Key reconstruction for QC-MDPC McEliece from imperfect distance spectrum

Motonari Ohtsuka[1][0000-0002-7618-8367], Takahiro Ishimaru[2][0009-0002-2413-6626], Rei Iseki[2][0009-0002-7323-3190], Shingo Kukita[2][0000-0003-1526-4570], and Kohtaro Watanabe[2][0000-0001-9674-0924]

[1] Test and Evaluation Unit, Japan Ground Self-Defense Force, Shizuoka, Japan
[2] Department of Computer Science, National Defense Academy of Japan, Kanagawa, Japan
{kukita,wata}@nda.ac.jp

Abstract. McEliece cryptosystems, based on code-based cryptography, is a candidate in Round 4 of NIST's post-quantum cryptography standardization process. The QC-MDPC (quasi-cyclic moderate-density parity-check) variant is particularly noteworthy due to its small key length. The Guo-Johansson-Stankovski (GJS) attack against the QC-MDPC McEliece cryptosystem was recently proposed and has intensively been studied. This attack reconstructs the secret key using information on decoding error rate (DER). However, in practice, obtaining complete DER information is presumed to be time-consuming. This paper proposes two algorithms to reconstruct the secret key under imperfection in the DER information and evaluates the relationship between the imperfection and efficiency of key reconstruction. This will help us to increase the efficacy of the GJS attack.

Keywords: McEliece cryptosystem · QC-MDPC codes · Reaction attacks · Distance spectrum · Key reconstruction.

1 Introduction

Cryptography is an indispensable concept in modern society, which cannot exist without networks. Cryptosystems based on difficulty of factoring, or similarly, difficulty of discrete logarithm problems are currently applied in many situations. However, they will be jeopardized if large quantum computers are realized because Shor's algorithm, which can be executed by quantum computers, solves factoring and discrete logarithm problems in polynomial time [24]. Thus, providing new cryptosystems that are secure even after the realization of large quantum computers (post-quantum era) are getting increasing attention. Consequently, National Institute of Standards and Technology (NIST) is calling for proposals of post-quantum cryptosystems [1]. Some lattice-based cryptosystems were recently released as Federal Information Processing Standards (FIPS), but alternatives are still being sought extensively.

The McEliece cryptosystem is a public key cryptosystem based on the difficulty of decoding random linear codes [17], and has extensively been studied [5, 8, 22]. Its original version with Goppa codes is considered as one of the Round-4 candidates in the post-quantum cryptography standardization initiated by NIST. However, its public key size is excessively large because one uses the

whole generator matrix of a linear code in this scheme. According to the demand of smaller key sizes, a variant based on Quasi-Cyclic Moderate-Density Parity-Check (QC-MDPC) codes was proposed [18]. Utilizing quasi-cyclicity of the generator matrix in this scheme, one represents it by its first row; hence, the key size can be reduced compared with the original McEliece scheme. The QC-MDPC scheme is still being considered as an alternative to lattice-based cryptography.

An attack on the QC-MDPC scheme was recently proposed [14]. This attack, which is called the Guo-Johansson-Stankovski (GJS) attack, tries to reconstruct the secret key from statistics of decoding errors. This is composed of two steps. In the first step, an adversary repeatedly sends specific messages to a legitimate receiver and observes receiver's reactions, i.e., whether they succeed or fail to decode the sent messages. Analyzing the decoding error rate (DER) through this process allows the adversary to estimate *distance spectrum*, the set of distances of all pairs of ones in the secret key, which is represented by a binary vector. The performance of the estimation can change depending on what decoding algorithm the legitimate receiver employs, and in particular, the bit-flipping (BF) algorithm [2, 7, 10, 15, 18, 19, 23] was shown to be vulnerable to this attack [4].

In the second part, the attacker tries to reconstruct the secret key from the estimated distance spectrum. This process is performed by depth-first recursive search, in the original proposal [14], and this is generally time-consuming. Recently, breadth-first search algorithm for reconstructing the secret key is proposed [20]. This breadth-first algorithm was shown to have smaller time complexity than the depth-first search at the cost of its space complexity.

The first part of estimating the distance spectrum is considered more time-consuming than the second part of reconstructing the secret key regardless of what reconstruction algorithm is employed. This is because we typically assume *perfect* estimation of the distance spectrum in the first part and the key reconstruction from the obtained *complete* distance spectrum in the second part. Obtaining the complete distance spectrum in the first part requires a large number of trials of sending messages to the receiver. If we reduce the number of trials, we will obtain an *imperfect* distance spectrum, and it will make the key reconstruction in the second part more time-consuming. In such cases, it is non-trivial which part is dominant in terms of time complexity.

In this paper, we consider key reconstruction from an imperfect distance spectrum. Assume that the number of trials of sending messages in the first part is insufficient to perfectly determine the distance spectrum. Even in this case, however, we will be able to obtain "upper and lower bounds" of the true spectrum. Extending the two above algorithms of depth-first and breadth-first search, we propose two reconstruction algorithms using the lower and upper bound instead of the distance spectrum itself. Both methods successfully reconstruct the secret key under some level of imperfection in the distance spectrum. We compare the time complexity of them and find that the breadth-first algorithm is more efficient than the depth-first one, which is similar to the case of reconstruction from the complete distance spectrum.

The remaining of this paper is organized as follows. In Sec. 2, we briefly review the QC-MDPC McEliece cryptosystem. The key recovery attack based on information of the DER is introduced in Sec. 3. In Sec. 4, we discuss how the imperfection of the distance spectrum is defined and propose two reconstruction

algorithm from an imperfect distance spectrum. Also in this section, we exhibits results of numerical experiments. Section 5 is devoted to summary of this work.

2 QC-MDPC McEliece cryptosystems

2.1 McEliece cryptosystems

The McEliece cryptosystem is characterized by a parity-check matrix $\boldsymbol{H} \in \mathbb{F}_2^{r\times n}$ and the corresponding generator matrix $\boldsymbol{G} \in \mathbb{F}_2^{k\times n}$ of an error-correcting code that can correct t-bit error. Here, n is the code length, k the information bit length, and r the codimension: $r = n - k$. $\mathbb{F}_2^{i\times j}$ represents an $i \times j$ matrix with elements in $\mathbb{F}_2 = \{0, 1\}$.

We randomly generate a permutation matrix $\boldsymbol{Q} \in \mathbb{F}_2^{n\times n}$ and a regular matrix $\boldsymbol{S} \in \mathbb{F}_2^{k\times k}$, and provide

$$\boldsymbol{G'} = \boldsymbol{SGQ}. \tag{1}$$

A public key of the McEliece cryptosystem is $(\boldsymbol{G'}, t)$ while the secret key is $(\boldsymbol{H}, \boldsymbol{S}, \boldsymbol{Q})$. It is worth mentioning that any person can formally make a parity-check matrix $\boldsymbol{H'}$ using $\boldsymbol{G'}$, but this is different from $\boldsymbol{H}$ in the secret key. A sender tries to convey a message $\boldsymbol{m} \in \mathbb{F}_2^k$. They encode the message as

$$\boldsymbol{c} = \boldsymbol{mG'} + \boldsymbol{e}, \tag{2}$$

with a randomly generated error $\boldsymbol{e} \in \mathbb{F}_2^n$, whose Hamming weight is smaller than t: $w(\boldsymbol{e}) \leq t$.

The encoded message $\boldsymbol{c}$ is sent to the receiver, and is decrypted through the following procedure.

(i) Operate $\boldsymbol{Q}^{-1}$ on $\boldsymbol{c}$:

$$\boldsymbol{cQ}^{-1} = \boldsymbol{mG'Q}^{-1} + \boldsymbol{eQ}^{-1} = \boldsymbol{mSG} + \boldsymbol{e'},$$

where $\boldsymbol{e'}$ denotes $\boldsymbol{eQ}^{-1}$.

(ii) $\boldsymbol{e'}$ can be corrected in this code because $w(\boldsymbol{e'}) = w(\boldsymbol{e}) \leq t$. Then, the receiver obtains,

$$\boldsymbol{c'} = \boldsymbol{mSG}.$$

(iii) $\boldsymbol{G^{-1}}, \boldsymbol{S^{-1}}$ are applied to $\boldsymbol{c'}$:

$$\begin{aligned}\boldsymbol{c'G^{-1}S^{-1}} &= \boldsymbol{mSGG^{-1}S^{-1}} \\ &= \boldsymbol{mSS^{-1}} \\ &= \boldsymbol{m}.\end{aligned}$$

Therefore, the receiver recovers the original message $\boldsymbol{m}$, which the sender tried to convey. The performance of this process depends on how efficiently the legitimate receiver corrects the error while the correction by attackers is kept to be difficult.

