Eq. (9). In these equations for each error $\mathbf{e}_i$ we introduce a new appropriate kernel element $\mathbf{v}_{\mathbf{e}_i}$. However, they all share the same unknown coefficients of the matrix $\mathbf{H}$. Thus, we can form the following system:

$$\begin{cases} \mathbf{v}_{\mathbf{e}_1} \cdot \bar{\mathbf{A}}_{\mathbf{e}_1}(\mathbf{h}) = \mathbf{0}_{1\times n-k} \\ \mathbf{v}_{\mathbf{e}_2} \cdot \bar{\mathbf{A}}_{\mathbf{e}_2}(\mathbf{h}) = \mathbf{0}_{1\times n-k} \\ \dots \\ \mathbf{v}_{\mathbf{e}_t} \cdot \bar{\mathbf{A}}_{\mathbf{e}_t}(\mathbf{h}) = \mathbf{0}_{1\times n-k} \end{cases} \tag{10}$$

The right value for t depends on several factors: the method of solving, the parameters of the system, but most notably the nature of the system. In principle, t should be big enough such that solving the system unambiguously gives the coefficients of $\mathbf{H}$. We will discuss the choice of t in the next section.

Suppose that t is chosen appropriately. Observe that system (10) is a system of bilinear equations reminiscent to the equations obtained in the MinRank problem [8]. We can thus try solve this system using similar strategies as in at least three different methods for solving MinRank - the Kernel method [18], Kipnis-Shamir method [25] and the minors method [9]. The main difference is that first, there are several polynomial matrices whose non trivial kernel needs to be found and second, all of these matrices are polynomial matrices in the same variables. At first sight, this situation bears similarities to Simultaneous MinRank [7,10] which is commonly encountered in $\mathcal{MQ}$ cryptography. However, since the different errors $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_t$ produce different matrices $\bar{\mathbf{A}}_{\mathbf{e}_1}(\mathbf{h}), \bar{\mathbf{A}}_{\mathbf{e}_2}(\mathbf{h}), \dots, \bar{\mathbf{A}}_{\mathbf{e}_t}(\mathbf{h})$, it is not clear how to use the common techniques that significantly speed up the attack in $\mathcal{MQ}$ cryptosystems.

In the next subsection, we will describe in detail a Kernel method - like approach to solving system (10). A straightforward application of the other algebraic methods results in a significantly larger complexity. Therefore a deeper insight into the properties of system (10) is necessary in order to apply these efficiently.

3.1 Solving System (10) by Kernel Guessing

Suppose that the adversary $\mathcal{A}$ has collected t error vectors $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_t$ that cause decryption failures. As before, we assume that the corresponding matrices $\bar{\mathbf{A}}_{\mathbf{e}_1}(\mathbf{h}), \bar{\mathbf{A}}_{\mathbf{e}_2}(\mathbf{h}), \dots, \bar{\mathbf{A}}_{\mathbf{e}_t}(\mathbf{h})$ are singular. This means that size of the kernels of these matrices is at least 1. The adversary now tries to guess the vectors $\mathbf{v}_{\mathbf{e}_i}$, such that $\mathbf{v}_{\mathbf{e}_i}$ belongs to the kernel of $\bar{\mathbf{A}}_{\mathbf{e}_i}(\mathbf{h})$ for each $i \in [1;t]$. If all vectors $\mathbf{v}_{\mathbf{e}_i}$ are correctly guessed, what remains, is to solve the obtained linear system in the unknown coefficients $\mathbf{h}$. In general, in order to obtain a unique solution we need to form at least the same number of equations as variables. For each $\mathbf{v}_{\mathbf{e}_i}$ we can form $n-k$ equations, so we need $t(n-k)$ to be at least as the number of variables. For a random LRPC code of rank d, the number of unknown coefficients of the matrix $\mathbf{H}$ is $n(n-k)d$. Hence we need:

$$t \geq nd.$$

If, in addition, the code is quasi-cyclic and its parity-check matrix is made of circulant matrices of size p, the number of unknown coefficients is $n(n-k)d/p$. In this case

$$t \geq \frac{nd}{p}. \tag{11}$$

Let us denote the probability of correctly guessing a kernel vector corresponding to an error $\mathbf{e}_i$ by $P_{\mathbf{e}_i}$. This probability clearly depends on the dimension of the kernel of $\bar{\mathbf{A}}_{\mathbf{e}_i}(\mathbf{h})$. Let us denote this dimension as $K_{\mathbf{e}_i} \geq 1$: then, we know that $|Ker(\bar{\mathbf{A}}_{\mathbf{e}_i}(\mathbf{h}))| = q^{K_{\mathbf{e}_i}}$. Then,

$$P_{\mathbf{e}_i} = \frac{q^{K_{\mathbf{e}_i}}}{q^{rd}} = q^{-(rd-K_{\mathbf{e}_i})}. \tag{12}$$

Clearly, a larger kernel of some of the matrices would make the attack faster. However, there is no way to detect whether a matrix $\bar{\mathbf{A}}_{\mathbf{e}_i}(\mathbf{h})$ associated to an error $\mathbf{e}_i$ would have a larger kernel. Therefore we must assume the worst case, i.e. a kernel of dimension 1. It remains open whether it is possible to devise a strategy to generate error vectors that induce matrices of larger kernels.

