A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors

Qian Guo[1⋆], Thomas Johansson[1⋆], and Paul Stankovski[1⋆]

Dept. of Electrical and Information Technology, Lund University, Lund, Sweden
{qian.guo,thomas.johansson,paul.stankovski}@eit.lth.se

Abstract. Algorithms for secure encryption in a post-quantum world are currently receiving a lot of attention in the research community, including several larger projects and a standardization effort from NIST. One of the most promising algorithms is the code-based scheme called QC-MDPC, which has excellent performance and a small public key size. In this work we present a very efficient key recovery attack on the QC-MDPC scheme using the fact that decryption uses an iterative decoding step and this can fail with some small probability. We identify a dependence between the secret key and the failure in decoding. This can be used to build what we refer to as a distance spectrum for the secret key, which is the set of all distances between any two ones in the secret key. In a reconstruction step we then determine the secret key from the distance spectrum. The attack has been implemented and tested on a proposed instance of QC-MDPC for 80 bit security. It successfully recovers the secret key in minutes.
A slightly modified version of the attack can be applied on proposed versions of the QC-MDPC scheme that provides IND-CCA security. The attack is a bit more complex in this case, but still very much below the security level. The reason why we can break schemes with proved CCA security is that the model for these proofs typically does not include the decoding error possibility.

Keywords: CCA-security, key-recovery attack, post-quantum cryptography, QC-MDPC, reaction attack.

1 Introduction

Given the existence of a large quantum computer, cryptosystems based on factoring or discrete logarithm will no longer be secure, as a quantum computer is able to solve both problems in polynomial time [33]. However, it is not yet known to what extent a future quantum computer can be used to successfully solve other types of problems. New algorithms for secure encryption in a post-quantum world (when large quantum computers exist) are currently receiving a lot of attention in the research community, including several larger projects and a standardization effort from NIST [9]. It is often mentioned that the new

⋆ Supported by the Swedish Research Council (Grants No. 2015-04528).

schemes could be from one of the areas: lattice-based, code-based, hash-based and multi-variate [5].

For code-based schemes, the basic construction is the McEliece public-key cryptosystem (PKC) [28], based on the hardness of decoding a random linear code. The general idea is to transform polynomially solvable instance of the problem into something that looks like a random instance. In this case we transform the generator matrix of a code with simple and efficient decoding to a generator matrix for a randomly looking code. Not knowing the inverse of this transformation, the attacker is facing a presumably hard problem, namely, decoding the random code.

The McEliece PKC has been extensively analyzed over a period of more than thirty years, and is still regarded secure in its original form using Goppa codes. Several other underlying codes have been proposed, but many of them have been broken [30]. A problem with the original McEliece construction is the size of the public key. McEliece proposed to use the generator matrix of a linear code as public key. The public key for the originally proposed parameters is roughly 500 Kbits. Although this can be managed today, it has motivated various attempts to decrease the key sizes but most of them have been unsuccessful.

Recently however, a very interesting version of the McEliece PKC was proposed, the QC-MDPC scheme [29]. This is a McEliece PKC that uses so-called moderate density parity check codes (MDPC codes) in quasi-cyclic (QC) form. The quasi-cyclic form allows us to represent a matrix by its first row, which leads to a small public key. As the MDPC codes have a random component, there is no need for scrambling and permutation matrices. Instead, the generator matrix is presented in systematic form. The QC-MDPC proposal with suitable parameters is yet unbroken and it is particularly interesting because of its simplicity and smaller key size.

An European initiative, PQCRYPTO, sponsored by the European Commission under its Horizon 2020 Program ICT-645622, is »developing cryptology that resists the unmatched power of quantum computers«. In September 2015 this group of researchers published a report entitled "Initial Recommendation of long-term secure post-quantum systems" [1], where they recommended several algorithms as being ready for use and several others that warrant further study and may be recommended in coming years. This report recommends the QC-MDPC scheme for further study, confirming its competitiveness as a post-quantum candidate.

Many papers on its implementation have appeared since the introduction of the QC-MDPC scheme. In [15] and [24], the QC-MDPC McEliece is implemented in hardware using the same parameters that we attack in this paper. Implementation with side-channel protection is considered in [25].

1.1 Attack Models and Previous Work

In code-based public-key cryptography, one is typically concerned with two types of attacks: structural attacks and decoding attacks. Structural attacks aim to recover the secret code - *key recovery*, while the decoding attacks target an

intercepted ciphertext and tries to recover the transmitted plaintext - *message recovery*. The plain versions of code-based schemes are designed to be secure in the chosen plaintext attack (CPA) model and it is known that chosen ciphertext attacks (CCA) can break them. To achieve security against adaptive chosen ciphertext attacks (CCA2), the schemes need to be converted. There are several standard conversions to achieve CCA2 security from CPA security, [4,21], and basically the decoding problem is changed in such a way that the noise added in the encryption is no longer in control by Alice who is encrypting.

The standard attacks on the original McEliece scheme can be applied on the QC-MDPC scheme. These attacks are decoding attacks using *information set decoding algorithms*, typically improved versions of the Stern algorithm [3]. These attacks are message recovery attacks and can be applied in a few different scenarios, one of them being the "decoding one-out-of-many" [20,32]. The family of MDPC codes have parity checks of moderate weight (low but not very low). In a structural attack, one can thus consider the dual code, which is given from the generator matrix, and search for low weight codewords in the dual code. This is again done by the same type of algorithms as above. Being well known attacks, the instantiation of QC-MDPC schemes make sure that the computational complexity for these attacks are well beyond the selected security limit. More details can be found in for example [30,31].