2.2 QC-MDPC McEliece cryptosystem

As the public key of the McEliece cryptosystem is $\boldsymbol{G'}$, which is typically a large matrix, the key size tends to be large. To resolve this issue, the QC-MDPC McEliece cryptosystem (QC-MDPC scheme, hereinafter) was proposed [18]. In this scheme, $\boldsymbol{Q}$ and $\boldsymbol{S}$ are set to be the identity matrices with dimension n and k, respectively. In Ref. [18], it is pointed out that assigning security functions to these matrices is based on a "folklore" reasoning: the most important condition to ensure security is that the public key does not divulge any useful information to attackers. As shown below, the QC-MDPC scheme actually satisfies this condition although it does not employ $\boldsymbol{Q}$ and $\boldsymbol{S}$.

The secret key in the scheme is a parity-check matrix in the following form:

$$\boldsymbol{H} = \left[\boldsymbol{H_0}\ \boldsymbol{H_1}\ \cdots \boldsymbol{H_{n_0-1}}\right] \in \mathbb{F}_2^{r\times n},$$

where each $\boldsymbol{H_i}$ $(0 \le i \le n_0 - 1)$ is a circulant matrix of size $r \times r$ and thus $n = n_0 r$. Let the first row of $\boldsymbol{H_i}$ be $\boldsymbol{h_i}$. We assume that the weight of each $\boldsymbol{h_i}$ is d_v. Correspondingly, the row weight of the whole parity-check matrix $\boldsymbol{H}$ is $w = d_v n_0$. In the QC-MDPC scheme, $\boldsymbol{H}$ is taken to be sparse and its weight w scales in $O(\sqrt{n \log n})$ [18].

The generator matrix corresponding to the parity-check matrix $\boldsymbol{H}$ is given by

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

It is sufficient to publish only the first rows of the blocks because each block $\left(\boldsymbol{H_{n_0-1}^{-1}} \cdot \boldsymbol{H_i}\right)^T$ $(0 \le i \le n_0 - 2)$ is a circulant matrix. Hence, the key size is much smaller than the original version of McEliece scheme, which needs to publish the whole generator matrix.

Encoding and decoding in this scheme is performed in a similar way to the original version. We again explain its encoding and decoding procedure and how this scheme works. The public key of this scheme is $(\boldsymbol{G}, t)$ while the secret key is $\boldsymbol{H}$. The sender encode their message $\boldsymbol{m}$ as

$$\boldsymbol{c} = \boldsymbol{mG} + \boldsymbol{e}, \tag{3}$$

with a randomly generated error $\boldsymbol{e}$ whose wight $w(\boldsymbol{e})$ is less than t. After receiving this encoded message $\boldsymbol{c}$, the receiver decodes it through the following procedure.

(i) Operate $\boldsymbol{H}^T$ on $\boldsymbol{c}$ to obtain the syndrome $\boldsymbol{s}$:

$$\boldsymbol{s} = \boldsymbol{cH}^T = \boldsymbol{mGH}^T + \boldsymbol{eH}^T = \boldsymbol{eH}^T.$$

(ii) Calculate $\boldsymbol{e}$ from the information of $\boldsymbol{s}$ and $\boldsymbol{H}$. This problem can efficiently be solved because of the sparsity of $\boldsymbol{H}$. Subtracting the calculated $\boldsymbol{e}$ from $\boldsymbol{c}$, the receiver acquire $\boldsymbol{c'} = \boldsymbol{mG}$.

(iii) The receiver operates $\boldsymbol{G^{-1}}$ to $\boldsymbol{c'}$ and succeed to reproduce $\boldsymbol{m}$.

One may consider that an eavesdropper can intercept the sent message $\boldsymbol{m}$ because they have the generator matrix itself, which is not camouflaged with $\boldsymbol{Q}$ and $\boldsymbol{S}$. The eavesdropper constructs the parity-check matrix $\widetilde{\boldsymbol{H}}$ from the public key $\boldsymbol{G}$ as

$$\widetilde{\boldsymbol{H}} = \left[(\boldsymbol{H}_{\boldsymbol{n_0-1}}^{-1} \cdot \boldsymbol{H_0}) \cdots (\boldsymbol{H}_{\boldsymbol{n_0-1}}^{-1} \cdot \boldsymbol{H}_{\boldsymbol{n_0-2}}) \mid \boldsymbol{I_k} \right], \tag{4}$$

and can perform the step (i) just like the legitimate receiver. Then, they acquire $\tilde{\boldsymbol{s}} = \boldsymbol{e}\widetilde{\boldsymbol{H}}$. The step (ii) is, however, almost impossible for them: finding $\boldsymbol{e}$ from $\tilde{\boldsymbol{s}}$ and $\widetilde{\boldsymbol{H}}$ is a difficult task because $\widetilde{\boldsymbol{H}}$ is not sparse in general. This problem is called syndrome decoding problem, and known to be NP-hard (NP-equivalent), which provides the security of the QC-MDPC scheme [6, 16].

The parameters (n, r, d_v) determine the security of the QC-MDPC scheme. Table 1 exhibits the parameters corresponding to several security levels proposed in Ref. [18]. Also, Table 2 shows parameters for Bit Flipping Key Encapsulation (BIKE) scheme [3]. Throughout the remaining of this paper, we consider the cases of $(n, r, d_v, n_0) = (9602, 4801, 45, 2)$ and $(24646, 12323, 71, 2)$. In these cases, the parity-check matrix $\boldsymbol{H}$ is expressed as

$$\boldsymbol{H} = [\boldsymbol{H_0}\ \boldsymbol{H_1}],$$

and the generator matrix $\boldsymbol{G}$ is

$$\boldsymbol{G} = \left[\boldsymbol{I_k} \middle| \left(\boldsymbol{H}_1^{-1} \cdot \boldsymbol{H_0} \right)^T \right].$$

Table 1. The parameters for the QC-MDPC scheme at 80, 128, and 256-bit security levels.

security	n_0	n	r	d_v	key size
80	2	9602	4801	45	4801
80	3	10779	3593	51	7186
80	4	12316	3079	55	9237
128	2	19714	9857	71	9857
128	3	22299	7433	81	14866
128	4	27212	6803	85	20409
256	2	65542	32771	137	32771
256	3	67593	22531	155	45062
256	4	81932	20483	161	61449

Table 2. The parameters for BIKE at 128, 192, 256 security levels.

Security	n_0	n	r	d_v	key size
128	2	24646	12323	71	12323
192	2	49318	24659	103	24659
256	2	81946	40973	137	40973

3 Review of the GJS attack

The goal of key recovery attacks is to obtain the secret key, i.e., $\boldsymbol{H}$ in the case of the McEliece cryptosystems. In the QC-MDPC scheme, the secret key $\boldsymbol{H}$ can easily be derived if we know $\boldsymbol{H_0}$ and the public key $\boldsymbol{G}$. As $\boldsymbol{H_0}$ is a circulant matrix, obtaining its first row $\boldsymbol{h_0}$ is sufficient to know $\boldsymbol{H_0}$. Summarizing, if we obtain $\boldsymbol{h_0}$, we can easily reconstruct the secret key $\boldsymbol{H}$. Therefore, we hereinafter identify $\boldsymbol{h_0}$ with the secret key itself. In Ref. [13], a key recovery attack against the QC-MDPC scheme, which aims to reconstruct $\boldsymbol{h_0}$ is proposed. This attack is comprised of two steps. First, an attacker repeatedly sends a specific message to a legitimate receiver, and collects the information of how many times the receiver fails to decode the message, i.e., the information of decoding error rate (DER). From this information, the attacker estimates the "distance spectrum" of the secret key $\boldsymbol{h_0}$. The distance spectrum is the set of distances of all pairs of ones in the key. Using the estimated distance spectrum, the attacker tries to construct a secret key candidate $\boldsymbol{h_0'}$ in an appropriate way. The attack is successful if the constructed candidate $\boldsymbol{h_0'}$ matches the secret key $\boldsymbol{h_0}$. In the original proposal in Ref. [13], recursive search is employed to construct the candidate. This search algorithm, however, tends to be time-consuming because of its recursive process. Recently, a breadth-first algorithm was proposed in Ref. [20], which can reduce the time complexity of reconstructing the key at the cost of its space complexity. In this section, we explain how one can acquire the distance spectrum of the secret key from the DER information, and review the above two methods for reconstructing the secret key.