Let us denote the probability of all vectors $\mathbf{v}_{\mathbf{e}_i}$ being correctly guessed by P_t. Then

$$P_t = P_{\mathbf{e}_i}^t = q^{-(rd-1)t} \tag{13}$$

After the kernel vectors have been guessed, system (10) becomes an overdetermined linear system over $\mathbb{F}_q$. Solving it gives the coefficients of $\mathbf{H}$. However, the basis F is still unknown. Luckily, knowing the coefficients of $\mathbf{H}$ turns out to be enough to find the basis F: this part is now easy and F can be obtained from sufficiently many message-ciphertext pairs and the syndrome equation. A high level description of the attack is given by Algorithm 2.

In Algorithm 2, through the procedure CollectErrors, the adversary interacts with the decryption oracle $\mathcal{D}$, by sending him encrypted messages and waiting for decryption failures. Each time there is a failure, the adversary saves the error he used, until enough errors that cause decryption failures are collected. The additional operations that the adversary performs are one encryption and one decryption of the scheme.

Once the errors have been obtained the main part of the attack can begin. Note that the collection of the errors can be done only once, and we need only

Algorithm 2 Reaction attack on LRPC codes

```
    Input: $d, t, \ell \in \mathbb{Z}$
    Output: Matrix $\mathbf{H}$ of rank $d$
 1: $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_t \leftarrow \mathsf{CollectErrors}(\mathsf{pk}, \mathcal{D}(\mathsf{sk}))$   ▷ Collect errors from decryption failures
 2: repeat
 3:     $\mathbf{v}_{\mathbf{e}_1}, \mathbf{v}_{\mathbf{e}_2}, \ldots, \mathbf{v}_{\mathbf{e}_t} \leftarrow_R \mathbb{F}_q^{rd}$   ▷ Guess kernel vectors
 4:     $\mathbf{h} \leftarrow \mathsf{SolveH}(\mathbf{v}_{\mathbf{e}_1}, \mathbf{v}_{\mathbf{e}_2}, \ldots, \mathbf{v}_{\mathbf{e}_t}, \mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_t)$   ▷ Solve system (10)
 5:     if $\mathbf{h} \neq \perp$ then
 6:         $\{(\mathbf{m}_i, \mathbf{e}_i, \mathbf{c}_i)\}_{i=1}^{\ell} \leftarrow \mathsf{CollectMEC}(\mathsf{pk})$   ▷ Collect messages, errors, ciphertexts
 7:         $F, \mathsf{success} \leftarrow \mathsf{SolveF}(\mathbf{h}, \{(\mathbf{m}_i, \mathbf{e}_i, \mathbf{c}_i)\}_{i=1}^{\ell})$   ▷ Find basis $F$
 8:     else $\mathsf{success} \leftarrow \perp$
 9:     end if
10: until success
11: $\mathbf{H} \leftarrow \mathsf{Reconstruct}(\mathbf{h}, F)$   ▷ Reconstruct the matrix $\mathbf{H}$
12: return $\mathbf{H}$
```

a handful of errors unlike in the Hamming metric (see Section 3.2 for a more detailed discussion).

Next, the function SolveH denotes the procedure for solving system (10) for some guessed kernel elements corresponding to the obtained errors. If system (10) has a solution, then SolveH will return this solution, otherwise it will return $\perp$. This solution is then used in SolveF together with ℓ valid triplets $(\mathbf{m}_i, \mathbf{e}_i, \mathbf{c}_i)$ of messages, errors and ciphertexts generated in the procedure $\mathsf{CollectMEC}$. SolveF is a procedure whose main goal is to find the basis F. However depending on the scheme, there might be other parts of the secret key sk that can be found in this procedure. The value of ℓ is also dependant on the scheme. We present an instantiation of SolveF for McNie [17] in Section 6.

We are now ready to state the total complexity of our attack. It is

$$\begin{aligned} Cost(\mathsf{React}) = & P_3^{-1}(Cost(\mathsf{Enc} \wedge \mathsf{Dec})) + \\ & + P_t^{-1}(Cost(\mathsf{SolveH}) + \ell Cost(\mathsf{Enc}) + Cost(\mathsf{SolveF})) \end{aligned} \tag{14}$$

where $P_3 = \frac{1}{q^{n-k+1-rd}}$ is the failure rate of the scheme (see Section 2.2), $P_t = q^{-(rd-1)t}$ for a d rank LRPC code and errors of rank r. In the case of random LRPC codes $t = nd$ and $Cost(\mathsf{SolveH}) = n^3(n-k)^3d^3$. When the code is quasi-cyclic and uses circulant matrices of size p, $t = \frac{nd}{p}$ and $Cost(\mathsf{SolveH}) = \frac{n^3(n-k)^3d^3}{p^3}$. As said earlier, $Cost(\mathsf{SolveF})$ depends on the scheme, but usually, $Cost(\mathsf{SolveF}) < Cost(\mathsf{SolveH})$. See Section 6 for more details about this.

