IEICE TRANS. FUNDAMENTALS, VOL.E101–A, NO.10 OCTOBER 2018

PAPER

Improving the Efficiency of a Reaction Attack on the QC-MDPC McEliece*,**

Thales BANDIERA PAIVA[†a)], ***Nonmember and*** **Routo TERADA**[†b)], ***Member***

SUMMARY The QC-MDPC McEliece scheme was considered one of the most promising public key encryption schemes for efficient post-quantum secure encryption. As a variant of the McEliece scheme, it is based on the syndrome decoding problem, which is a hard problem from Coding Theory. Its key sizes are competitive with the ones of the widely used RSA cryptosystem, and it came with an apparently strong security reduction. For three years, the scheme has not suffered major threats, until the end of 2016, at the Asiacrypt, when Guo, Johansson, and Stankovski presented a reaction attack on the QC-MDPC that exploits one aspect that was not considered in the security reduction: the probability of a decoding failure to occur is lower when the secret key and the error used for encryption share certain properties. Recording the decoding failures, the attacker obtains information about the secret key and then use the information gathered to reconstruct the key. Guo et al. presented an algorithm for key reconstruction for which we can point two weaknesses. The first one is that it cannot deal with partial information about the secret key, resulting in the attacker having to send a large number of decoding challenges. The second one is that it does not scale well for higher security levels. To improve the attack, we propose a key reconstruction algorithm that runs faster than Guo's et al. algorithm, even using around 20% less interactions with the secret key holder than used by their algorithm, considering parameters suggested for 80 bits of security. It also has a lower asymptotic complexity which makes it scale much better for higher security parameters. The algorithm can be parallelized straightforwardly, which is not the case for the one by Guo et al.

key words: *QC-MDPC McEliece, post-quantum cryptography, reaction attack*

1. Introduction

In 1994, Shor [1] published two algorithms for quantum computers that solve efficiently two critical problems in modern cryptography: the discrete logarithm and integer factorization problems. With the engineering progress on building larger quantum computers, the main cryptographic schemes used today become more vulnerable. This seeds the quest for finding cryptographic primitives that do not rely on the hardness of these problems, which is an active research area known as post-quantum cryptography [2]. The need for post-quantum secure standards is recognized by the National Institute of Standards and Technology (NIST), which is calling for proposals [3].

The McEliece scheme [4] relies on the syndrome decoding problem, which is known to be $\mathcal{NP}$-hard [5]. This scheme was developed in the same decade as the RSA [6], but it was put aside because the public key size of around 100kB, for a security level of 100 bits, were intractably large at that time. Despite its huge keys, both encryption and decryption are more efficient than the corresponding operations of the RSA and elliptic curves [7] schemes.

In the McEliece scheme, Alice's secret key is a linear code $\mathcal{G}$ capable of correcting t errors, for which Alice knows an efficient decoder. Her public key is the generator matrix of $\mathcal{G}$, possibly scrambled, in such a form that it is unfeasible for an attacker to use this matrix for building an efficient decoder for $\mathcal{G}$. To send Alice a message $\mathbf{m}$, we encode $\mathbf{m}$ using her public key and add t intentional errors to this encoding, giving us $\mathbf{c}$. We now can safely send her $\mathbf{c}$, because Alice is the only one that has an efficient decoder for $\mathcal{G}$, which she uses to obtain $\mathbf{m}$ from $\mathbf{c}$.

One of the post-quantum cryptography research trends is to lower the public key sizes of the McEliece scheme by choosing families of error correcting codes that allow compact representation, without compromising security. Originally McEliece suggested the use of binary irreducible Goppa codes [8], which are still considered secure, at the expense of huge keys. The majority of other proposed code families rely on a quasi-cyclic or quasi-dyadic structure for compact representation [9]–[12]. Although they obtain compact keys, most of them were shown insecure [13]–[15].

In 2013, a new variant of the McEliece scheme that uses quasi-cyclic moderate-density parity-check codes (QC-MDPC) was presented by Misoczki et al. [16]. This variant promises extremely compact public keys of only 4801 bits for a security level of 80 bits, and has an apparently strong security reduction. The European initiative PQCRYPTO, which supports the development of post-quantum cryptography, considered this variant as a serious candidate for a post-quantum secure standard in the 2015 revision of its paper with recommendations for post-quantum secure systems [17].

Until the end of 2016, the QC-MDPC has not suffered critical attacks. However at Asiacrypt 2016 Guo, Johansson, and Stankovski [18] present an efficient reaction attack for key recovery on the QC-MDPC. This attack is based on the fact that QC-MDPC decoders can fail. When a decoding failure occurs, the receiver asks the sender to resend the message, which hopefully, will be encrypted with an error pattern that the decoder will be able to correct. The main observation of

Manuscript received September 22, 2017.
Manuscript revised March 22, 2018.

[†]The authors are with the Dept. of Computer Science, Universidade de São Paulo, São Paulo State, Brazil.

[*]This work was supported by CAPES grant number 00.889.834/0001-08.
[**]This work was supported by FAPESP grant number 2015/01587-0, CNPq grant number 442014/2014-7.

a) E-mail: tpaiva@ime.usp.br
b) E-mail: rt@ime.usp.br

DOI: 10.1587/transfun.E101.A.1676

the authors is that the probability that the decoder fails when correcting the error **e** is significantly smaller when **e** and the secret key share some certain properties.

The proposed attack is done in two parts. In the first part, an attacker Eve sends a number of ciphertexts to her target Alice, and records for which error patterns the decoder failed or succeeded, to obtain some information, which they called the spectrum, about Alice's secret key. The second part is the reconstruction part, where Eve, without interacting any further with Alice, tries to build Alice's secret key with the information gathered in the first part.

We address the following three problems of Guo's et al. reconstruction algorithm:

1. it cannot recover the secret key when the information about it is incomplete;
2. the number of operations needed to find the key grows very fast with respect to the security level;
3. it is recursive in nature, and it is not obvious how to parallelize it, neither how much one gains by doing it.

Our proposed algorithm is iterative, it can be parallelized straightforwardly, and it is based only on linear algebra over $\mathbb{F}_2$. Our algorithm is able to deal with less information, therefore an attacker needs only to perform a lower number of decoding trials. This is particularly important because one way to protect the scheme against the reaction attack is to limit the lifetime of a secret key, which must be determined as a function of the number of decoding trials that are sufficient for a successful attack. The algorithm also has lower asymptotic complexity, which makes it scale a lot better for higher security parameters.

Upon preparation of this paper, it came to our knowledge that the linear relation which we exploit was also used by Rossi et al. [19] in their recent paper on a side-channel attack against QcBits [20]. However our algorithm is significantly different because it uses more efficiently the information given by the reactions of the secret key holder in the first part of the attack. This not only makes it more efficient, but it also allows the algorithm to recover the key with less interaction with the secret key holder. Further, our analysis of the algorithm performance is more detailed.

This paper is organized as follows. Section 2 presents some preliminary concepts. A review of the QC-MDPC McEliece is done in Sect. 3, which ends with a discussion on the performance of the attack by Guo et al. [18]. The presentation of our contributions starts in Sect. 4, where our proposed reconstruction algorithm and its underlining ideas are shown. The probabilistic analysis of the proposed algorithm is in Sect. 5. Sect. 6 presents the experimental results. The concluding remarks are in Sect. 7.

2. Preliminaries

We review some concepts from Coding Theory.

Definition 1 (Linear codes): A *binary* $[n, k]$*-linear code* is a k-dimensional linear subspace of $\mathbb{F}_2^n$.