3.1 Distance spectrum

Consider a binary vector $\boldsymbol{C} = [c_0, \cdots, c_{r-1}] \in \mathbb{F}_2^r$. A (cyclic) distance between the i-th component c_i and the j-th component c_j is defined as follows:

$$d(i,j) := \min\{|i-j|, r-|i-j|\}. \tag{5}$$

It is worth mentioning that $\max_{i,j} d(i,j) = U := \lfloor r/2 \rfloor$. We define the distance multiplicity $\mu_C(d)$ of distance d in the vector $\boldsymbol{C}$ and the distance spectrum $D(\boldsymbol{C})$ as

$$\mu_C(d) = \left|\left\{(i,j) \mid (c_i = c_j = 1) \wedge d(i,j) = d\right\}\right|, \tag{6}$$

$$D(\boldsymbol{C}) = \{d \in [U] \mid \mu_C(d) > 0\}, \tag{7}$$

where $[U] = \{1, 2, \cdots, U\}$. We also introduce the set of positions of ones in $\boldsymbol{C}$ as

$$\begin{aligned}\boldsymbol{P}(\boldsymbol{C}) = &\{p_0, p_1, \ldots, p_{w(\boldsymbol{C})-1}\}, \text{ such that} \\ &c_{p_i} = 1, \quad 0 \le i \le w(\boldsymbol{C}) - 1.\end{aligned}$$

In this paper, we often identify $\boldsymbol{P}(\boldsymbol{C})$ with $\boldsymbol{C}$ itself unless it causes any confusion. For instance, we will use $D(\boldsymbol{P}(\boldsymbol{C}))$ instead of $D(\boldsymbol{C})$ when this makes notation simpler.

Let us explain how an attacker obtains the distance spectrum of the secret key $\boldsymbol{h_0}$. To this end, we define the set of error patterns Ψ_d as follows:

$$\begin{aligned}\Psi_d = \big\{&\boldsymbol{v} = (\boldsymbol{e}, \boldsymbol{f}) \mid w(\boldsymbol{f}) = 0, \\ &\boldsymbol{P}(\boldsymbol{e}) = \{s_0, s_1, \ldots, s_t\},\ s_{2i} = (s_{2i-1} + d) \mod r,\ i = 1, \ldots, t/2\big\}.\end{aligned} \tag{8}$$

This set is composed of errors $\boldsymbol{v}$ of length $2r$ where the first half $\boldsymbol{e}$ with length r contains $t/2$ pairs of ones at distance d and the second half $\boldsymbol{f}$ is a zero vector. The attacker randomly selects one error $\boldsymbol{v}$ from Ψ_d and send a message with the selected error. They observe reactions of the receiver, and determine whether the sent message is successfully decoded on not. The attacker repeats this process M times for each d, and count the number of decoding failures, denoted by N. An important observation pointed out in Ref. [14] is that the observed failure ratio $\mathrm{DER}_d := N/M$ relates with the distance multiplicity $\mu_{\boldsymbol{h}_0}(d)$: the smaller multiplicity a distance d has, the lower the DER_d tends to be. In particular, distances with $\mu_{\boldsymbol{h}_0}(d) = 0$ will be separated from the other distances with non-zero multiplicities, and thus we can estimate the distance spectrum $D(\boldsymbol{h}_0)$. Ideally, if we obtain the "true" DERs by taking the limit of $M \to \infty$, we should perfectly determine $\mu_{\boldsymbol{h}_0}(d)$. In practice, we can only perform a large but finite number of trials and obtain an empirical DERs, which makes the determination imperfect.

BF decoding and its variants have often been used as decoding methods for the QC-MDPC scheme [7,10,15,18,19,23]. These methods are fast but vulnerable to the above reaction attack. We employ the BF decoding to clearly exhibit the effectiveness of this attack and show empirical DERs for all distances in Fig. 1. We take the parameters $(n, r, d_v, t) = (9602, 4801, 45, 110)$ and the number of trials $M = 10000$ in this experiment. From this figure, one can observe that the empirical DERs for $d \in D(\boldsymbol{h}_0)$ are lower than that for $d \notin D(\boldsymbol{h}_0)$. Using this property, one can make a threshold that separates DERs for $d \in D(\boldsymbol{h}_0)$ and for $d \notin D(\boldsymbol{h}_0)$, and estimates the distance multiplicities for all distances. In particular, the distance spectrum of the secret key $D(\boldsymbol{h}_0)$ is estimated.

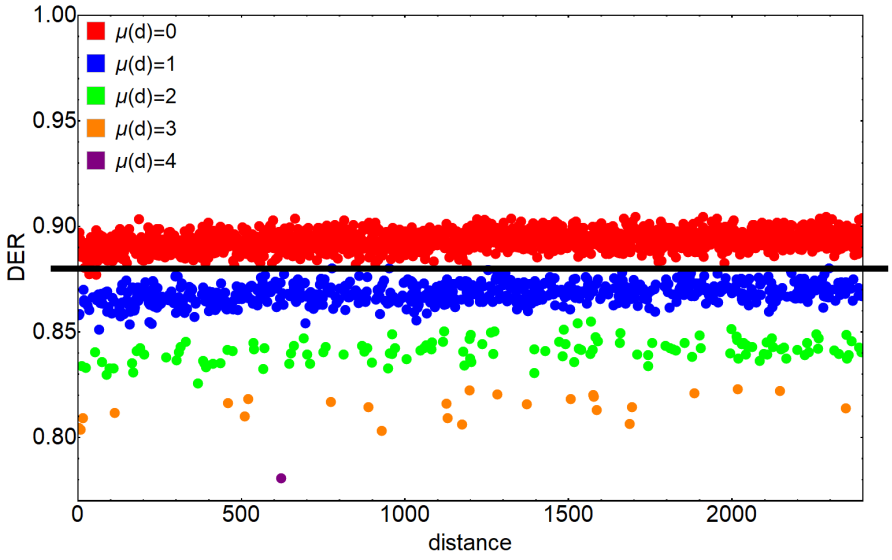

Fig. 1. Empirical DERs for all distances. The parameters are taken to be $(n, r, d_v, t) = (9602, 4801, 45, 110)$. The number of trials is $M = 10000$.

3.2 Construction of secret key candidates

We assume that the attacker succeeds to acquire the distance spectrum perfectly. Let $\boldsymbol{h}_{\boldsymbol{0}}^{'}$ denote a secret key candidate to be obtained utilizing this spectrum. The key candidate constructed from the distance spectrum can be different from the secret key itself because the distance spectrum is invariant under cyclic shifts and the mirror inversion. Also, as mentioned later, there can be several non-trivially different candidates with the same distance spectrum. This is why we represent the constructed candidates by $\boldsymbol{h}_{\boldsymbol{0}}^{'}$, not by $\boldsymbol{h}_{\boldsymbol{0}}$.

Our goal is to obtain $\boldsymbol{h}_{\boldsymbol{0}}^{'}$, or equivalently, the corresponding set of positions of ones in the candidate:

$$\boldsymbol{P}(\boldsymbol{h}_{\boldsymbol{0}}^{'}) = \{p_0, p_1, \ldots, p_{d_v-1}\}, \tag{9}$$

where d_v is the weight of the secret key $\boldsymbol{h}_{\boldsymbol{0}}$. The original proposal in Ref. [14] employed recursive search to construct $\boldsymbol{P}(\boldsymbol{h}_{\boldsymbol{0}}^{'})$. We will briefly review this search algorithm.

Let us align the distances in the spectrum in increasing order as

$$D(\boldsymbol{h}_{\boldsymbol{0}}) = \{i_0, i_1, \cdots\}, \quad 0 < i_0 < i_1 < \cdots .$$

We can set $p_0 = 0$ and $p_1 = i_0$ without loss of generality because as aforementioned, the distance spectrum is invariant under cyclic shifts and the mirror inversion. Then, we tentatively set $p_2 = p_1 + i_0$ and check whether the distance between p_1 and p_2 is in $D(\boldsymbol{h}_{\boldsymbol{0}})$. If it is, we search candidates of p_3 while fixing $p_2 = p_1 + i_0$; otherwise, consider the possibility of $p_2 = p_1 + i_1$ in the same way. We repeat this process recursively until p_{d_v-1} is fixed. Note that there is no proof that the distance spectrum uniquely determines the secret key: a binary vector, which is neither a cyclic shift nor the mirror image of the secret key, may have the same spectrum. Hence, we need to exhaust all candidates that have the obtained spectrum. The algorithm is summarized in Algorithms 1 and 2.