3.2 Analogies and Differences with the Hamming Metric

The cryptanalysis procedure we have described in this section resembles the one proposed by Guo et al. in [21], tailored at the McEliece cryptosystem using quasi-cyclic Moderate-Density Parity-Check (MDPC) codes [33], decoded in the Hamming metric. Essentially, these codes are a special case of Low-Density

Parity-Check (LDPC) codes [16], i.e., codes which are described by a parity-check matrix that contains a low number of set entries. These codes admit efficient decoding algorithms, like the Bit Flipping (BF) decoder or some of its variants, which are all based on the sparsity of the parity-check matrix and are characterized by some intrinsic decoding failure probability which, as originally observed in [21], somehow depends on geometrical relations between the error vector and the secret parity-check matrix.

Then, reaction attacks can be mounted, by means of statistical tests on the decoding outcomes of a large number of decryption queries. In particular, these tests are used to guess the number of overlapping ones between columns in the secret key [41]; clearly, in order to achieve statistical reliability, the number of observed decryption instances (i.e., the number of queries) needs to be sufficiently large. For instance, we can consider the empirical results for the parameters that were broken in [21]: the authors used, in all successful attacks, more than 10^8 decryption queries. Considering a decoding failure probability approximately equal to 10^{-4}, this leads to a number of observed events of decoding failures in the order of 10^4.

There are clear differences between reaction attacks in the Hamming metric and the one we propose in this paper. First of all, in the rank metric case, no statistical test is needed: events of decoding failures are due (with overwhelming probability) to some rank deficiency in the syndrome, and this fact is used to establish algebraic relations like that in Eq. (9). This difference is emphasized by the fact that the number of failure events that an adversary needs to collect is significantly lower than the one that is needed for the Hamming metric case.

Additionally, in the Hamming metric case, the feasibility of reaction attacks is somehow related to the chosen decoder and to its setting [36], in the sense that modifications in the decoding procedure and/or slight variations in its setting might lead to significant differences in the attack outcome. This difference arises a question on the existence of alternative LRPC decoding techniques, and on their eventual effect on reaction attacks procedures.

Another crucial difference is represented by the fact that, for LDPC codes, only few parity-check matrices can be used to efficiently perform decoding on a given corrupted codeword. Indeed, for a given parity-check matrix $\mathbf{H}$, each matrix $\mathbf{H}' = \mathbf{W}\mathbf{H}$, with $\mathbf{W}$ being non-singular, is again a valid parity-check matrix, but $\mathbf{W}$ preserves the density only when it is a permutation matrix. When $\mathbf{W}$ is not a permutation matrix, in fact, rows of $\mathbf{H}'$ correspond to linear combinations of rows of $\mathbf{H}$: thus, their density is, with overwhelming probability, larger than that of $\mathbf{H}$. This means that only the actual $\mathbf{H}$, or a row-permuted version of it, guarantees efficient decoding of intercepted ciphertexts. Then, when mounting a reaction attack, the adversary's goal is that of reconstructing exactly one of these matrices. In the rank metric case the number of parity-check matrices that allow for efficient decoding techniques is significantly larger - we show in the next section that any matrix of the form $\mathbf{W}\mathbf{H}$ can be used to efficiently decode. In such a case, we speak of *equivalent keys*: this fact, as we describe in the next section, allows for significant reductions in the attack complexity.

4 Equivalent Keys in LRPC Cryptosystems

In the previous section we described the basic attack that makes use of decryption failures. Now, we dig a little deeper, and show that due to existence of particular equivalent keys, it is possible to speed up the attack by an exponential factor.

We start with a well known property of weight preservation in the rank metric. For completeness we include a proof that will be useful later on.

Proposition 2. *Let* $\mathbf{b} \in \mathbb{F}_{q^m}^n$*, and let* $\mathbf{W} \in GL_n(\mathbb{F}_q)$*. Then:*

$$\mathrm{wt}(\mathbf{b}) = \mathrm{wt}(\mathbf{b} \cdot \mathbf{W}).$$

In other words, weight is preserved under multiplication by non-singular matrices over $\mathbb{F}_q$*.*[6]

Proof. Let $\mathrm{wt}(\mathbf{b}) = d$. This means that $\mathbf{b}$ can be represented as $\mathbf{b} = \mathbf{F} \cdot \bar{\mathbf{B}}$, where $\mathbf{F} = (F_1, \dots, F_d)$, and $< F_1, \dots, F_d >$ is a basis of some d-dimensional subspace of $\mathbb{F}_{q^m}^n$, and $\bar{\mathbf{B}} \in \mathcal{M}_{d,n}(\mathbb{F}_q)$ is the (full rank) matrix representation of $\mathbf{b}$. Now

$$\mathbf{b} \cdot \mathbf{W} = \mathbf{F} \cdot \bar{\mathbf{B}} \cdot \mathbf{W} = \mathbf{F} \cdot (\bar{\mathbf{B}} \cdot \mathbf{W}).$$