Definition 2 (Generator and parity-check matrix): Let C be a binary $[n, k]$-linear code. If C is the linear subspace spanned by the rows of a matrix $\mathbf{G}$ of $\mathbb{F}_2^{k\times n}$, we say that $\mathbf{G}$ is a *generator matrix* of C. Similarly, if C is the kernel of a matrix $\mathbf{H}$ of $\mathbb{F}_2^{(n-k)\times n}$, we say that $\mathbf{H}$ is a *parity-check matrix* of C.

Since we are interested in codes defined by their parity-checks, we usually consider the co-dimension $r = n - k$ of the code, instead of its dimension k.

Definition 3: The *Hamming weight* of a vector $\mathbf{v}$, denoted by weight($\mathbf{v}$), is the number of its non-null entries.

Definition 4: The circular distance between the indexes i and j in a vector of length r is

$$\mathrm{dist}_r(i, j) = \begin{cases} |i - j| & \text{if } |i - j| < \lfloor r/2 \rfloor, \\ r - |i - j| & \text{otherwise.} \end{cases}$$

We next define the spectrum of a vector, which is a crucial concept for the rest of the paper. The importance of the spectrum for the attack comes from the fact that it is precisely the spectrum of the key that can be recovered by a reaction attack. Intuitively, the spectrum of a binary vector $\mathbf{v}$ is the set of circular distances that occur between two non-null entries of $\mathbf{v}$. Note that spectrum of a vector is invariant for its circular shifts.

Definition 5: Let $\mathbf{v} = [v_1, v_2, \ldots, v_r]$ be an element of $\mathbb{F}_2^r$. Then the spectrum of $\mathbf{v}$ is the set

$$\sigma(\mathbf{v}) = \{\mathrm{dist}_r(i, j) : i \neq j,\ v_i = 1,\ \text{and } v_j = 1\}.$$

When describing the proposed algorithm, it is useful to consider the circular shift of a set of indexes, defined next.

Definition 6: Let $Z \subset [r] = \{1, 2, \ldots, r\}$ be a set of possible indexes of a vector in $\mathbb{F}_2^r$. Then the circular shift of Z by p positions is the set

$$Z^{(p)} = \{i + p \mod r : i \in Z\}.$$

3. The QC-MDPC McEliece

3.1 The QC-MDPC McEliece

In 2013, a new variant of the McEliece scheme was proposed by Misoczki et al. [16]. This variant uses quasi-cyclic codes with moderate density parity-check matrices, in contrast with Gallager's low density parity-check codes (LDPC) [21]. The idea is to increase the density of the parity-check matrix to make the code stronger against attacks based on the search for low weight codewords [22], but keeping a sufficiently low density so that efficient LDPC decoders can still be used.

Definition 7 (QC-MDPC): An (n, r, w)-QC-MDPC code is a quasi-cyclic linear code of length n, co-dimension r which divides n, that has a sparse parity-check matrix $\mathbf{H}$ with row weights $w = O\left(\sqrt{n \log n}\right)$, and formed by $n_0 = n/r$ cyclic

Table 1 Suggested QC-MDPC parameters for some security levels [16].

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

blocks.

The parameters suggested by Misoczki et al. [16] are shown in Table 1. The suggested parameters for the QC-MDPC McEliece entail extremely small keys when compared with the key size of hundreds of megabytes for the original McEliece scheme [23].

Until the end of 2016, the QC-MDPC McEliece was considered one of the most promising candidates for efficient code based cryptography. The lack of algebraic structure in the code, and the apparently strong security reduction of the scheme, suggested that it would be hard to compromise the security of the QC-MDPC codes. One of the main initiatives for post-quantum cryptography, the European PQCRYPTO, considered the QC-MDPC McEliece as a strong candidate for long-term secure communication [17].

We now show the algorithms for key generation, encryption, and decryption. The encryption and decryption algorithms can be adapted to support CCA2 security, for example, using one of Kobara and Imai conversions [24].

3.1.1 Key Generation

Suppose we want to generate keys for the QC-MDPC McEliece with the security level λ. Let n_0, n, r, and w be parameters supporting this security level λ, which can be taken from Table 1.

First generate a random QC-MDPC code by choosing at random a vector $\mathbf{h}$ from $\mathbb{F}_2^n$, such that the weight of $\mathbf{h}$ is w. Break $\mathbf{h}$ into $n_0 = n/r$ equal parts $\mathbf{h} = [\,\mathbf{h}_0\,|\,\mathbf{h}_1\,|\,\ldots\,|\,\mathbf{h}_{n_0-1}\,]$. Then build the parity-check matrix $\mathbf{H}$ as

$$\mathbf{H} = [\,\mathbf{H}_0\,|\,\mathbf{H}_1\,|\,\ldots\,|\,\mathbf{H}_{n_0-1}\,],$$

where each $\mathbf{H}_i$ is the cyclic matrix with $\mathbf{h}_i$ as its first row. It is required that the block $\mathbf{H}_{n_0-1}$ is invertible. If it is not, restart the key generation procedure by picking another $\mathbf{h}$ at random.

We now build the generator matrix as

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \left|\begin{matrix} \left(\mathbf{H}_{n_0-1}^{-1}\cdot\mathbf{H}_0\right)^T \\ \left(\mathbf{H}_{n_0-1}^{-1}\cdot\mathbf{H}_1\right)^T \\ \vdots \\ \left(\mathbf{H}_{n_0-1}^{-1}\cdot\mathbf{H}_{n_0-2}\right)^T \end{matrix}\right. \end{bmatrix}, \tag{1}$$

and one can check that $\mathbf{G}\mathbf{H}^T = \mathbf{0}$ with a simple evaluation. Since each $\mathbf{H}_i$ is cyclic, each product $\left(\mathbf{H}_{n_0-1}^{-1}\mathbf{H}_i\right)^T$ is also cyclic.

The secret key is the matrix $\mathbf{H}$ and the public key is the matrix $\mathbf{G}$. Both of the matrices admit compact representation because of their cyclic blocks.

Note that if one attacker can recover the cyclic matrix $\mathbf{H}_{n_0-1}$, which is invertible by construction, then she can recover all the other cyclic matrices $\mathbf{H}_i$ using simple linear algebra. In fact, since each matrix $\mathbf{H}_i$ has high rank with high probability, if an attacker can recover any cyclic matrix $\mathbf{H}_j$, she might be able to recover $\mathbf{H}_{n_0-1}$ by finding its first line $\mathbf{h}_{n_0-1}$, which is a low weight solution of the linear equation

$$\mathbf{h}_{n_0-1}\mathbf{B}_j = \mathbf{h}_j,$$

where $\mathbf{h}_j$ is the first line of $\mathbf{H}_j$, and $\mathbf{B}_j = \mathbf{H}_{n_0-1}^{-1}\mathbf{H}_j$ is known. As such, it is desirable that the hardness to recover each block be approximately the same. This implies that the weights of the vectors $\mathbf{h}_i$ must not differ too much, which is easily done by fixing the weight of each block as $\hat{w} = w/n_0$.

3.1.2 Encryption

Let the vector $\mathbf{m}$ of $\mathbb{F}_2^{(n-r)}$ be the message to be encrypted. First encode the word $\mathbf{m}$ obtaining $\hat{\mathbf{c}} = \mathbf{m}\mathbf{G}$. Then add a random vector $\mathbf{e}$ of weight t to $\hat{\mathbf{c}}$ to get the ciphertext $\mathbf{c} = \hat{\mathbf{c}} + \mathbf{e}$.

3.1.3 Decryption

Since the ciphertext is just a corrupted codeword, decrypting $\mathbf{c}$ is equivalent to correct the errors from $\mathbf{c}$. This is achieved with iterative decoders [21], which can be of two types: based on either hard or soft decision. Soft decision based decoders have better error correction capability, but are less efficient. It is usual to decode QC-MDPC codes with hard decision based decoders, because the main problem here is not efficient communication, but secure communication. Therefore one does not need to correct a lot of errors, but only enough to make the scheme secure. These decoders are called *bit-flipping decoders*, and some of its variants were studied by Maurich et al. [25] in terms of performance and error correction capability.