Algorithm 1 Recursive search for candidates

Input: $D(\boldsymbol{h}_{\boldsymbol{0}})$, $p_0 = 0$, $p_1 = \min(D(\boldsymbol{h}_{\boldsymbol{0}}))$, $l = 2$.
Output: set of secret key candidates $\mathcal{H}_0^{'}$

1: $\mathcal{H}_0^{'} \leftarrow \{\}$.
2: $\boldsymbol{h}_{\boldsymbol{0}}^{'} \leftarrow \{p_0, p_1\}$.
3: Call $\mathcal{R}(\boldsymbol{h}_{\boldsymbol{0}}^{'}, l = 2)$ in Algorithm 2.
4: **return** $\mathcal{H}_0^{'}$.

Algorithm 2 Recursive function $\mathcal{R}(\boldsymbol{h}_{\boldsymbol{0}}^{'}, l)$

Input: $\boldsymbol{h}_{\boldsymbol{0}}^{'} = \{p_0, p_1, \cdots, p_{l-1}\}$, l.
1: **for all** i $(p_{l-1} + 1 \le i \le r)$ **do**
2: **for all** j $(0 \le j \le l-1)$ **do**
3: **if** $d(i, p_j) \notin D(\boldsymbol{h}_{\boldsymbol{0}})$ **then**
4: goto line 25.
5: **end if**
6: **end for**
7: $p_l \leftarrow i$.

8: append p_l to $\boldsymbol{h}_{\boldsymbol{0}}^{'}$.
9: **if** $l \leq \frac{d_v+1}{2}$ **and** $p_l \geq \left\lceil \frac{r+p_1}{2} \right\rceil$ **then**
10: goto line 25.
11: **end if**
12: **if** $l = d_v - 1$ **then**
13: Calculate $D(\boldsymbol{h}_{\boldsymbol{0}}^{'})$.
14: **if** $D(\boldsymbol{h}_{\boldsymbol{0}}^{'}) = D(\boldsymbol{h_0})$ **then**
15: append $\boldsymbol{h}_{\boldsymbol{0}}^{'}$ to $\boldsymbol{\mathcal{H}}_0^{'}$.
16: goto line 25.
17: **else**
18: goto line 25.
19: **end if**
20: **end if**
21: $l \leftarrow l + 1$.
22: Recursive call of $\mathscr{R}(\boldsymbol{h}_{\boldsymbol{0}}^{'}, l)$.
23: Remove p_l from $\boldsymbol{h}_{\boldsymbol{0}}^{'}$.
24: $l \leftarrow l - 1$.
25: **end for**
26: **return**

In the lines 9 – 11 of Algorithm 2, we prune some candidates that match the conditions. Naively speaking, this process prunes candidates in which more than half of ones are concentrated in its latter part (latter-concentrated) while reserves candidates in which more than half of ones are concentrated in its former part (former-concentrated). Note that the mirror image of a latter-concentrated candidate is a former-concentrated one. If we do not perform this pruning process, we will construct both of them. Although both can be candidates of the true secret key, it is sufficient to construct one of them in this algorithm; after that, the other is efficiently reproduced just by taking the mirror image of the constructed one. Hence, we can prune latter-concentrated candidates without loss of generality. Moreover, performing this process reduces the search space and improve the time performance of the algorithm. This idea will be applicable to any other construction algorithms.

This method uses recursive calls, which increases the time required to construct the candidate set $\boldsymbol{\mathcal{H}}_0^{'}$. In Ref. [20], another method using breadth-first search was proposed to construct key candidates. This is based on a construction method for QC-LDPC McEliece cryptosystems [12], which translates the key reconstruction to a search problem for d_v-clique in a graph corresponding to the distance spectrum. Algorithm 3 shows the explicit procedure of this breadth-first key reconstruction.

Algorithm 3 Breadth-first key key reconstruction

Input: $D(\boldsymbol{h_0})$, $p_0 = 0$, $p_1 = \min(D(\boldsymbol{h_0}))$, r: length of the secret key.
Output: set of secret key candidates $\boldsymbol{\mathcal{H}}_0^{'}$.
1: $\boldsymbol{\mathcal{H}}_0^{'} \leftarrow \{\}$.
2: $\overline{D(\boldsymbol{h_0})} = \{p_0\} \cup D(\boldsymbol{h_0}) \cup (r - D(\boldsymbol{h_0})))$.
3: $S_0 \leftarrow \{p_0, p_1\}$.
4: $S_0^{'} \leftarrow \{b \in \overline{D(\boldsymbol{h_0})} \setminus S_0 \mid d(b, p_1) \in D(\boldsymbol{h_0})\}$.
5: $N \leftarrow 1, t \leftarrow 2$.

6: $\mathcal{S} = \{S_0\}$.
7: $\mathcal{S}' = \{S_0'\}$
8: **while** $t < d_v$ and $N > 0$ **do**
9: $\tilde{\mathcal{S}} \leftarrow \{\}$, $\tilde{\mathcal{S}}' \leftarrow \{\}$, $n = 0$.
10: **for all** i $(0 \le i < N)$ **do**
11: $S_i' = \{a_0, a_1, \cdots\}$.
12: **for all** l $(0 \le l < |S_i'|)$ **do**
13: $S_{i,l} \leftarrow S_i \cup \{a_l\}$.
14: $S_{i,l}' \leftarrow \{b \in S_i' \setminus \{a_l\} \mid b > a_l, d(b, a_l) \in D(\boldsymbol{h_0})\}\}$.
15: $B_{i,l} \leftarrow S_{i,l} \cup S_{i,l}'$.
16: **if** $(B_{i,l})_{\frac{d_v+1}{2}} \ge \lceil \frac{r+p_1}{2} \rceil$ **then**
17: goto line 24.
18: **end if**
19: **if** $|B_{i,l}| = d_v$ and $D(B_{i,l}) = D(\boldsymbol{h_0})$ **then**
20: append $B_{i,l}$ to $\mathcal{H}_0$.
21: **else if** $|B_{i,l}| > d_v$ **then**
22: append $S_{i,l}$ to $\tilde{\mathcal{S}}$, $S_{i,l}'$ to $\tilde{\mathcal{S}}'$ as the nth elements.
23: $n \leftarrow n + 1$.
24: **end if**
25: **end for**
26: **end for**
27: $\mathcal{S} \leftarrow \tilde{\mathcal{S}}, \mathcal{S}' \leftarrow \tilde{\mathcal{S}}'$, $N \leftarrow n$, and $t \leftarrow t + 1$.
28: **end while**
29: **for all** i $(0 \le i < N)$ **do**
30: **if** $D(S_i) = D(\boldsymbol{h_0})$ **then**
31: append S_i to $\mathcal{H}_0$.
32: **end if**
33: **end for**
34: **return** $\mathcal{H}_0$.

We will explain the procedure more precisely. First, an augmented distance spectrum is defined as

$$\overline{D(\boldsymbol{h_0})} := \{0\} \cup D(\boldsymbol{h_0}) \cup (r - D(\boldsymbol{h_0})), \tag{10}$$

where $(r - D(\boldsymbol{h_0}))$ represents the set whose elements are obtained by subtracting each element of $D(\boldsymbol{h_0})$ from the key length r. This augmented distance spectrum $\overline{D(\boldsymbol{h_0})}$ is the set of possible positions of ones in the key when we set $p_0 = 0$. At the t-th step $(t > 2)$, we have a family of sets $\mathcal{S}^{(t)} = \{S_0^{(t)}, S_1^{(t)}, \ldots, S_{N-1}^{(t)}\}$ and its complement $\mathcal{S}'^{(t)} = \{S_0'^{(t)}, S_1'^{(t)}, \ldots, S_{N-1}'^{(t)}\}$. Picking up an element $a_l \in S_i'^{(t)}$ $(0 \le i < N)$, we make a new pair of sets,

$$\begin{aligned} S_{i,l}^{(t+1)} &= S_i \cup \{a_l\}, \\ S_{i,l}^{(t+1)'} &= \{b \in S_i' \setminus \{a_l\} \mid b > a_l, b - a_l \in D(\boldsymbol{h_0})\}, \end{aligned}$$

where their union $B_{i,l}^{(t+1)} := S_{i,l}^{(t+1)} \cup S_{i,l}'^{(t+1)}$ can be a candidate of the key. When the size of $B_{i,l}^{(t+1)}$ is less than d_v, it cannot be a candidate, and then we discard

it. When the size equals to d_v, we check whether $D(B_{i,l}) = D(\boldsymbol{h_0})$ is satisfied. If it is, we append $B_{i,l}^{(t+1)}$ into $\boldsymbol{\mathcal{H}}_0^{'}$ as a candidate; otherwise, it is discarded. When the size is larger than d_v, $S_{i,l}^{(t+1)}$ and $S_{i,l}^{'(t+1)}$ need to be further sieved, and then proceed to the next step by being appended into $\mathcal{S}^{(t+1)}$ and $\mathcal{S}^{'(t+1)}$, respectively. It is worth mentioning that as in Algorithm 1 and 2, the line 21 of Algorithm 3 prune latter-concentrated candidates.