Since $\mathbf{W}$ is invertible, $\bar{\mathbf{B}} \cdot \mathbf{W}$ is of full rank, i.e., $\mathrm{wt}(\mathbf{b} \cdot \mathbf{W}) = d$. □

As a direct consequence, we have the following:

Proposition 3. *Let* $\mathbf{H} \in \mathcal{M}_{n-k,n}(\mathbb{F}_{q^m})$ *be the parity check matrix of an LRPC code* $\mathcal{C}$ *of rank* d*. Let* $\mathbf{W} \in GL_{n-k}(\mathbb{F}_q)$ *be arbitrary. Then* $\mathbf{WH}$ *is a parity check matrix for the code* $\mathcal{C}$ *of the same rank* d*.*

Proof. Follows directly from the previous proposition, by considering the columns of $\mathbf{H}$ as vectors of weight d. □

Definition 1. *Let* $P = (KeyGen, Enc, Dec)$ *be an LRPC cryptosystem with a secret key* $\mathsf{sk} = (\mathbf{H}, \cdot)$*. We say that* P *has an equivalent key* $\mathsf{sk}' = (\mathbf{H}', \cdot')$*, if* $\mathsf{sk}' \neq \mathsf{sk}$ *and* sk' *can be used as a secrete key for* P *with equal efficiency as* sk*. In particular,* $\mathbf{H}'$ *is of the same rank as* $\mathbf{H}$ *and can be used in the decoding procedure with the same efficiency as* $\mathbf{H}$*.*

With some abuse of this definition, we will also say that $\mathbf{H}'$ *is an equivalent key of* $\mathbf{H}$*.*

As a direct consequence of Proposition 3, we have:

Corollary 1. *Let* $P = (KeyGen, Enc, Dec)$ *be an LRPC cryptosystem with a secret key* $\mathsf{sk} = (\mathbf{H}, \cdot)$ *where* $\mathbf{H} \in \mathcal{M}_{n-k,n}(\mathbb{F}_{q^m})$*. Let* $\mathbf{W} \in GL_{n-k}(\mathbb{F}_q)$ *be arbitrary. Then,* $\mathsf{sk}' = (\mathbf{WH}, \cdot)$ *is an equivalent key for* P*.*

[6] Recall that in Hamming metric, weight is preserved under multiplication by permutation matrices

A particular equivalent key is of our interest; later we present a key recovery attack that recovers exactly such a key.

Let $\mathbf{H} \in \mathcal{M}_{n-k,n}(\mathbb{F}_{q^m})$ be the parity check matrix of an LRPC code $\mathcal{C}$ of rank d. We rewrite $\mathbf{H}$ as:

$$\mathbf{H} = \begin{bmatrix} \sum_{i=1}^{d} h_{1,1,i}F_i & \sum_{i=1}^{d} h_{1,2,i}F_i & \cdots & \sum_{i=1}^{d} h_{1,n,i}F_i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{d} h_{n-k,1,i}F_i & \sum_{i=1}^{d} h_{n-k,2,i}F_i & \cdots & \sum_{i=1}^{d} h_{n-k,n,i}F_i \end{bmatrix} =$$

$$= \sum_{i=1}^{d} \left(\begin{bmatrix} h_{1,1,i} & h_{1,2,i} & \cdots & h_{1,n,i} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k,1,i} & h_{n-k,2,i} & \cdots & h_{n-k,n,i} \end{bmatrix} \right) F_i$$

i.e as

$$\mathbf{H} = \sum_{i=1}^{d} \hat{\mathbf{H}}_i \cdot F_i = \sum_{i=1}^{d} [\hat{\mathbf{H}}_{i1} | \hat{\mathbf{H}}_{i2}] \cdot F_i \tag{15}$$

where $\hat{\mathbf{H}}_i$ is the matrix of coefficients corresponding to the basis element F_i.

Without loss of generality, assume: $\hat{\mathbf{H}}_1 = [\hat{\mathbf{H}}_{11} | \hat{\mathbf{H}}_{12}]$ where $\hat{\mathbf{H}}_{11} \in GL_{n-k}(\mathbb{F}_q)$. Then $\mathbf{H}' = \hat{\mathbf{H}}_{11}^{-1} \cdot \mathbf{H}$ is an equivalent key and can be written as:

$$\mathbf{H}' = [\mathbf{I}_{n-k} | \hat{\mathbf{H}}'_{12}] \cdot F_1 + \sum_{i=2}^{d} [\hat{\mathbf{H}}'_{i1} | \hat{\mathbf{H}}'_{i2}] \cdot F_i \tag{16}$$

where $\hat{\mathbf{H}}'_{t1} = \hat{\mathbf{H}}_{11}^{-1} \cdot \hat{\mathbf{H}}_{t1}$ and $\hat{\mathbf{H}}'_{t2} = \hat{\mathbf{H}}_{11}^{-1} \cdot \hat{\mathbf{H}}_{t2}$.