3.2 Reaction Attack

The reaction attack is based on the observation that the probability of a failure to occur when decoding a vector $\mathbf{c} = \hat{\mathbf{c}} + \mathbf{e}$ is lower when the spectrums of the blocks of $\mathbf{h}$ are similar to the corresponding spectrums of the blocks of the error $\mathbf{e}$.

When a decoding failure occurs, the receiver asks the sender to resend the message. Therefore the attacker knows when a failure occurs, and by sending decoding challenges to the secret key holder, she can get some structural information about the key.

The attack is done in two parts. In the first, the attacker collects information about the key by sending challenge ciphertexts to the secret key holder Alice, and recording Alice's reactions when she tries to decode these ciphertexts. The first part is the only part where the attacker needs to interact with the secret key holder. In the second part, the attacker tries to reconstruct the key using the information previously collected. These two parts are described next.

3.2.1 Spectrum Recovery

This is the part where the reactions of the receiver are exploited. Let Alice be the secret key holder who we want to attack. The idea is to send Alice valid ciphertexts, and record when she could not decode the challenge. Since the attacker generated all ciphertexts, then, for each one of them, he knows the error $\mathbf{e} = [\,\mathbf{e}_0 \,|\, \mathbf{e}_1\,]$ that was added to the codeword, and thus he can compute both $\sigma(\mathbf{e}_0)$ and $\sigma(\mathbf{e}_1)$. Unfortunately, CCA2 conversions, such as the ones by Kobara and Imai [24], do not protect the algorithm against this recovery procedure because each challenge ciphertext is valid.

A slightly modified version of Guo's et al. spectrum recovery algorithm is given by Algorithm 1. The spectrum recovery procedure by Guo et al. only recovers the spectrum of $\mathbf{h}_1$ because their reconstruction algorithm gains nothing by also considering the spectrum of $\mathbf{h}_0$. Guo et al. chose to recover $\mathbf{h}_1$ because, from it, one can completely recover the secret key by using simple linear algebra, as discussed in the end of Sect. 3.1.1. In contrast, our reconstruction algorithm uses information on both blocks of $\mathbf{h}$, therefore the presented spectrum recovery algorithm simultaneously recovers information on the spectrums of both $\mathbf{h}_0$ and $\mathbf{h}_1$. Further, simultaneously recovering both spectrums does not affect significantly the complexity of the spectrum recovery algorithm because the cost of the decryption dominates the costs involved in estimating the spectrums.

To study how large should be the number of decoding trials M for a successful attack, we introduce the concept of d-exclusion obtained by the spectrum recovery algorithm. It is simply the maximum number d of distances which we can consider outside of the spectrum based only on the recovered vector of estimated probabilities of failure. The formal definition is given next.

Definition 8: We say that the output $\mathbf{p}_0$ of the spectrum recovery algorithm obtains a d-exclusion for the spectrum of a vector $\mathbf{h}_i$ if there exists a number $p \in (0, 1)$ such that

1. $\mathbf{p}_0[d] \geq p$ only if $d \notin \sigma(\mathbf{h}_i)$;
2. $\#\{\mathbf{p}_0[d] \geq p\} = d$.

When $d = \lfloor r/2 \rfloor - |\sigma(\mathbf{h}_i)|$, that is, d is the number of distances outside the spectrum of $\mathbf{h}_i$, we say that $\mathbf{p}_0$ obtains total exclusion.

Algorithm 1: Estimating the probabilities of error for distances in $\sigma(\mathbf{h}_0)$ and $\sigma(\mathbf{h}_1)$

Data: n, r, w, t parameters of the QC-MDPC code
$\mathcal{D}$ target's decoder reaction oracle (1 for success, 0 for fail)
M number of decoding challenges
Result: $\mathbf{p}_0, \mathbf{p}_1$ estimated probabilities of error for distances in $\sigma(\mathbf{h}_0)$ and $\sigma(\mathbf{h}_1)$

```
begin
    a0, b0, a1, b1 ← zero-initialized arrays with ⌊r/2⌋ entries each
    for each decoding trial i = 1, 2, ..., M do
        c ← a random ciphertext encrypted with error e = [ e0 | e1 ] of weight t
        v = D(c)
        for each distance d in σ(e0) do
            a0[d] ← a0[d] + v
            b0[d] ← b0[d] + 1
        for each distance d in σ(e1) do
            a1[d] ← a1[d] + v
            b1[d] ← b1[d] + 1
    p0, p1 ← zero-initialized array with ⌊r/2⌋ positions
    for each distance d in {1, 2, ..., ⌊r/2⌋} do
        p0[d] ← a0[d]/b0[d]
        p1[d] ← a1[d]/b1[d]
    return p0 and p1
```

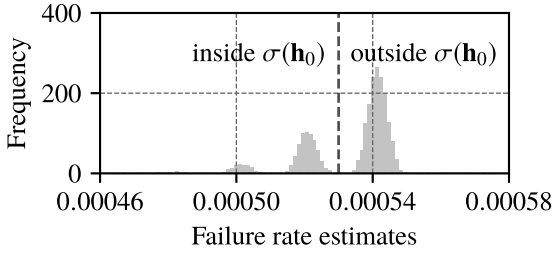

Fig. 1 Histogram of failure rates for 200M decoding trials.

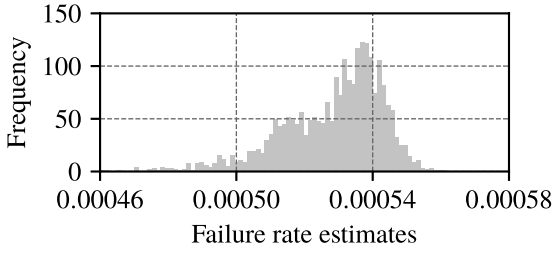

Fig. 2 Histogram of failure rates for 30M decoding trials.

One of the weaknesses of the reconstruction algorithm by Guo et al., which we discuss in Sect. 3.2.2, is that its efficiency depends on an almost total exclusion. As such, the number of decoding challenges M must be set so that it occurs. Guo et al. empirically evaluated that, for 80 bits security parameters, total exclusion usually occurs for values of M between 203 million and 356 million decoding challenges. Figure 1 illustrate a total exclusion, while Fig. 2 shows the result of 30 million decoding challenges, where

it is not possible to clearly classify a distance as inside or outside the spectrum.

When total exclusion for $\mathbf{h}_1$ occurs, the authors suggest the use of some clustering procedure for gathering the distances inside $\sigma(\mathbf{h}_1)$. Then $\sigma(\mathbf{h}_1)$ is passed to the reconstruction algorithm, which is described next.

3.2.2 Key Reconstruction

Given $\sigma(\mathbf{h}_1)$, the reconstruction algorithm by Guo et al. is a simple pruned depth-first search in a tree. As such, it is recursive, and it is not obvious how one should parallelize it. Its description is given as Algorithm 2.

Guo's et al. reconstruction algorithm as presented assumes that one knows the weight $\hat{w}$ of $\mathbf{h}_1$, which might not be the case for a real attacker. But one can adapt this algorithm to consider an interval of weights for $\mathbf{h}_1$. Guo et al. consider $\hat{w} = w/n_0$, since making the weights of the cyclic blocks equal is a secure approach for generating the secret key.