As aforementioned, the distance spectrum may not uniquely determine the secret key. Thus, in practice, after collecting all candidates $\boldsymbol{\mathcal{H}}_0^{'}$, we need to decode messages using them, their cyclic shifts, and mirror images. A candidate that successfully decodes messages will be the true secret key. However, many numerical experiments imply that the distance spectrum should uniquely determine the recovered key up to the cyclic shifts and mirror images; this may be due to the sparsity of the secret key [20].

4 Key reconstruction from imperfect distance spectrum

Clearly estimating the distance spectrum is in general a time-consuming task. For instance, when we consider the 80-bit security parameters $(n, r, d_v, n_0) = (9602, 4801, 45, 2)$, it is implied that we need $M = 100,000$ or more trials to separate the DERs [13]. As shown in Fig. 1, 10000 trials cannot sufficiently separate DERs for $d \in D(\boldsymbol{h_0})$ and $d \notin D(\boldsymbol{h_0})$: one can see that some red (blue) points representing $d \notin D(\boldsymbol{h_0})$ ($\in D(\boldsymbol{h_0})$) is beneath (above) the threshold. More trials make spectrum estimation more precise, but it takes a longer time. Figure 2 shows DERs with 5000 trials, which halves the time required for the estimation. Clearly, it is more difficult to sufficiently separate the DERs than in the case of $M = 10000$. Therefore, we cannot determine a clear threshold T.

To deal with this inconvenience, we define an interval $\mathcal{T} = [T_1, T_2](T_1 < T_2)$, where it is unclear whether the distance d belongs to $\mu(d) = 0$ or $\mu(d) = 1$ (Fig. 2). Assume that if DER_d is larger than T_2, we consider that the distance d does not exist in $D(\boldsymbol{h_0})$. Meanwhile, if a distance d has DER_d less than T_1, it is assumed to exist in $D(\boldsymbol{h_0})$ certainly. These assumptions will be justified by using an appropriate classification method. Although we do not go into the details of what method should be employed in practice, we provide a proposal to define the interval $\mathcal{T}$ in Appendix A.

The set of distances d with such ambiguous DERs, which we refer to as suspicious distances, is defined as

$$\mathrm{sus} := \{d \in [U] \mid \mathrm{DER}_d \in \mathcal{T}\}.$$

It is assumed that the "true" threshold T, which will perfectly separate $\mu(d) = 0$ and $\mu(d) = 1$ at the limit of the infinite number of trials, exists in the region $\mathcal{T}$.

4.1 Upper and lower spectra

Our goal here is to consider key reconstruction algorithms, which can be valid even when we only have the region $\mathcal{T}$ instead of the threshold T. In this case, we cannot obtain the true distance spectrum $D(\boldsymbol{h_0})$ of the secret key, and hence the reconstruction algorithms need to be modified. Note that if $\mathcal{T}$ is correctly specified, we acquire an "upper" and a "lower" spectra of the true spectrum.

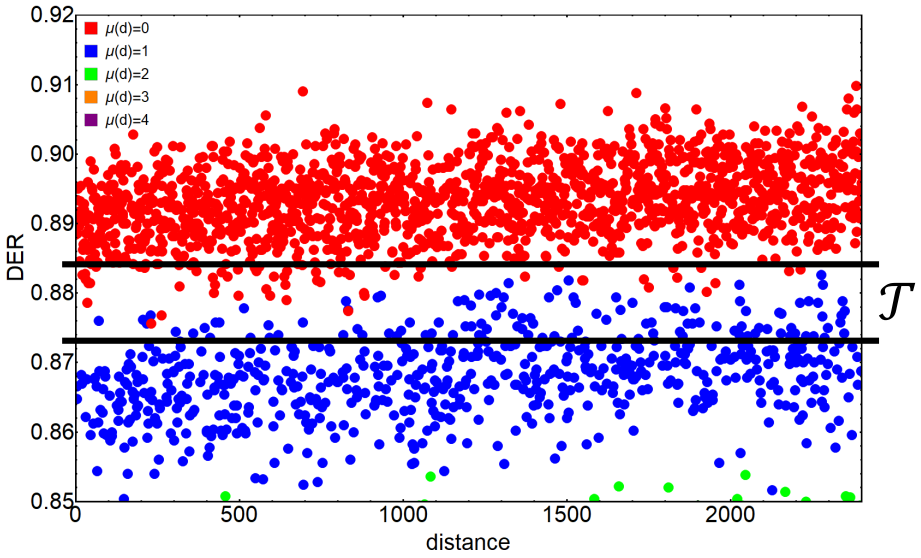

Fig. 2. Empirical DERs for all distances. The parameters are taken to be $(n, r, d_v, t) = (9602, 4801, 45, 110)$ and $M = 5000$. The red points represent DERs for $\mu_{\boldsymbol{h_0}}(d) = 0$ while the blue points does $\mu_{\boldsymbol{h_0}}(d) = 1$. A region sandwiched by the black lines is $\mathcal{T}$.

First, consider distances d with $\mathrm{DER}_d \leq T_1$. As aforementioned, such distances d are guaranteed to be in the true spectrum. We call the collection of these spectrum the "lower" spectrum $\mathscr{d}(\boldsymbol{h_0})$. This relates to the true spectrum as follows:

$$\mathscr{d}(\boldsymbol{h_0}) = D(\boldsymbol{h_0}) \setminus \mathrm{sus},$$

and therefore, it trivially satisfies the relation:

$$\mathscr{d}(\boldsymbol{h_0}) \subset D(\boldsymbol{h_0}).$$

Next, consider distances d with $\mathrm{DER}_d > T_2$. These are guaranteed *not* to be in the true spectrum. We refer to the complement of the collection of them, i.e., the set of distances with $\mathrm{DER}_d < T_2$, as the "upper " spectrum $\mathscr{D}(\boldsymbol{h_0})$. The upper spectrum contains the distances that *may* be in the true one. This relates to the true spectrum as

$$\mathscr{D}(\boldsymbol{h_0}) = D(\boldsymbol{h_0}) \cup \mathrm{sus},$$

and thus,

$$D(\boldsymbol{h_0}) \subset \mathscr{D}(\boldsymbol{h_0}).$$

The algorithm for constructing the upper spectrum $\mathscr{D}(\boldsymbol{h_0})$ and the lower one $\mathscr{d}(\boldsymbol{h_0})$ is provided in Algorithm 4.

Algorithm 4 Compute the lower and upper spectra

Input: T_1, T_2, number of decoding trials M per distance, upper distance U.
Output: $\mathscr{D}(\boldsymbol{h_0})$, $\mathscr{d}(\boldsymbol{h_0})$.
1: $\mathscr{D}(\boldsymbol{h_0}) \leftarrow \{\}$, $\mathscr{d}(\boldsymbol{h_0}) \leftarrow \{\}$.

2: **for all** i $(1 \le i \le U)$ **do**
3: $f \leftarrow 0$.
4: **for all** j $(0 \le j < M)$ **do**
5: take $\boldsymbol{v} \in \Psi_i$.
6: Apply BF decoding to $\boldsymbol{v}$.
7: **if** $\boldsymbol{H}\boldsymbol{v}^T \neq \boldsymbol{0}$ **then**
8: $f \leftarrow f + 1$.
9: **end if**
10: **end for**
11: **if** $f/M \le T_1$ **then**
12: $\mathscr{A}(\boldsymbol{h_0}) \leftarrow \mathscr{A}(\boldsymbol{h_0}) \cup \{i\}$.
13: **end if**
14: **if** $f/M \le T_2$ **then**
15: $\mathscr{D}(\boldsymbol{h_0}) \leftarrow \mathscr{D}(\boldsymbol{h_0}) \cup \{i\}$.
16: **end if**
17: **end for**
18: **return** $\mathscr{D}(\boldsymbol{h_0})$, $\mathscr{A}(\boldsymbol{h_0})$.