A crucial observation to make is that in the general case, the equivalent key $\mathbf{H}'$ is determined by $n(n-k)d - (n-k)^2$ coefficients, as opposed to $n(n-k)d$ coefficients for $\mathbf{H}$. This observation can be used to reduce the size of the private key by storing $\mathbf{H}'$ instead of $\mathbf{H}$. It can also be used to speed up our reaction attack if we recover $\mathbf{H}'$ instead of $\mathbf{H}$ because now we have less unknown coefficients, i.e., less variables in system (10). We need however to show that the collected error vectors that correspond to a secret $\mathbf{H}$ are also valid for the equivalent keys.

Proposition 4. *For an arbitrary LRPC code $\mathcal{C}$ of length n, dimension k and rank d over $\mathbb{F}_{q^m}$, with parity check matrix $\mathbf{H}$, if an error vector $\mathbf{e}$ causes a decoding failure, then the same error vector causes decoding failure for any equivalent $\mathbf{WH}$ where $\mathbf{W} \in GL_{n-k}(\mathbb{F}_q)$.*

Proof. Recall that the error vector $\mathbf{e}$ in system (10) cause the syndrome to be of non-maximal rank. By multiplying the syndrome equation by $\mathbf{W} \in GL_{n-k}(\mathbb{F}_q)$ we obtain $(\mathbf{WH}) \cdot \mathbf{e}_i^\top = \mathbf{W} \cdot \mathbf{s}^\top$, which from Proposition 2 means that the same error vectors cause syndrome of non-maximal rank for the matrix $\mathbf{WH}$. From Proposition 3 we know that $\mathbf{WH}$ is an equivalent key. □

Equivalent Keys for Quasi-cyclic codes. Note that if $\mathbf{H}$ is quasi-cyclic, then so are the matrices $\hat{\mathbf{H}}_i$ in Eq. (15). Then $\mathbf{H}' = \hat{\mathbf{H}}_{11}^{-1} \cdot \mathbf{H}$ is an equivalent key of the form (16). This key $\mathbf{H}'$ is determined by $\frac{n(n-k)d-(n-k)^2}{p}$ coefficients, as opposed to $\frac{n(n-k)d}{p}$ coefficients for $\mathbf{H}$.

5 Equivalent Key Attack on Quasi-Cyclic H

In the previous section we showed that under some plausible conditions, there exists an equivalent key that is determined by less coefficients than the original one. Therefore it makes sense to look for this key in our attack. In the case of QC codes, we need $\frac{(n-k)^2}{p}$ less variables, so the number of kernel elements that we need to guess instead of (11) becomes

$$t \geq \frac{nd-(n-k)}{p}. \tag{17}$$

The gain in the probability compared to (13) is a factor of $q^{(rd-1)\frac{n-k}{p}}$, so looking for an equivalent key speeds the attack by an exponential factor.

5.1 Probability of Success

The success of the attack described above depends on two conditions being satisfied. First, recall that in Section 3 we assumed that all collected errors are a result of the syndrome not being of full rank. The results from Section 2.2 show that this is not always the case. However, as we will show shortly, this happens with significant probability, so we can conclude that the feasibility of our attack is not affected by these other types of decryption failures.

There is however one more place where the attack may fail - if we want to recover the good equivalent key from Section 4, the success of the attack further depends on the probability that such an equivalent As we will see later, this probability is also big, so it is safe to assume that an appropriate equivalent key exists.

Syndrome of non-full rank. We say that an observed decoding failure event is *useful* if it is due to the case of $\mathrm{Dim}(S) < rd$. This happens with probability

$$\rho = \frac{P_3}{P_1+P_2+P_3} = \frac{1}{1+\frac{P_1+P_2}{P_3}}, \tag{18}$$

where the above probabilities depend on the system parameters and have been defined in Section 2.2. Since we typically have $P_1, P_2 \ll P_3$, we commonly have $\rho \approx 1$. Now the the attack will be successful only if all of the t collected errors are useful, i.e.,

$$Pr_s = \rho^t. \tag{19}$$

Existence of good equivalent key. The attack presented in Section 5 requires that a good equivalent key exists. Recall that in Eq. (20), we assumed that $\hat{\mathbf{H}}_{11}$ is invertible matrix of size $n-k$. Actually, note that our attack does not make a difference between the matrices $\hat{\mathbf{H}}_{l1}$, for $l \in [1;d]$, so it is enough that at least one of these is invertible. In other words, our attack will find an equivalent key of the form:

$$\mathbf{H}' = [\mathbf{I}_{n-k} | \hat{\mathbf{H}}'_{l2}] \cdot F_l + \sum_{\substack{i=2 \\ i \neq l}}^{d} [\hat{\mathbf{H}}'_{i1} | \hat{\mathbf{H}}'_{i2}] \cdot F_i \tag{20}$$

for any $l \in [1;d]$.