Algorithm 2: Guo's et al. key recovery algorithm

Data: n, r, w, t parameters of the QC-MDPC code
$\hat{w} = w/n_0$ the weight of $\mathbf{h}_1$
$\sigma(\mathbf{h}_1)$ the spectrum of $\mathbf{h}_1$
V the partially recovered support of a shift of $\mathbf{h}_1$ (initially set to $\{1, s_1 + 1\}$, where $s_1 \in \sigma(\mathbf{h}_1)$)
Result: $\mathbf{V}$ the support of some shift of $\mathbf{h}_1$, or $\perp$ if $\sigma(\mathbf{h}_1)$ is an invalid spectrum

```
1  begin
2      if |V| = ŵ then
3          if V is the support of a shift of h1 then
4              return V
5          else
6              return ⊥
7      for each position j = 2, ..., r which are not in V do
8          if dist_r(v, j) ∈ σ(h1) for all v in V then
9              Add j to V
10             ret ← recursive call with the updated set V
11             if ret ≠⊥ then
12                 return V
13             Remove j from V
14     return ⊥
```

We explain how the test in line 3 can be done. We want to test if a vector $\mathbf{v}$ of weight $\hat{w}$ and support V is a shift of $\mathbf{h}_1$. Consider $\mathbf{B} = (\mathbf{H_1}^{-1}\mathbf{H_0})^T$ the right part of the public matrix $\mathbf{G}$. Compute the vector $\mathbf{u} = (\mathbf{B}^{-1})^T\mathbf{v}$. If the weight of $\mathbf{u}$ is $\hat{w}$, then there are two possibilities. One, which has high probability, is that $\mathbf{v}$ is a shift of $\mathbf{h}_1$ and $\mathbf{u}$ is a shift of $\mathbf{u}$ by the same number of positions. The other possibility is that we can use $[\,\mathbf{v}\,|\,\mathbf{u}\,]$ to build an alternative moderate density quasi-cyclic parity-check matrix for the secret code. The key found is valid in both cases.

The main argument by Guo et al. for the efficiency of their algorithm is that unfruitful branches are pruned relatively early in the search. More specifically, if the position of a small number of non-null entries in the first half of $\mathbf{h}_1$ are known, then all the other positions in the support of $\mathbf{h}_1$ are easily determined.

Following Guo's et al. notation, we now describe their analysis for estimating the number Γ of maximum paths in the search tree to be explored until the key is found. Let α be the fraction of the $\lfloor r/2 \rfloor$ possible distances that belong to the spectrum $\sigma(\mathbf{h}_1)$, that is $\alpha = |\sigma(\mathbf{h}_1)|/\lfloor r/2 \rfloor$. The search starts in the root, at level 0, which contains the positions 1 and $s_1 + 1$. Then it is expected that approximately $|\sigma(\mathbf{h}_1)|\alpha$ of the possible positions j in the first half of $\mathbf{h}_1$, that is $j < r/2$, are such that both $\mathrm{dist}_r(j, 1)$ and $\mathrm{dist}_r(j, s_1 + 1)$ are in $\sigma(\mathbf{h}_1)$. For each new level in the search tree, it is expected that a fraction α of the possible positions in the previous level survive the sieve imposed by line 8. Then at level ℓ in the search tree, we expect to have around $|\sigma(\mathbf{h}_1)|\alpha^\ell = \lfloor r/2 \rfloor \alpha^{\ell+1}$ nodes. Denote by ϕ the level for which each node in level ϕ has an expected number of child nodes lower than or equal to 1, that is $\phi = \min\{\ell : \lfloor r/2 \rfloor \alpha^{\ell+2} \leq 1\}$. Then, Guo's et al. [18] estimate on the number of possible paths is

$$\Gamma = \prod_{\ell=1}^{\phi} |\sigma(\mathbf{h}_1)|\alpha^\ell = \prod_{\ell=1}^{\phi} (\lfloor r/2 \rfloor \alpha)\, \alpha^\ell = \lfloor r/2 \rfloor^\phi \alpha^{\phi(\phi+3)/2}.$$

Guo et al. did not present a detailed analysis of the practical performance of their algorithm, they just state that it typically succeeds in a few minutes. However in the Asiacrypt presentation [26] of their work, Guo shows that their algorithm runs in 144 seconds on average, and it runs in 49 minutes in the worst case, for the security level of 80 bits using $n_0 = 2$ cyclic blocks. They did not consider security levels higher than 80 or other values of n_0.

The estimates on the number of paths Γ for different security levels, and supposing that the complete spectrum of $\mathbf{h}_1$ is known, are shown in Table 2. For each line of the table, the parameters α were obtained by simulations by considering the lengths of the spectrums of 5000 different vectors of length r and weight $\hat{w}$ generated at random. We can see that the number of paths grows significantly as the security level increases or higher values of n_0 are considered. This can make the use of Guo's et al. algorithm impractical for security levels higher than 80 and $n_0 > 2$.

As observed by Fabšič et al. [27], given a vector $\mathbf{v}$ with support $V = \{j_0, j_1, \ldots, j_{\hat{w}-1}\}$, for some j_i, then one can

Table 2 Estimates on the maximum number of paths Γ in the search tree used by Guo's et al. algorithm for different security parameters.

Security	n_0	r	$\hat{w} = w/n_0$	α	ϕ	Γ
80	2	4801	45	0.340528	6	$2^{25.41}$
80	3	3593	51	0.513207	10	$2^{45.55}$
80	4	3079	55	0.625694	14	$2^{67.73}$
128	2	9857	71	0.398296	8	$2^{39.70}$
128	3	7433	81	0.585685	14	$2^{74.19}$
128	4	6803	85	0.654577	18	$2^{95.62}$
256	2	32771	137	0.435026	10	$2^{61.95}$
256	3	22531	155	0.655827	21	$2^{129.28}$
256	4	20483	161	0.718417	26	$2^{166.50}$

build a vector with the same spectrum as $\mathbf{v}$, namely the vector with support $\hat{V} = \{j_0, j_1, r - j_{\hat{w}-1} + j_1, r - j_{\hat{w}-2} + j_1, \ldots, r - j_2 + j_1\}$. Suppose a valid support V with $\hat{w}$ elements is found but turns out to be an invalid key in the check of line 3 of Algorithm 2. Then one can build another set $\hat{V}$ as described above, and perform the same check of line 3 for this set $\hat{V}$. This optimization may cut in half the number of paths Guo's et al. algorithm have to explore until a key is found. For the experimental analysis of Sect. 6, our implementation of their algorithm uses this optimization; that is why our implementation is twice as efficient than Guo's et al. However, since the number of paths Γ can be very large, this optimization is not enough to make the algorithm efficient against most of the security parameters, as indicated by our experimental analysis in Sect. 6.

4. Our Reconstruction Algorithm

Up to this point, we analyzed the algorithm by Guo et al. From this section on, we describe our original results. We begin by considering the reconstruction algorithm for the case where the number of cyclic blocks in the secret matrix $\mathbf{H}$ is $n_0 = 2$, because the exposition is simpler in this case. We then discuss how to adapt the algorithm for the case where $n_0 = 2$ for $n_0 \geq 3$.