4.2 reconstruction algorithms from imperfect spectrum

We propose two algorithms for key candidate construction provided that we have the upper and lower spectra instead of the true spectrum. These can be obtained by slight modification of the two algorithms explained in the previous section. They differ in the following points. First, both algorithms use the upper spectrum $\mathscr{D}(\boldsymbol{h_0})$ for candidate construction. Candidates constructed from this spectrum include those constructed from the true spectrum because of the condition $D(\boldsymbol{h_0}) \subset \mathscr{D}(\boldsymbol{h_0})$. Second, when checking an obtained candidate $\boldsymbol{h_0'}$, we use the lower spectrum $\mathscr{A}(\boldsymbol{h_0})$. In line 14 of Algorithm 2 and lines 19 and 30 of Algorithm 3, we check whether the distance spectrum of the candidate matches the true spectrum or not. Instead, we now check whether the spectrum of the candidate satisfies the following condition:

$$\mathscr{A}(\boldsymbol{h_0}) \subset D(\boldsymbol{h_0'}),$$

which is a necessary condition for $\boldsymbol{h_0'}$ to be the true key because of the trivial relation $\mathscr{A}(\boldsymbol{h_0}) \subset D(\boldsymbol{h_0})$.

The modified recursive algorithm is given in Algorithm 5 and 6. As easily seen, modifications are not that many: we just replace $D(\boldsymbol{h_0})$ in Algorithms by the upper spectrum $\mathscr{D}(\boldsymbol{h_0})$ or the lower one $\mathscr{A}(\boldsymbol{h_0})$.

Algorithm 5 Recursive search for candidates from the imperfect spectrum

Input: $\mathscr{D}(\boldsymbol{h_0})$, $\mathscr{A}(\boldsymbol{h_0})$, $p_0 = 0$, $p_1 = \min(D(\boldsymbol{h_0}))$, $l = 2$.
Output: set of secret key candidates $\mathscr{K}_0'$
1: $\mathscr{K}_0' \leftarrow \{\}$.
2: $\boldsymbol{h_0'} \leftarrow \{p_0, p_1\}$.
3: Call $\mathscr{R}'(\boldsymbol{h_0'}, l = 2)$ in Algorithm 6.
4: **return** $\mathscr{K}_0'$.

Algorithm 6 Recursive function $\mathscr{R}'(\boldsymbol{h_0'}, l)$ for the imperfect spectrum

Input: $\boldsymbol{h}_{\boldsymbol{0}}^{'} = \{p_0, p_1, \cdots, p_{l-1}\}$, l.
1: **for all** i $(p_{l-1} + 1 \leq i \leq r)$ **do**
2: **for all** j $(0 \leq j \leq l-1)$ **do**
3: **if** $d(i, p_j) \notin \mathscr{D}(\boldsymbol{h_0})$ **then**
4: goto line 25.
5: **end if**
6: **end for**
7: $p_l \leftarrow i$.
8: append p_l to $\boldsymbol{h}_{\boldsymbol{0}}^{'}$.
9: **if** $l \leq \frac{d_v+1}{2}$ **and** $p_l \geq \left\lceil \frac{r+p_1}{2} \right\rceil$ **then**
10: goto line 25.
11: **end if**
12: **if** $l = d_v - 1$ **then**
13: Calculate $D(\boldsymbol{h}_{\boldsymbol{0}}^{'})$.
14: **if** $\mathscr{d}(\boldsymbol{h_0}) \subset D(\boldsymbol{h}_{\boldsymbol{0}}^{'})$ **then**
15: append $\boldsymbol{h}_{\boldsymbol{0}}^{'}$ to $\boldsymbol{\mathcal{H}}_{0}^{'}$.
16: goto line 25.
17: **else**
18: goto line 25.
19: **end if**
20: **end if**
21: $l \leftarrow l + 1$.
22: Recursive call of $\mathscr{R}(\boldsymbol{h}_{\boldsymbol{0}}^{'}, l)$.
23: Remove p_l from $\boldsymbol{h}_{\boldsymbol{0}}^{'}$.
24: $l \leftarrow l - 1$.
25: **end for**
26: **return**

The modification of the breadth-first algorithm is obtained in a similar way to the recursive one. The construction process utilizes the upper spectrum instead of the true spectrum while the checking process employs the lower one. All other processes are performed in the same way as in Algorithm 3.

Algorithm 7 Breadth-first key reconstruction from the imperfect spectrum

Input: $\mathscr{D}(\boldsymbol{h_0})$, $\mathscr{d}(\boldsymbol{h_0})$, $p_0 = 0$, $p_1 = \min(D(\boldsymbol{h_0}))$, r: length of the secret key.
Output: set of secret key candidates $\boldsymbol{\mathcal{H}}_{0}^{'}$.
1: $\boldsymbol{\mathcal{H}}_{0}^{'} \leftarrow \{\}$.
2: $\overline{\mathscr{D}(\boldsymbol{h_0})} = \{p_0\} \cup \mathscr{D}(\boldsymbol{h_0}) \cup (r - \mathscr{D}(\boldsymbol{h_0})))$.
3: $S_0 \leftarrow \{p_0, p_1\}$.
4: $S_0^{'} \leftarrow \{b \in \overline{\mathscr{D}(\boldsymbol{h_0})} \setminus S_0 \mid d(b, p_1) \in \mathscr{D}(\boldsymbol{h_0})\}$.
5: $N \leftarrow 1, t \leftarrow 2$.
6: $\mathcal{S} = \{S_0\}$.
7: $\mathcal{S}^{'} = \{S_0^{'}\}$
8: **while** $t < d_v$ and $N > 0$ **do**
9: $\tilde{\mathcal{S}} \leftarrow \{\}$, $\tilde{\mathcal{S}}^{'} \leftarrow \{\}$, $n = 0$.
10: **for all** i $(0 \leq i < N)$ **do**
11: $S_i^{'} = \{a_0, a_1, \cdots\}$.
12: **for all** l $(0 \leq l < \left|S_i^{'}\right|)$ **do**
13: $S_{i,l} \leftarrow S_i \cup \{a_l\}$.

$S'_{i,l} \leftarrow \{b \in S'_i \setminus \{a_l\} \mid b > a_l, d(b, a_l) \in \mathscr{D}(\boldsymbol{h_0})\}\}$.
$B_{i,l} \leftarrow S_{i,l} \cup S'_{i,l}$.
if $\left(B_{i,l}\right)_{\frac{d_v+1}{2}} \geq \left\lceil \frac{r+p_1}{2} \right\rceil$ **then**
  goto line 24.
end if
if $\left|B_{i,l}\right| = d_v$ and $\mathscr{A}(\boldsymbol{h_0}) \subset D(B_{i,l})$ **then**
  append $B_{i,l}$ to $\boldsymbol{\mathcal{H}}_0$.
else if $\left|B_{i,l}\right| > d_v$ **then**
  append $S_{i,l}$ to $\tilde{\mathcal{S}}$, $S'_{i,l}$ to $\tilde{\mathcal{S}}'$ as the nth elements.
  $n \leftarrow n + 1$.
end if
end for
end for
$\mathcal{S} \leftarrow \tilde{\mathcal{S}}, \mathcal{S}' \leftarrow \tilde{\mathcal{S}}'$, $N \leftarrow n$, and $t \leftarrow t + 1$.
end while
for all i $(0 \leq i < N)$ **do**
  if $\mathscr{A}(\boldsymbol{h_0}) \subset D(S_i)$ **then**
    append S_i to $\boldsymbol{\mathcal{H}}_0$.
  end if
end for
return $\boldsymbol{\mathcal{H}}_0$.

4.3 Numerical experiments

We demonstrate numerical experiments of key candidate construction from the upper and lower spectra. We fisrt consider the 80-bit security parameters shown in Table 1: $(n, r, d_v, n_0) = (9602, 4801, 45, 2)$. Ten different binary vectors are randomly generated as secret keys. The suspicious set sus, which makes the distance spectrum imperfect, is also randomly generated: Note that we do not perform actual processes of obtaining the distance spectrum. The size of sus, $s = |\mathrm{sus}|$ is used as a parameter representing the spectrum imperfection. We perform both construction algorithms for each secret key while increasing s.