Since we are dealing with quasi-cyclic codes, the matrices $\hat{\mathbf{H}}_{l1}$, for $l \in [1;d]$ are block matrices of circulant matrices (block-circulant). Each of these circulant matrices can be uniquely represented by a polynomial. Now considering the determinant of the block-circulant matrix as a polynomial and assuming it behaves as a random polynomial, we can use the result from Proposition 1. Hence, the probability that the block-circulant matrix is invertible is given by:

$$Pr_c = \frac{\prod_{i=1}^{\tau} (q^{d_i \alpha_i} - q^{d_i(\alpha_i - 1)})}{q^{n-k}} \tag{21}$$

where $x^{n-k} - 1 = p_1^{\alpha_1}(x) \cdot \dots \cdot p_\tau^{\alpha_\tau}(x)$ is the factorization over $\mathbb{F}_q$.

Now the probability that at least one of the matrices $\hat{\mathbf{H}}_{l1}$, for $l \in [1;d]$ is invertible is

$$Pr_{ek} = 1 - (1 - Pr_c)^d. \tag{22}$$

Eq. (22) gives the probability that an equivalent key exists.

Remark 2. We should emphasize that forcing the matrices $\hat{\mathbf{H}}_{l1}$, for $l \in [1;d]$ to be singular in the design of the scheme does not help prevent our attack. It only requires a small modification on the equivalent key. The rest of the attack is essentially the same.

5.2 A Quantum-Enhanced Attack

Since cryptosystems based on LRPC codes are considered post-quantum, it makes sense to estimate their security against quantum-enhanced attack, using the full power of quantum computers. A second look at our attack immediately shows a possibility for a quantum speed-up using Grover's algorithm [20]. Recall that Grover's algorithm searches for an item in an unsorted database satisfying a given condition. In our attack, a huge component is represented by searching for elements in appropriate kernels (see Eq. (10)). The rest is just solving linear equations. It follows that it is straightforward to apply Grover's algorithm, and we can expect roughly a quadratic speed-up in the search phase, i.e., we can find a vector in the kernel with a number of trials which is about

$$T_e = O(\sqrt{q^{rd-1}}). \tag{23}$$

We could also think to apply a quantum algorithm for solving the linear systems like for example HHL [23]. However, in our case, there is no benefit from doing so, since HHL requires a large amount of quantum memory, and is not particularly suited for the systems that we have. Therefore, we decided to simply "Groverize" our attack. For the design of the oracle, we can reuse [42] with a small modification. The modification is that in [42] the authors use multiple variables and the cost in number of quantum gates is $2m(n^2 + 2n)$. However, we are using a linear system which means that for our attack, using an $m \times n$ matrix and a vector of length n, we have a gate complexity of mn.

6 Case Study: McNie

6.1 Recovering the Secret Key in McNie

Recall the generic structure of our attack from Algorithm 2. The two main procedures are SolveH and SolveF. The first recovers the coefficients of a key equivalent to $\mathbf{H}$ and is generic for LRPC cryptosystems. SolveF, instead, finds the secret basis F and the rest of the secret key. In the case of McNie the secret key is $\mathsf{sk} = (\mathbf{H}, \mathbf{S})$ where $\mathbf{S}$ is an invertible $(n-k) \times (n-k)$ matrix.

Recall that for McNie (see Figure 2) it is true that:

$$\mathbf{c}_1\mathbf{H}^\top - \mathbf{c}_2\mathbf{S}^{-1} = \mathbf{e}\mathbf{H}^\top$$

Suppose that in SolveH we have recovered an equivalent key $\mathbf{H}' = \mathbf{T} \cdot \mathbf{H}$. Multiplying the previous equation by $\mathbf{T}^\top$ we obtain

$$\mathbf{c}_1(\mathbf{T} \cdot \mathbf{H})^\top - \mathbf{c}_2((\mathbf{T}^\top)^{-1}\mathbf{S})^{-1} = \mathbf{e}(\mathbf{T} \cdot \mathbf{H})^\top$$

i.e., $\mathsf{sk}' = (\mathbf{H}', \mathbf{S}') = (\mathbf{T} \cdot \mathbf{H}, (\mathbf{T}^\top)^{-1}\mathbf{S})$ is an equivalent secret key for $\mathsf{sk} = (\mathbf{H}, \mathbf{S})$, so we can continue with recovering $\mathbf{S}'$ instead of $\mathbf{S}$. Now we can rewrite the previous equation as

$$(\mathbf{c}_1 - \mathbf{e})\mathbf{H}'^\top = \mathbf{c}_2\mathbf{S}'. \tag{24}$$