4.1 Case $n_0 = 2$

Consider the public generator matrix of a QC-MDPC with $n_0 = 2$ given as

$$\mathbf{G} = \left[\quad \mathbf{I} \quad \middle| \left(\mathbf{H}_1^{-1} \cdot \mathbf{H}_0\right)^T \right].$$

Let $\mathbf{B} = \mathbf{H}_1^{-1}\mathbf{H}_0$ be the transpose of the right block of the public generator matrix. Our main idea is to explore the relation $\mathbf{h}_1\mathbf{B} = \mathbf{h}_0$, where $\mathbf{h}_1$ and $\mathbf{h}_0$ are corresponding lines of the matrices $\mathbf{H}_1$ and $\mathbf{H}_0$, respectively. Let Z_0 and Z_1 be sets of indexes of some of the null entries of $\mathbf{h}_0$ and $\mathbf{h}_1$, respectively. Denote by $\mathbf{B}^{Z_0}$ the matrix consisting of the columns of $\mathbf{B}$ whose indexes are in Z_0. Then it should be clear that

$$\mathbf{h}_1\mathbf{B}^{Z_0} = \mathbf{0}.$$

Note that we can discard the entries of $\mathbf{h}_1$ whose indexes are in Z_1, if we discard the corresponding columns in $\mathbf{B}^{Z_0}$. Let Z_1' be the complement of Z_1 with respect to the possible indexes of $\mathbf{h}_1$. Then

$$\mathbf{h}_1^{Z_1'}\mathbf{B}_{Z_1'}^{Z_0} = \mathbf{0},$$

where $\mathbf{h}_1^{Z_1'}$ is the vector consisting of the columns of $\mathbf{h}_1$ whose indexes are in Z_1', and $\mathbf{B}_{Z_1'}^{Z_0}$ is the matrix consisting of the lines from $\mathbf{B}^{Z_0}$ whose indexes are in Z_1'. In other words, $\mathbf{h}_1^{Z_1'}$ is in the left kernel of the matrix $\mathbf{B}_{Z_1'}^{Z_0}$. Then we can compute the kernel matrix of $\mathbf{B}_{Z_1'}^{Z_0}$ and hope to find $\mathbf{h}_1^{Z_1'}$ in one of its columns. To use this in our favor, we need to solve three problems, which we discuss next.

The first problem is that if the kernel of $\mathbf{B}_{Z_1'}^{Z_0}$ is a large subspace, then finding a low weight vector in it can be very difficult. To deal with it, we must find out how large must the sets Z_0 and Z_1 be so that the kernel of $\mathbf{B}_{Z_1'}^{Z_0}$ consists only of the vector $\mathbf{h}_1$. The answer is given in Sect. 5.1, and it is that the kernel consists only of $\mathbf{h}_1$ with approximate probability $1 - 1/2^{|Z_0|+|Z_1|-r}$. Therefore we want the sum $|Z_0| + |Z_1|$ to be larger than r, but not necessarily much larger since the probability increases fast with respect to the difference.

The second problem is to find how much information we need about the spectrums of both $\mathbf{h}_0$ and $\mathbf{h}_1$ so that we can build large enough sets Z_0 and Z_1. Let us consider $\sigma(\mathbf{h}_0)$ the spectrum of $\mathbf{h}_0$. Suppose we know that s_0 is in $\sigma(\mathbf{h}_0)$, and we also know that the distances $d_1, \ldots, d_l$ are not in $\sigma(\mathbf{h}_0)$. Letting $*$ denote unknown entries, we know that there must exist a shift of $\mathbf{h}_0$ which has the following format

$$[\underbrace{0 * \ldots * \overbrace{1 * \ldots * 0}^{s_0} * \ldots * 0 * \ldots * 1}_{d_1} * \ldots * \underbrace{0}_{d_1} * \ldots *].$$

Further, if we consider all other d_i, it is possible to know the positions of a hopefully large number of zeros in this shift of $\mathbf{h}_0$, which will be our set Z_0. To get a distance s_0 that is in the spectrum $\sigma(\mathbf{h}_0)$ with high probability, one can take the distance with least probability of failure estimated by the spectrum recovery algorithm. An analogous construction can be made for Z_1. In Sect. 5.2, we discuss the relation between the number of known distances inside and outside $\sigma(\mathbf{h}_0)$ and $\sigma(\mathbf{h}_1)$, and the sizes of the sets Z_0 and Z_1, respectively.

The third and last problem is that the positions Z_0 and Z_1 are built based only on *circular* distances. As such, there are no guaranties that Z_0 and Z_1 are positions of zeros in *shifts by the same number of positions* of both $\mathbf{h}_0$ and $\mathbf{h}_1$, respectively. To deal with it, we can fix one of the sets, namely Z_1, and try iteratively, for each $p = 0, 1, \ldots, r-1$, to find a low weight row in the kernel of the matrix $\mathbf{B}_{Z_1'}^{Z_0^{(p)}}$, where $Z_0^{(p)}$ is the set of indexes in Z_0, circularly shifted by p positions.

We now put everything together to give a full description of our algorithm to reconstruct the key as Algorithm 3. Since the algorithm is not recursive and uses common operations, it is straightforward to compute its complexity.

Lemma 9: Let $\mathbf{H} = [\mathbf{H}_0|\mathbf{H}_1]$ be the secret matrix of an $[n, r, w]$-QC-MDPC code. Let Z_0 be a set of indexes of null entries in some line of $\mathbf{H}_0$. Similarly, let Z_1 be a set of indexes of null entries in some line of $\mathbf{H}_1$. Consider Z_1' to be the complement of Z_1 with respect to all possible indexes, that is $Z_1' = [r] - Z_1$. Then the complexity of Algorithm 3 is

$$O\left(r + r|Z_1'| + r\left(r + |Z_0||Z_1'| + |Z_1'|^2|Z_0 + Z_1'| + |Z_1'|\right)\right).$$

Further, since both $|Z_0|$ and $|Z_1'|$ are $O(r)$, the complexity of the attack can be simplified to $O(r^4)$.

Algorithm 3: Proposed key recovery algorithm

Data: r, w parameters of the QC-MDPC code to be broken
D_0 a set of distances not in the spectrum of $\mathbf{h}_0$
D_1 a set of distances not in the spectrum of $\mathbf{h}_1$
s_0 a distance in the spectrum of $\mathbf{h}_0$
s_1 a distance in the spectrum of $\mathbf{h}_1$
$\mathbf{B}$ the right block of the public generator matrix
Result: $\mathbf{h}_1$ which is some line of the matrix $\mathbf{H}_1$, or $\perp$ if $\mathbf{h}_1$ could not be found

1 $Z'_1 \leftarrow \{i \in [r] : \text{dist}(1, i) \notin D_1 \text{ and } \text{dist}(s_1 + 1, i) \notin D_1\}$
2 $\mathbf{B}_{Z'_1} \leftarrow$ rows of $\mathbf{B}$ whose indexes are in Z'_1
3 **for** $p = 0$ *to* $r - 1$ **do**
4 $\quad Z_0 \leftarrow \{i + p \mod r : \text{dist}(1, i) \in D_0 \text{ or } \text{dist}(s_0 + 1, i) \in D_0\}$
5 $\quad \mathbf{B}^{Z_0}_{Z'_1} \leftarrow$ columns of $\mathbf{B}^{Z'_1}$ whose indexes are in Z_0
6 $\quad \mathbf{K} \leftarrow$ left kernel matrix of $\mathbf{B}^{Z_0}_{Z'_1}$
7 $\quad$ **if** $\dim \mathbf{K} = 1$ **then**
8 $\quad\quad \mathbf{v} \leftarrow$ the only row in $\mathbf{K}$
9 $\quad\quad$ **if** $\text{weight}(\mathbf{v}) \le w$ **then**
10 $\quad\quad\quad \mathbf{h}_1 \leftarrow \mathbf{0} \in \mathbb{F}_2^r$
11 $\quad\quad\quad$ **for** *each* $i = 1$ *to* $|\mathbf{v}|$ **do**
12 $\quad\quad\quad\quad \mathbf{h}_1[Z_1[i]] \leftarrow \mathbf{v}[i]$
13 $\quad\quad\quad$ **return** $\mathbf{h}_1$
14 **return** $\perp$

Proof We analyze the cost of each line of Algorithm 3. Lines 1 and 4 are $O(r)$. The cost of building $\mathbf{B}_{Z'_1}$ in line 2 is $O(r|Z'_1|)$, and the cost of building $\mathbf{B}^{Z_0}_{Z'_1}$ in line 5 is $O(|Z_0||Z'_1|)$. The loop in line 3 performs r iterations in the worst case. The kernel computation in line 6 is the most expensive computation, and it costs $|Z'_1|^2|Z'_1 + Z_0|$. The weight computation in line 9 costs $O(r)$. Finally, the loop in line 11 iterates $|\mathbf{v}| = |Z'_1|$ times. Putting these costs together gives the desired result. □

It is important to note that the lemma above does not say anything about the probability of the algorithm finding the key. It only states that the algorithm runs in $O(r^4)$, whether it finds the key or not.