Table 3 exhibits the time required for constructing the set of key candidates $\boldsymbol{\mathcal{H}}'_{\boldsymbol{0}}$ with changes in s for each algorithm. From top to bottom, it shows the minimum, average, and maximum construction time for the ten secret keys. These experiments are conducted using a 12th Gen Intel(R) Core(TM) i9-12900KF.

As one can see, both algorithms succeed to construct the set of candidates in feasible time scales when s is small. The construction time of both methods exponentially increases as s increases. In all cases, Breadth-first method terminates in a shorter time than the recursive one, which is the same behavior observed in Ref. [20]. Surprisingly, we could not find non-trivially different candidates from the secret key even when $s = 600$. This may suggest that the sparsity of the key is a strong condition for uniquely determining the key from its spectrum.

Table 3. The construction time (in seconds) of the set of candidates $\mathcal{H}_0'$ for both methods. The security parameters are set as $(n, r, d_v, n_0) = (9602, 4801, 45, 2)$. $s = |\mathrm{sus}|$ represents the spectrum imperfection. The minimum, average, and maximum construction times for ten secret keys are shown from top to bottom in each cell.

s	Recursive construction (Algorithm 5 and 6)	Breadth-first construction (Algorithm 7)
0	8.29 101 496	1.61 1.78 1.86
100	18.8 174 691	1.81 1.97 2.10
200	63.7 250 812	2.39 2.83 3.42
300	233 514.9 1221	4.56 6.25 8.77
400	883 1539 2844	13.2 20.2 28.3
500	3616 6483 11413	51.6 89.1 127
600	20106 34814 50459	281 483 750

We also evaluate the time required for constructing a secret key in the 128-bit security parameters of BIKE, $(n, r, d_v, n_0) = (24646, 12323, 71, 2)$ in a similar manner. The recursive search for five instances out of ten fails to construct secret keys within the time limit (7days) even with the perfect spectrum. Meanwhile, the breadth-first search succeeds key construction with $s = 600$ spectrum imperfection.

Table 4. The construction time (in seconds) of the set of candidates $\mathcal{H}_0'$ for both methods. The security parameters are set as $(n, r, d_v, n_0) = (24646, 12323, 71, 2)$: the 128-bit security parameter of BIKE. $s = |\mathrm{sus}|$ represents the spectrum imperfection. The minimum, average, and maximum construction times for ten secret keys are shown from top to bottom in each cell. We set the time limit of experiments to be 7days (= 604800 s).

s	Recursive construction (Algorithm 5 and 6)	Breadth-first construction (Algorithm 7)
0	30684.63 - >7days	2.84 3.86 4.93
100	37715.74 - >7days	6.43 8.61 11.00
200	39567.36 - >7days	18.01 24.51 30.43
300	43557.87 - >7days	60.54 82.74 117.24
400	51930.30 - >7days	208.41 304.91 508.32
500	68510.49 - >7days	618.11 1210.65 2566.46
600	99027.03 - >7days	1993.89 4661.15 11516.88

5 Summary

The key recovery attack on QC-MDPC McEliece cryptosystems (QC-MDPC scheme) proposed in Ref. [14] consists of two steps. In the first step, an attacker sends specific messages many times and estimates the distance spectrum of the secret key, which is the collection of distances of all pairs of ones in the key. In the second step, the attacker constructs candidates of the key from the estimated distance spectrum. If the number of messages is sufficiently large, they acquire the perfect distance spectrum, and the second step can easily be performed. In practice, sending that many messages is time-consuming. Therefore, the attacker will only send a finite number of messages, which makes the estimation imperfect. For example, the perfect estimation of the spectrum requires over 100,000 trials when we consider the QC-MDPC scheme with the 80-bit security parameters $(n, r, d_v, n_0) = (9602, 4801, 45, 2)$ and the error number $t = 110$.

In this paper, we have proposed two algorithms than can construct key candidates from an imperfect distance spectrum. In our algorithms, we utilize "upper and lower bounds" of the true spectrum. One of our algorithms is based on the recursive search, which is originally proposed in Ref. [14] while the other is based on the breadth-first search proposed in Ref. [20]. We have shown that both algorithms successfully construct key candidates from an imperfect distance

spectrum up to some level of the imperfection for the 80-bit security parameters $(n, r, d_v, n_0) = (9602, 4801, 45, 2)$. This implies that even when the number of trials in the first step is insufficient to estimate the perfect spectrum, it will still be possible to construct key candidates. The time required for the second step will be almost negligible compered to that for the first step even under the spectrum imperfection; therefore, reducing the number of trials as much as possible will probably be efficient. This will pave the way into shortening the total time required for the key recovery attack. Furthermore, we found that the construction time of the breadth-first recovery algorithm is shorter than the recursive one. This is the same behavior as in the case where the distance spectrum is fully acquired [20]. More importantly, for the 128-bit security of BIKE, only the breadth-first recovery succeeds key construction within the time limit of 7 days.

We need to evaluate the imperfection level where the key construction does not succeed any more. This will be our future work. Also, another type of imperfection is worth investigating. In this paper, we have assumed that the imperfection in the distance spectrum occurs randomly. Reference [25] suggested that the imperfection may intensively be distributed in the latter half of the spectrum when one uses Black-Gray-Flip decoding [25]. Therefore, it will be important to evaluate the key construction in the case where the imperfection is concentrated in the latter half of the spectrum. Also, it is worth mentioning that our approach will be applicable not only to the original GJS reaction attack but alto to a timing attack proposed in Ref. [11].

Another method for constructing secret keys from imperfect spectrum was provided in Ref [21], and should be compared with our method. An important difference between their method and ours is that they use two (and more) distance spectra of $\boldsymbol{h_0}$ and $\boldsymbol{h_1}$, which is the first row of $\boldsymbol{H_1}$. As obtaining two distance spectra is time-consuming compared with obtaining one, our method may be advantageous to theirs, but careful considerations are required.

A How to determine suspicious interval

Here, we provide how to define the suspicious interval $\mathcal{T}$ whereby the suspicious distances are classified. According to our numerical experiments and those in Ref. [14], DER_d's with distances d having a fixed multiplicity $\mu(d) = k$ looks to be sampled from a Gaussian distribution, whose average and variance depend on k. Thus, we employ the Gaussian Mixture distribution to classify the distance multiplicities of distances in a secret key.

The Gaussian Mixture distribution is defined as

$$p(x) = \sum_{k=0}^{K} \pi_k \mathcal{N}(x|m_k, \sigma_k), \quad \sum_{k=0}^{K} \pi_k = 1, \tag{11}$$

where k represents a multiplicity, π_k is the probability of appearance of the multiplicity k in a secret key, and $\mathcal{N}(m_k, \sigma_k)$ is the Gaussian distribution with the average m_k and the variance σ_k. DER's obtained for a secret key with ignorance on the multiplicity will approximately be generated from this distribution. The standard deviation will obey the central limit theorem, and therefore decrease as $1/\sqrt{M}$, where M is the trial number for computing each DER_d. the number of Gaussian distributions K should be taken depending on the upper bound of the

multiplicity k, which is related with the density of a key in considered security parameters.

To estimate the parameters π_k, m_k, and σ_k, we employ the expectation-maximization algorithm [9] using U experimental data of DER's. The explicit algorithm is shown in Algorithm 8.

Algorithm 8 Estimate π_k, m_k, and σ_k from experimental data

Input: distribution number K, upper distance U, iteration L, experimental data DER_d $(1 \le d \le U)$, acceptable difference δ $(\ll 1)$.
Output: π_k, m_k, σ_k $(0 \le k \le K)$.

1: Initialize π_k, m_k, and σ_k.
2: $L_{\mathrm{prev}} = \sum_{d=1}^{U} \ln\left(\sum_{k=0}^{K} \pi_k \mathcal{N}(\mathrm{DER}_d | m_k, \sigma_k)\right)$.
3: **for all** l $(1 \le l \le L)$ **do**
4: **for all** k $(0 \le k \le K)$ **do**
5: **for all** d $(1 \le d \le U)$ **do**
6: $\gamma_{d,k} \leftarrow \frac{\pi_k \mathcal{N}(\mathrm{DER}_d | m_k, \sigma_k)}{\sum_{j=0}^{K} \pi_j \mathcal{N}(\mathrm{DER}_d | m_j, \sigma_j)}$.
7: **end for**
8: $N_k = \sum_{d=1}^{U} \gamma_{d,k}$.
9: **end for**
10: $N = \sum_{k=0}^{K} N_k$.
11: **for all** k $(0 \le k \le K)$ **do**
12: $\pi_k = N_k / N$.
13: $m_k = \frac{1}{N_k} \sum_{d=1}^{U} \gamma_{d,k} \cdot \mathrm{DER}_d$.
14: $\sigma_k = \frac{1}{N_k} \sum_{d=1}^{U} \gamma_{d,k} \, (\mathrm{DER}_d - m_k)^2$.
15: **end for**
16: $L = \sum_{d=1}^{U} \ln\left(\sum_{k=0}^{K} \pi_k \mathcal{N}(\mathrm{DER}_d | m_k, \sigma_k)\right)$
17: **if** $\left|L - L_{\mathrm{prev}}\right| \le \delta$ **then**
18: goto line 22.
19: **end if**
20: $L_{\mathrm{prev}} = L$
21: **end for**
22: **return** π_k, m_k, σ_k $(0 \le k \le K)$.