Notice that if we know a triple $(\mathbf{m}, \mathbf{e}, (\mathbf{c}_1, \mathbf{c}_2))$ of message, error and ciphertext (which of course anyone can generate from the public key), once the coefficients of $\mathbf{H}'$ are known, the remaining unknowns in Equation (24) are the $(n-k)^2$ coefficients of $\mathbf{S}'$ and the d basis elements of F, all in $\mathbb{F}_{q^m}$. Furthermore, seen as a system of equations over $\mathbb{F}_{q^m}$ in these unknowns, Equation (24) is a system of $n-k$ linear equations. Hence, by generating at least $\lceil \frac{(n-k)^2+d}{n-k} \rceil = n-k+1$ valid triples $(\mathbf{m}_i, \mathbf{e}_i, (\mathbf{c}_1, \mathbf{c}_2)_i)$ we can form an overdetermined system in $(n-k)^2 + d$ variables. Solving this system will give the remaining parts of the secret key. Thus in the case of McNie we can define SolveF as the procedure that solves this system. Its cost is $Cost(\mathsf{SolveF}) = ((n-k)^2 + d)^3$.

Based on the results from this Section and Sections 3 and 5 we have estimated our attack complexity for the McNie parameters given in their NIST submission [17]. The results are given in Table 1. We recall that we do not exploit any specific properties of McNie as the attack in [26] which dramatically decreases the security of the scheme, since our goal is that of providing a general reaction attack against LPRC cryptosystems.

We have implemented the attack using SAGE Math [46], and verified that, under the assumption that the kernel vectors have been found, an equivalent key can be successfully found.

Table 1. McNie parameters proposed to the first round of the NIST competition, complexities and success probability of our proposed attack

n	k	d	r	q	m	Dec. Failure	Security (bits)	Attack (Classical)	Attack (Quantum)	t	Success $Pr_s \cdot Pr_{ek}$
93	62	3	5	2	37	2^{-17}	128	128.8	82.8	8	$0.5 \cdot 0.8$
105	70	3	5	2	37	2^{-20}	128	139.7	83.7	8	$2^{-10} \cdot 0.74$
111	74	3	7	2	41	2^{-17}	192	188	108	8	$0.08 \cdot 0.87$
123	82	3	7	2	41	2^{-20}	192	189	109	8	$2^{-15} \cdot 0.875$
111	74	3	7	2	59	2^{-17}	256	188	108	8	$1 \cdot 0.875$
141	94	3	9	2	47	2^{-20}	256	238	134	8	$2^{-22} \cdot 0.875$
60	30	3	5	2	37	2^{-16}	128	166.5	96.5	10	$0.63 \cdot 0.67$
72	36	3	5	2	37	2^{-21}	128	168	98	10	$2^{-20} \cdot 0.75$
76	38	3	7	2	41	2^{-18}	192	228.3	128.3	10	$2^{-6} \cdot 0.875$
84	42	3	7	2	41	2^{-21}	192	229	129	10	$2^{-37} \cdot 0.623$
76	38	3	7	2	53	2^{-18}	256	228.3	128.3	10	$1 \cdot 0.875$
88	44	3	8	2	47	2^{-20}	256	259.5	144.5	10	$2^{-8} \cdot 0.875$

References

1. N. Aragon, J.-C. Blazy, Olivier Deneuville, P. Gaborit, A. Hauteville, O. Ruatta, J.-P. Tillich, and G. Zémor. Lake, Dec. 2017. NIST Post-Quantum Cryptography Project: First Round Candidate Algorithms.
2. N. Aragon, J.-C. Blazy, Olivier Deneuville, P. Gaborit, A. Hauteville, O. Ruatta, J.-P. Tillich, and G. Zémor. Locker, Dec. 2017. NIST Post-Quantum Cryptography Project: First Round Candidate Algorithms.
3. N. Aragon, O. Blazy, P. Gaborit, A. Hauteville, and G. Zémor. Durandal: A rank metric based signature scheme. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, pages 728–758, 2019.
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. T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani. Reducing key length of the McEliece cryptosystem. In B. Preneel, editor, *AFRICACRYPT*, volume 5580 of *Lecture Notes in Computer Science*, pages 77–97. Springer, 2009.
6. D. J. Bernstein, T. Chou, T. Lange, I. von Maurich, R. Niederhagen, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, and W. Wang. *https://classic.mceliece.org/.*
7. L. Bettale, J.-C. Faugère, and L. Perret. Cryptanalysis of HFE, multi-HFE and variants for odd and even characteristic. *Designs, Codes and Cryptography*, 69(1):1–52, Oct 2013.
8. J. F. Buss, G. S. Frandsen, and J. O. Shallit. The computational complexity of some problems of linear algebra. *Journal of Computer and System Sciences*, 58(3):572–596, 1999.
9. N. T. Courtois. Efficient zero-knowledge authentication based on a linear algebra problem MinRank. In *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 402–421. Springer, 2001.