4.2 Case $n_0 \ge 3$

When $n_0 \ge 3$, the public generator matrix $\mathbf{G}$ is given by Equation 1, that is

$$\mathbf{G} = \left[\begin{array}{c|c} & \mathbf{B_0}^T \\ & \mathbf{B_1}^T \\ \mathbf{I} & \vdots \\ & \mathbf{B_{n_0-2}}^T \end{array}\right],$$

where $\mathbf{B}_i = \left(\mathbf{H}^{-1}_{n_0-1} \cdot \mathbf{H}_i\right)$ for each $i = 0, 1, \ldots, n_0 - 2$.

Then we can write, for each i, an equation of the form $\mathbf{H}_{n_0-1}\mathbf{B}_i = \mathbf{H}_i$, where $\mathbf{B}_i$ is known, and we can obtain information on the spectrums of $\mathbf{H}_{n_0-1}$ and $\mathbf{H}_i$ by adapting Algorithm 1 to the corresponding number of blocks n_0. Therefore, we can use our algorithm to recover $\mathbf{H}_{n_0-1}$ and $\mathbf{H}_i$. Once $\mathbf{H}_{n_0-1}$ is recovered, all the other $\mathbf{H}_i$ blocks can be recovered because $\mathbf{H}_i = \mathbf{H}_{n_0-1}\mathbf{B}_i$.

The complexity of the reconstruction algorithm when $n_0 \ge 3$ is the same as when $n_0 = 2$, that is $O(r^4)$. The weight of each block of $\mathbf{H}$ when $n_0 \ge 3$ is slightly higher than for $n_0 = 2$. For example, for 80 bits of security, the weight of each block of $\mathbf{H}$ is around $90/2 = 45$ when $n_0 = 2$, and around $220/4 = 55$ when $n_0 = 4$. When the number of blocks n_0 increases, the size of r gets smaller, for the same security level. This implies that the blocks of $\mathbf{H}$ are less sparse when $n_0 \ge 3$ than when $n_0 = 2$, which can impact negatively on the probability of the key reconstruction algorithm finding the key. This increased density is also bad for Guo's et al. algorithm, which has to explore a larger search tree.

Upon preparation of this paper, it came to our knowledge that the linear relation $\mathbf{h}_1\mathbf{B} = \mathbf{h}_0$ was also explored by Rossi et al. [19]. However our algorithm is significantly different than theirs, mainly because we use information on both $\mathbf{h}_0$ and $\mathbf{h}_1$, while their algorithm uses only information on $\mathbf{h}_0$. This allows our algorithm to perform better.

The next sections are to show how to estimate the probability that the attack succeeds in recovering the key, and its relation to the size of the d-exclusion obtained by the spectrum recovery algorithm.

5. Probabilistic Analysis

5.1 The Probability of the Attack Failing

Let Z_0 and Z_1 be sets of indexes, possibly shifted by the same amount of positions, of some of the null entries of $\mathbf{h}_0$ and $\mathbf{h}_1$. For the algorithm to run successfully, the kernel matrix $\mathbf{K}$ of $\mathbf{B}^{Z_0}_{Z'_1}$ must have exactly one row. If Z_0 and Z_1 are indeed positions of null entries, then $\mathbf{K}$ has at least one row. In other words, we want the linear equation $\mathbf{x}\mathbf{B}^{Z_0}_{Z'_1} = \mathbf{0}$ to have no more than one solution for non-null $\mathbf{x}$.

Since the solutions of this equation impose a linear combination of the rows of $\mathbf{B}^{Z_0}_{Z'_1}$ that must sum to $\mathbf{0}$, it is natural to expect that this equation has fewer solutions when $\mathbf{B}^{Z_0}_{Z'_1}$ has a large number of columns and a small number of rows. That is, we want Z_0 to be large, and Z'_1 to be small, which is equivalent to Z_1 also large. We want to find how large must these sets be for the equation to have exactly one solution.

This is a difficult task because the matrix $\mathbf{B} = \mathbf{H}_1^{-1}\mathbf{H}_0$ is inherently dependent on the sets Z_0 and Z_1, which are also related to $\mathbf{H}_0$ and $\mathbf{H}_1$. In fact, this dependency is what makes the equation always having at least one solution.

To deal with this dependency problem we suggest an approximation that works well on practice consisting in considering $\mathbf{B}^{Z_0}_{Z'_1}$ as a random binary matrix with $|Z_0|$ columns and $|Z'_1|$ rows, such that its kernel contains at least one vector. Under this hypothesis, the sum of any set of rows of $\mathbf{B}^{Z_0}_{Z'_1}$ is a random binary vector of $|Z_0|$ columns. Therefore, the probability that any linear combination of the matrix $\mathbf{B}^{Z_0}_{Z'_1}$ sums to

$\mathbf{0}$ is $2^{-|Z_0|}$. Since there are $2^{Z'_1}$ possible linear combinations of the rows of $\mathbf{B}^{Z_0}_{Z'_1}$, the probability that one of them is $\mathbf{0}$, that is, the probability of the attack failing, is upper bounded by

$$\begin{aligned}\Pr(\text{Algorithm does not recover the key}) &\lessapprox \frac{2^{|Z_1|'}}{2^{|Z_0|}} \\ &\lessapprox \frac{2^{r-|Z_1|}}{2^{|Z_0|}} \\ &\lessapprox 2^{r-|Z_1|-|Z_0|}.\end{aligned}$$

5.2 Finding a d-Exclusion for a High Probability of Success

The approximation above shows that the value of $|Z_0| + |Z_1|$ determines the probability that the attack succeeds, for a fixed r. We now assess how much the sizes of D_0 and D_1, the distances which are known to be outside the spectrums of $\mathbf{h}_0$ and $\mathbf{h}_1$, impact on the sizes of Z_0 and Z_1, respectively.

Fix the probability that the attack fails as $1/2^\gamma$. We want to find the least integer d such that, if both $|D_0|$ and $|D_1|$ are greater than or equal to d, then the attack succeeds with probability $1 - 1/2^\gamma$, for a large fraction α of all possible secret keys. Of course we want α and γ to be large.

The idea is simply to perform a binary search for the least d in the set $[\lfloor r/2 \rfloor] = \{1, \ldots, \lfloor r/2 \rfloor\}$. The problem we face is that the interdependence of the distances outside the spectrum makes it hard to analytically compute the distribution of the size $|Z_0|$ as a function of $|D_0|$, and similarly for $|Z_1|$ and $|D_1|$. Therefore we use simulations, for each d, to estimate the fraction f_d of secret keys for which the attack succeeds with probability at least $1 - 1/2^\gamma$ when $|D_0| = |D_1| = d$. For this, we build N valid pairs of (D_0, Z_0) and (D_1, Z_1), and our estimator for f_d is given by

$$\hat{f}_d = \frac{\#\{\text{Generated } Z_0 \text{ and } Z_1 : |Z_0| + |Z_1| - r \geq \gamma\}}{N}.$$

The search ends with the least value of d such that $\hat{f}_d \geq \alpha$.