We demonstrate the above estimation procedure for a randomly generated binary vector with the bit length 4801 and the weight 45, which correspond to the 80-bit security parameters discussed in the main text. The number of Gauss distributions K is taken to be 3 because the multiplicities more than 4 hardly occur for this density. Experimental data of DER's are calculated with the trial number $M = 5000,\ 10000$ for each distance. Figure 3 shows good agreement between the histogram of experimental data and the calculated Gaussian mixture distributions with appropriate scaling.

Using the calculated parameters $\{\pi_k, m_k, \sigma_k\}_{k=0}^{K}$, we can estimate the suspicious interval $\mathcal{T}$ defined by T_1 and T_2. Let us first consider how to determine T_1. If DER_d is less than T_1, the multiplicity $\mu(d)$ is almost certainly non-zero. The probability of appearance of the multiplicity k in U sample data is roughly estimated as π_k; therefore, the number of distances with the multiplicity k in the sample data will be $U\pi_k$. The probability that all distances with $k = 0$ have

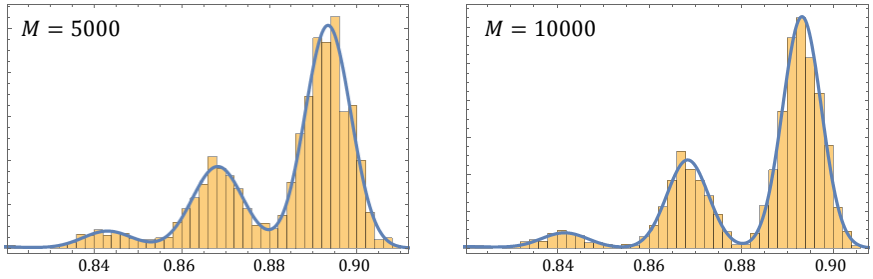

Fig. 3. The histgram of experimental data and the calculated Gaussian mixture distributions. In both panels, the blue lines represent the calculated distributions. The left panel shows them for the trial number $M = 5000$ while the right one shows them for $M = 10000$.

DER_d more than T_1 is calculated by

$$\left(\int_{T_1}^{1} dy \mathcal{N}(y|m_0, \sigma_0)\right)^{\lceil U\pi_0 \rceil}. \tag{12}$$

By letting this be larger than an appropriate success probability $\alpha(< 1)$, we obtain T_1 as a function of α: $T_1 = T_1(\alpha)$. T_2 is determined in a similar manner:

$$\left(\int_{0}^{T_2} dy \mathcal{N}(y|m_1, \sigma_1)\right)^{\lceil U\pi_1 \rceil} > \beta \Longrightarrow T_2 = T_2(\beta), \tag{13}$$

where $\beta(< 1)$ is an appropriately chosen success probability. Here we ignore the contribution of the distributions for k more than 2. The total probability of successfully determining T_1 and T_2 is roughly estimated as $\alpha\beta$. Obviously, taking higher α and β gives a higher success probability while it gives a larger number of suspicious distances |sus|. In our experiments with $(n, r, d_v) = (9602, 4801, 45)$, taking $\alpha = \beta = 0.9$, we found that |sus| is roughly 400 for $M = 5000$ and |sus| = 50 for $M = 10000$. If we obtain $T_1 > T_2$, it is better to define a threshold T rather than the interval $\mathcal{T}$. For instance, $T = (T_1 + T_2)/2$ will be a good threshold.

References

1. Post-Quantum Cryptography. Available: `https://csrc.nist.gov/Projects/post-quantum-cryptography/`.
2. QC-MDPC decoder. `https://github.com/vvasseur/qcmdpc\_decoder`.
3. N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor, et al. BIKE: bit flipping key encapsulation. 2017.
4. H. Bartz and G. Liva. On decoding schemes for the MDPC-McEliece cryptosystem. In *SCC 2019; 12th International ITG Conference on Systems, Communications and Coding*, pages 1–6. VDE, 2019.

5. T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani. Reducing key Length of the McEliece Cryptosystem. In *International Conference on Cryptology in Africa*, pages 77–97. Springer, 2009.
6. E. Berlekamp, R. McEliece, and H. Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.
7. I. E. Bocharova, T. Johansson, and B. D. Kudryashov. Improved iterative decoding of QC-MDPC codes in the McEliece public key cryptosystem. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 1882–1886. IEEE, 2019.
8. J. Bolkema, H. Gluesing-Luerssen, C. A. Kelley, K. E. Lauter, B. Malmskog, and J. Rosenthal. Variations of the McEliece Cryptosystem. In *Algebraic Geometry for Coding Theory and Cryptography: IPAM, Los Angeles, CA, February 2016*, pages 129–150. Springer, 2017.
9. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.
10. N. Drucker, S. Gueron, and D. Kostic. QC-MDPC Decoders with Several Shades of Gray. In *International Conference on Post-Quantum Cryptography*, pages 35–50. Springer, 2020.
11. E. Eaton, M. Lequesne, A. Parent, and N. Sendrier. QC-MDPC: a timing attack and a CCA2 KEM. In *International conference on post-quantum cryptography*, pages 47–76. Springer, 2018.
12. T. Fabšič, V. Hromada, P. Stankovski, P. Zajac, Q. Guo, and T. Johansson. A reaction attack on the QC-LDPC McEliece cryptosystem. In *Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8*, pages 51–68. Springer, 2017.
13. Q. Guo, T. Johansson, and P. S. Wagner. A key recovery reaction attack on qc-mdpc. *IEEE Transactions on Information Theory*, 65(3):1845–1861, 2018.
14. Q. Guo, T. Johansson, and P. S. Wagner. A key recovery reaction attack on QC-MDPC. *IEEE Transactions on Information Theory*, 65(3):1845–1861, 2019.
15. H. Kaneko. Look-Ahead Bit-Flipping Decoding of MDPC Code. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 2922–2927. IEEE, 2022.
16. K. Kurosawa and K. Inaba. Consideration on the np completeness of linear codes. *The transactions of the Institute of Electronics, Information and Communication Engineers. A*, J68-A(9):953–956, 1985.
17. R. McEliece. A public key cryptosystem based on algebraic coding theory. *DSN Prog. Re.*, 42-44:114–116, 1978.
18. R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *2013 IEEE international symposium on information theory*, pages 2069–2073. IEEE, 2013.
19. A. Nilsson, I. E. Bocharova, B. D. Kudryashov, and T. Johansson. A Weighted Bit Flipping Decoder for QC-MDPC-based Cryptosystems. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1266–1271. IEEE, 2021.
20. M. Ohtsuka, T. Ishimaru, Y. Tsukie, S. Kukita, and K. Watanabe. Efficient reconstruction in key recovery attack on the qc-mdpc mceliece cryptosystems. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 2024.
21. T. B. Paiva and R. Terada. Improving the efficiency of a reaction attack on the QC-MDPC McEliece. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 101(10):1676–1686, 2018.
22. M. Repka and P. Zajac. Overview of the McEliece cryptosystem and its security. *Tatra Mountains Mathematical Publications*, 60(1):57–83, 2014.
23. N. Sendrier and V. Vasseur. About low DFR for QC-MDPC decoding. In *International Conference on Post-Quantum Cryptography*, pages 20–34. Springer, 2020.

24. P. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.
25. K. Xagawa, A. Ito, R. Ueno, J. Takahashi, and N. Homma. Fault-injection attacks against nist's post-quantum cryptography round 3 kem candidates. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part II 27*, pages 33–61. Springer, 2021.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Declaration. The views expressed here are the authors' own and do not represent any organization the authors are affiliated with.