10. J.-C. Faugère, D. Gligoroski, L. Perret, , S. Samardjiska, and E. Thomae. A polynomial-time key-recovery attack on MQQ cryptosystems. In J. Katz, editor, *Public-Key Cryptography – PKC 2015*, volume 9020, pages 150–174, 2015. https://eprint.iacr.org/2014/811.
11. J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298. Springer, 2010.
12. E. M. Gabidulin. Theory of codes with maximum rank distance. *Problemy Peredachi Informatsii*, 21(1):3–16, 1985.
13. 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.
14. P. Gaborit, G. Murat, O. Ruatta, and G. Zémor. Low rank parity check codes and their application to cryptography. In *Proceedings of the Workshop on Coding and Cryptography WCC*, volume 2013, 2013.
15. P. Gaborit, O. Ruatta, J. Schrek, and G. Zémor. New results for Rank-Based cryptography. In D. Pointcheval and D. Vergnaud, editors, *Progress in Cryptology – AFRICACRYPT 2014*, pages 1–12, Cham, 2014. Springer International Publishing.
16. R. G. Gallager. *Low-Density Parity-Check Codes.* PhD thesis, M.I.T., 1963.
17. L. Galvez, J.-L. Kim, M. J. Kim, Y.-S. Kim, and N. Lee. McNie: Compact Mceliece-Niederreiter Cryptosystem, Dec. 2017. NIST Post-Quantum Cryptography Project: First Round Candidate Algorithms.
18. L. Goubin and N. T. Courtois. Cryptanalysis of the TTM cryptosystem. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 44–57. Springer, 2000.
19. R. M. Gray. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2006.
20. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.
21. Q. Guo, T. Johansson, and P. Stankovski. *A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors*, pages 789–815. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
22. C. Hall, I. Goldberg, and B. Schneier. Reaction attacks against several public-key cryptosystem. In V. Varadharajan and Y. Mu, editors, *Information and Communication Security*, pages 2–12, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.
23. A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.
24. J.-L. Kim, Y.-S. Kim, L. Galvez, M. J. Kim, and N. Lee. McNie: A code-based public-key cryptosystem. *arXiv preprint arXiv:1812.05008*, 2018.
25. A. Kipnis and A. Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In *Annual International Cryptology Conference*, pages 19–30. Springer, 1999.
26. T. S. C. Lau and C. H. Tan. Key recovery attack on McNie based on low rank parity check codes and its reparation. In *Advances in Information and Computer Security - 13th International Workshop on Security, IWSEC 2018, Sendai, Japan, September 3-5, 2018, Proceedings*, pages 19–34, 2018.
27. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, Jan. 1978.

28. C. Melchior Aguilar, N. Aragon, M. Bardet, S. Bettaieb, L. Bidoux, J.-C. Blazy, Olivier Deneuville, P. Gaborit, A. Hauteville, A. Otmani, O. Ruatta, J.-P. Tillich, and G. Zémor. Locker, Dec. 2017. NIST Post-Quantum Cryptography Project: First Round Candidate Algorithms.
29. C. Melchior Aguilar, N. Aragon, S. Bettaieb, L. Bidoux, J.-C. Blazy, Olivier Deneuville, P. Gaborit, A. Hauteville, and G. Zémor. Ouroboros-R, Dec. 2017. NIST Post-Quantum Cryptography Project: First Round Candidate Algorithms.
30. L. Minder and A. Shokrollahi. Cryptanalysis of the Sidelnikov cryptosystem. In M. Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 347–360. Springer, 2007.
31. R. Misoczki and P. S. L. M. Barreto. Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography*, pages 376–392, 2009.
32. R. Misoczki, J.-P. Tillich, N. Sendrier, and P. L. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *IEEE International Symposium on Information Theory – ISIT'2013*, pages 2069–2073, Istambul, Turkey, 2013. IEEE.
33. R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes, 2012.
34. A. Nicolas and G. Philippe. A key recovery attack against LRPC using decryption failures. In *International Workshop on Coding and Cryptography (WCC 2019)*, Saint-Jacut-de-la-Mer, Norway, 2019.
35. H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.
36. A. Nilsson, T. Johansson, and P. Stankovski. Error amplification in code-based cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):238–258, Nov. 2018.
37. *https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization.*
38. *https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions.*
39. E. Persichetti. Compact McEliece keys based on quasi-dyadic Srivastava codes. *Journal of Mathematical Cryptology*, 6(2):149–169, 2012.
40. E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions*, IT-8:S5–S9, 1962.
41. P. Santini, M. Battaglioni, F. C. Chiaraluce, and M. Baldi. Analysis of reaction and timing attacks against cryptosystems based on sparse parity-check codes. 2019.
42. P. Schwabe and B. Westerbaan. Solving Binary MQ with Grover's Algorithm. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 303–322. Springer, 2016.
43. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
44. V. M. Sidelnikov. A public-key cryptosystem based on binary Reed-Muller codes. *Discrete Mathematics and Applications*, 4(3):191 – 208, 1994.
45. V. M. Sidelnikov and S. O. Shestakov. On insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics and Applications*, 2(4):439 – 444, 1992.
46. The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 8.7)*, 2019. `https://www.sagemath.org`.