Table 3 shows the values of d obtained for different security levels, considering as simulation parameters $N = 5000$, $\alpha = 99\%$, and $\gamma = 20$. For comparison between different security levels, we also put the fractions d/σ', where σ' is the average number of distances outside the spectrum. For each security level, we note that the relative d-exclusion needed for the attack to successfully find the key grows with n_0. The explanation for this fact is that the blocks of the keys get more dense when n_0 is increased, as discussed in the end of Sect. 4.2. For the security level of 256, considering $n_0 = 4$ cyclic blocks, the blocks of the key are so dense that even knowing all distances outside the spectrum is not enough for the algorithm to successfully find the key.

Table 3 Simulation results for the values of d with simulation parameters $N = 5000$, $\alpha = 99\%$, and $\gamma = 20$.

Security level λ	n_0	Estimated size for D_0 and D_1 by the simulation d	Average number of distances outside the spectrum σ'	d/σ'
80	2	$d_{80} = 722$	1582.12	45.63%
80	3	$d_{80} = 545$	873.85	62.36%
80	4	$d_{80} = 471$	576.06	81.76%
128	2	$d_{128} = 1470$	2964.72	49.58%
128	3	$d_{128} = 1116$	1538.19	72.55%
128	4	$d_{128} = 1025$	1175.08	87.28%
256	2	$d_{256} = 4845$	9256.64	52.34%
256	3	$d_{256} = 3347$	3877.48	86.31%
256	4	-	2882.49	-

6. Experimental Results

The analysis of our algorithm presented in the previous section shows that it has better asymptotic complexity than Guo's et al. key reconstruction algorithm. We now show the experimental analysis of the implementation of our algorithm.

We do the experimental analysis in two parts. In the first part, we analyze the performance of our reconstruction algorithm when reconstructing the key for different security parameters. We also compare our algorithm with the one by Guo et al. in the best possible scenario for their algorithm, which is when the spectrum of the key is fully known.

In the second part, we present an empirical analysis of the number of challenges needed for a CCA2 attack for the 80 bits security parameters with $n_0 = 2$. For this analysis, we simulated the first part of the attack, that is, the spectrum recovery procedure, against 200 different private keys.

All the source code and data used in this section is publicly available at www.ime.usp.br/~tpaiva/msc/src.

6.1 Performance of the Proposed Algorithm

The algorithm's performance was analyzed on an Intel i7 870 Lynnfield CPU, at a 2.93GHz clock frequency, and 8GB of RAM. We implemented the algorithm in C using the M4RI [28] library for the kernel matrix computation.

6.1.1 The Algorithm When Full Spectrums are Known

For some security parameters, when both the spectrums of $\mathbf{h}_0$ and $\mathbf{h}_1$ are fully known, the value of $Z_0 + Z_1$ is typically much larger than r. For these parameters, we can consider the set Z_0 as

$$Z_0 \leftarrow \{i + p \mod r : \text{dist}(1, i) \in D_0\},$$

in line 4 of Algorithm 3. In other words, for some parameters, we can consider only one non-null entry when building Z_0, instead of considering two, and $Z_0 + Z_1$ still is sufficiently larger than r.

This simple change allows our algorithm to perform a lot better than Guo's et al. algorithm when full spectrums are known. Table 4 shows the comparison between our algorithm and Guo's et al. algorithm when full spectrums are known. The performance of our algorithm is marked with an asterisk

Table 4 Performance comparison of our algorithm and Guo's et al. algorithm when *full* spectrums are known, for different security parameters.

Security level λ	n_0	Average running time of Guo's et al. algorithm	Average running time of our algorithm
80	2	71.18s	0.72s*
80	3	-	0.21s*
80	4	-	31s
128	2	~ 24 days	11s*
128	3	-	3.75s*
128	4	-	183s
256	2	-	141s*
256	3	-	3h40m
256	4	-	-

Table 5 Performance of our algorithm when *partial* spectrums are known, for different security parameters.

Security level λ	n_0	d-exclusion	Average time per iteration	Average total running time
80	2	$d_{80} = 722$	0.023s	55s
80	3	$d_{80} = 545$	0.014s	25s
80	4	$d_{80} = 471$	0.010s	15s
128	2	$d_{128} = 1470$	0.14s	690s
128	3	$d_{128} = 1116$	0.07s	260s
128	4	$d_{128} = 1025$	0.05s	170s
256	2	$d_{256} = 4845$	4.15s	18h54m
256	3	$d_{256} = 3347$	1.20s	3h46m
256	4	-	-	-

(∗) on the parameters for which the modification above is applicable, that is, the algorithm still finds the key when Z_0 is built with only one non-null entry as reference.

Guo's et al. algorithm performed very poorly for security parameters different from $\lambda = 80$ with $n_0 = 2$. For the security parameters $\lambda = 128$ and $n_0 = 2$, their algorithm took the median time of around 27 hours to explore each branch of the search tree. Parameters for which the algorithm did not run in reasonable time are marked with a hyphen (-) in Table 4. Considering the values of Γ in Table 2, it is unrealistic to expect that Guo's et al. algorithm runs efficiently for the parameters in which $n_0 \geq 3$, or $\lambda = 256$.

6.1.2 The Algorithm with Partial Spectrums

We assess the performance of the algorithm using the minimum amount of information possible, while retaining a good probability of success. For this we use values d given in Table 3 for the sizes of the sets D_0 and D_1.

Note that total execution time, or total number of cycles, alone are not very informative metrics for this algorithm. This is because the number of iterations for reconstructing a key is a random variable following a uniform distribution in the set $[r]$. Therefore, if we know the execution time per iteration, we can model more accurately the performance of the algorithm. Table 5 shows the performance results of our algorithm for different security parameters. We do not consider the performance of Guo's et al. algorithm in this table because their algorithm takes too long to finish when partial spectrums are considered, making its application infeasible.

6.2 The Number of Challenges Needed for a CCA2 Attack

We now analyze the number of decoding trials for a CCA2 attack when we use the proposed algorithm for key reconstruction. For comparison purposes, we consider the security level of 80 bits and $n_0 = 2$, which are the same parameters considered by Guo et al.

For the analysis, 200 different codes were randomly generated and for each of them, were performed 100 million decoding trials. The decoder used was the same used by Guo et al., that is, algorithm $\mathcal{B}$ considered by Maurich et al. [25]. Each battery of decoding trials looks like the one in Fig. 3. The y-axis shows the value $d = \min(d_0, d_1)$, where d_0 and d_1 are the exclusions obtained for $\sigma(\mathbf{h}_0)$ and $\sigma(\mathbf{h}_1)$, respectively.

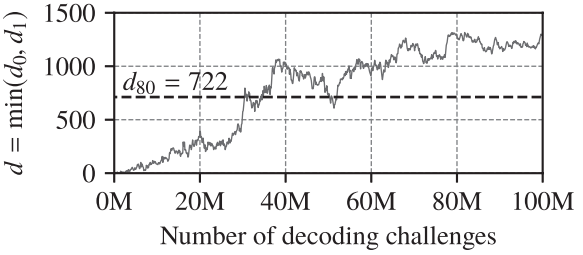

Fig. 3 Example of the result of the decoding challenges for one of the 200 codes.

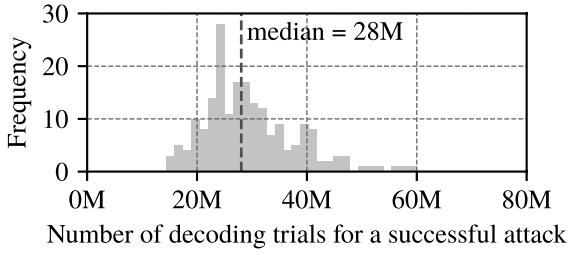

Fig. 4 Histogram of the number of decoding trials for a successful key reconstruction.

Since we can successfully reconstruct the key when the exclusion for both $\sigma(\mathbf{h}_0)$ and $\sigma(\mathbf{h}_1)$ are greater than or equal to $d_{80} = 722$, as shown in Table 3, we are interested in knowing how many decoding trials are needed to obtain a d-exclusion of 722 for both of the spectrums. In the example shown in Fig. 3, the first point where the attack is possible is at around 30 million decoding trials.

The histogram of the number of decoding trials necessary for a successful attack is shown in Fig. 4. Guo's et al. algorithm need an almost complete d-exclusion to run efficiently, which occurs after around 200 million[†] decoding

[†]In their paper, Guo et al. report that in the best case of their simulation the spectrum was fully recovered after 203 million decoding challenges.

challenges [18]. Our results suggest that when our algorithm is used to reconstruct the key only around 28 million decoding trials are needed, which is a significant improvement on the amount of interaction with the secret key holder.

7. Conclusion

We propose a new algorithm for key reconstruction that is faster than Guo's et al. algorithm, and uses significantly less interaction with the secret key holder than needed by the latter. Our algorithm has lower asymptotic complexity, which suggest it scales much better for higher security levels. Our experimental analysis of the algorithm shows that it performs much better than Guo's et al. algorithm for all security parameters.

The proposed algorithm is different in nature than the one by Guo et al.: it is based on linear algebra instead of a pruned depth-first search in a tree. It can also be straightforwardly parallelized, which is not the case for the Guo's et al. approach. Our algorithm has a small probability of failing in recovering the key, but we show how to select the parameters in such a way that the probability of failure becomes negligible.

For future works, it would be interesting to study how the use of other decoding algorithms impacts on the number of decoding trials needed for the attack. Another possible research question is to consider variants of the proposed algorithm that are successful with even fewer knowledge about both spectrums. This is important because it is related to the determination of the lifetime of a QC-MDPC secret key.

Acknowledgments

We thank the anonymous referees for important recommendations and helpful comments. The simulations were performed using the High Performance Computing resources provided by the Technology Superintendence of Universidade de São Paulo.

References

[1] P.W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," Proc. 35th Annual Symposium on Foundations of Computer Science, pp.124–134, IEEE, 1994.
[2] D.J. Bernstein, J. Buchmann, and E. Dahmen, Post-Quantum Cryptography, Springer Science & Business Media, 2009.
[3] L. Chen, L. Chen, S. Jordan, Y.K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, "Report on post-quantum cryptography," US Department of Commerce, National Institute of Standards and Technology, 2016.
[4] R.J. McEliece, "A public-key cryptosystem based on algebraic coding theory," Deep Space Network Progress Report, vol.44, pp.114–116, 1978.
[5] E.R. Berlekamp, R.J. McEliece, and H.C. Van Tilborg, "On the inherent intractability of certain coding problems," IEEE Trans. Inf. Theory, vol.24, no.3, pp.384–386, 1978.
[6] R.L. Rivest, A. Shamir, and L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM, vol.21, no.2, pp.120–126, 1978.
[7] V. Miller, "Use of elliptic curves in cryptography," Advances in Cryptology (CRYPTO85), pp.417–426, 1986.
[8] V.D. Goppa, "A new class of linear correcting codes," Problemy Peredachi Informatsii, vol.6, no.3, pp.24–30, 1970.
[9] P. Gaborit, "Shorter keys for code based cryptography," Proc. 2005 International Workshop on Coding and Cryptography (WCC 2005), pp.81–91, 2005.
[10] T.P. Berger, P.L. Cayrel, P. Gaborit, and A. Otmani, "Reducing key length of the McEliece cryptosystem," Progress in Cryptology–AFRICACRYPT 2009, pp.77–97, Springer, 2009.
[11] M. Baldi, F. Chiaraluce, R. Garello, and F. Mininni, "Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem," Communications, 2007. ICC'07. IEEE International Conference on, pp.951–956, IEEE, 2007.
[12] R. Misoczki and P.S. Barreto, "Compact McEliece keys from goppa codes," Selected Areas in Cryptography, pp.376–392, Springer, 2009.
[13] A. Otmani, J.P. Tillich, and L. Dallot, "Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes," Math. Comput. Sci., vol.3, no.2, pp.129–140, 2010.
[14] J.C. Faugere, A. Otmani, L. Perret, and J.P. Tillich, "Algebraic cryptanalysis of McEliece variants with compact keys," Advances in Cryptology–Eurocrypt 2010, pp.279–298, Springer, 2010.
[15] J.C. Faugere, A. Otmani, L. Perret, F. De Portzamparc, and J.P. Tillich, "Structural cryptanalysis of McEliece schemes with compact keys," Des. Codes Cryptogr., vol.79, no.1, pp.87–112, 2016.
[16] R. Misoczki, J.P. Tillich, N. Sendrier, and P.S. Barreto, "MDPC-McEliece: New McEliece variants from moderate density parity-check codes," Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on, pp.2069–2073, IEEE, 2013.
[17] D. Augot, L. Batina, D.J. Bernstein, J. Bos, J. Buchmann, W. Castryck, O. Dunkelmann, T. Güneysu, S. Gueron, and A. Hülsing, "Initial recommendations of long-term secure post-quantum systems (2015)," URL: https://pqcrypto.eu.org/docs/initial-recommendations.pdf. Citations in this document, vol.16.
[18] Q. Guo, T. Johansson, and P. Stankovski, "A key recovery attack on MDPC with CCA security using decoding errors," 22nd Annual International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT), pp.789–815, 2016.
[19] M. Rossi, M. Hamburg, M. Hutter, and M.E. Marson, "A side-channel assisted cryptanalytic attack against QcBits," 2017. http://eprint.iacr.org/2017/596
[20] T. Chou, "QcBits: Constant-time small-key code-based cryptography," International Conference on Cryptographic Hardware and Embedded Systems, pp.280–300, Springer, 2016.
[21] R. Gallager, "Low-density parity-check codes," IRE Trans. Inf. Theory, vol.8, no.1, pp.21–28, 1962.
[22] A. Shokrollahi, C. Monico, and J. Rosenthal, "Using low density parity check codes in the McEliece cryptosystem," IEEE International Symposium on Information Theory (ISIT 2000), p.215, 2000.
[23] D.J. Bernstein, T. Chou, and P. Schwabe, "McBits: Fast constant-time code-based cryptography," International Workshop on Cryptographic Hardware and Embedded Systems, pp.250–272, Springer, 2013.
[24] K. Kobara and H. Imai, "Semantically secure McEliece public-key cryptosystems-conversions for McEliece PKC," International Workshop on Public Key Cryptography, pp.19–35, Springer, 2001.
[25] I.V. Maurich, T. Oder, and T. Güneysu, "Implementing QC-MDPC McEliece encryption," ACM Trans. Embed. Comput. Syst. (TECS), vol.14, no.3, p.44, 2015.
[26] Q. Guo, "Guo's presentation at Asiacrypt," https://youtu.be/tKvDdGLJLZc?t=1006, 2016.
[27] T. Fabšič, V. Hromada, P. Stankovski, P. Zajac, Q. Guo, and T. Johansson, "A reaction attack on the QC-LDPC McEliece cryptosystem," International Workshop on Post-Quantum Cryptography, pp.51–68, Springer, 2017.
[28] M. Albrecht and G. Bard, The M4RI Library – Version 20121224, The M4RI Team, 2012.

Thales Bandiera Paiva MSc student at the Department of Computer Science, Universidade de São Paulo, Brazil, received his bachelor's degree in Computer Science from Universidade de São Paulo in 2015.

Routo Terada Professor at the Department of Computer Science, Universidade de São Paulo, Brazil, MSc in Applied Math from Universidade de São Paulo and PhD in Computer Science from University of Wisconsin-Madison, USA.