For a plain QC-MDPC scheme without CCA2 conversion we can identify a few attacks that require more than the CPA assumption. Using a *partially known plaintext attack* [7], the attacker can reduce the code dimension in the decoding and thus achieve a lower complexity for the information set decoding. In a *resend attack*, Alice is resending the same message twice, or possible two related messages. Also in this case we can efficiently find the message [6]. A *reaction attack* [14] is a weaker version of a chosen ciphertext attack. The attacker sends an intercepted ciphertext with a modification (for example adding a single bit) and observes the reaction of the recipient (but not the result of decoding). Again, one can in certain cases efficiently find the message corresponding to the intercepted ciphertext. It is worth noting that all these attacks are message recovery attacks.

As mentioned before, to achieve a stronger security notion, the QC-MDPC scheme (as any McEliece PKC) can use a CCA2 conversion [21,26]. In this case, the above attacks are no longer possible. So to summarize the current state-of-the-art regarding attacks, for the plain schemes we have possibly some message recovery attacks using the model of reaction attacks. For CCA2 secure versions, we have no known successful attacks.

1.2 Contributions

Our basic scenario is the following. Bob has publicly announced his public key and Alice is continuously sending messages to him using the QC-MDPC scheme. Occasionally, Bob will suffer from a decoding error and will tell Alice, who may retransmit or simply discard sending that message. After sending a number of

messages, Alice will be able to recover Bob's secret key using our proposed attack.

We present a very efficient **key recovery attack** on the QC-MDPC scheme using the fact that decryption uses an iterative decoding step and this can fail with some small probability. We identify a dependence between the secret key and the failure in decoding. This can be used to build what we call a distance spectrum for the secret key, which is the set of all distances between any two ones in the secret key. In a reconstruction step we then determine the secret key from the distance spectrum. The attack has been implemented and tested on a proposed instance of QC-MDPC for 80 bit security. It successfully recovers the secret key in minutes.

A slightly modified version of the attack can be applied on proposed versions of the QC-MDPC scheme that provides CCA2 security. The attack is a bit more complex in this case, but still very much below the security level. The reason why we can break schemes with proved CCA2 security is that the model for these proofs typically does not include the decoding error possibility. A similar situation has been identified and analyzed for the lattice-based scheme NTRU (NTRUEncrypt) [18,19].

The paper is organized as follows. We give some background in Section 2 and describe the QC-MDPC scheme in Section 3. We then present an overview of our new attack in Section 4 and give some related analysis in Section 5. In Section 6 we consider the case when we have a CCA2 converted version and demonstrate that a modified version of the attack is still valid. Section 7 presents some results from implementing the different steps of the attack. Finally, we conclude the paper in Section 8.

2 Background in Coding Theory and Public-Key Cryptography

Let us start by reviewing some basics from coding theory and how it can be applied to public-key cryptography through the McEliece PKC.

Definition 1 (Linear codes) *An $[n, k]$ linear code $\mathcal{C}$ over a finite field $\mathbb{F}_q$ is a linear subspace of $\mathbb{F}_q^n$ of dimension k.*

Definition 2 (Generator matrix) *A $k \times n$ matrix $\mathbf{G}$ with entries from $\mathbb{F}_q$ having rowspan $\mathcal{C}$ is a generator matrix for the $[n, k]$ linear code $\mathcal{C}$.*

Equivalently, $\mathcal{C}$ is the kernel of an $(n-k) \times n$ matrix $\mathbf{H}$ called a *parity-check matrix* of $\mathcal{C}$. We then have $\mathbf{c}\mathbf{H}^{\mathrm{T}} = \mathbf{0}$, if and only if $\mathbf{c} \in \mathcal{C}$, where $\mathbf{H}^{\mathrm{T}}$ denotes the transpose of $\mathbf{H}$.

A code $\mathcal{C}$ can be represented by different generator matrices. An important representation is the systematic form, i.e., when each input symbol are in one-to-one correspondence with a position in the codeword. Then, one can find a $k \times k$ submatrix of $\mathbf{G}$ forming the identity matrix. After a row permutation we

can consider $\mathbf{G}$ in the form $\mathbf{G} = \left(\mathbf{I}\ \mathbf{P}\right)$. If $\mathbf{G}$ has the form $\mathbf{G} = \left(\mathbf{I}\ \mathbf{P}\right)$, then $\mathbf{H} = \left(-\mathbf{P}^{\mathrm{T}}\ \mathbf{I}\right)$.

The Hamming weight $w_{\mathrm{H}}(\mathbf{x})$ of a vector in $\mathbf{x} \in \mathbb{F}_q^n$ is the number of nonzero entries in the vector. The minimum (Hamming) distance of the code $\mathcal{C}$ is defined as $d \stackrel{\text{def}}{=} \min_{\mathbf{x},\mathbf{y}\in\mathcal{C}} w_{\mathrm{H}}(\mathbf{x}-\mathbf{y})$, where $\mathbf{x} \neq \mathbf{y}$. Continuing, we only consider the binary case $q = 2$.

Definition 3 (Quasi-cyclic codes) *An $[n, k]$-quasi-cylic (QC) code $\mathcal{C}$ is a linear block code such that for some integer n_0, every cyclic shift by n_0 is again a codeword.*

In particular, if $n = n_0 k$, then a generator matrix of the form

$$\mathbf{G} = \left(\mathbf{I}\ \mathbf{P}_0\ \mathbf{P}_1 \cdots \mathbf{P}_{n_0-1}\right)$$

is a useful way to represent a QC code, where $\mathbf{P}_i$ is a $k \times k$ cyclic matrix, i.e. the rows (or columns) of $\mathbf{P}$ is obtained by cyclic rotation of the first row one step. Also, the algebra of $k \times k$ binary circulant matrices is isomorphic to the algebra of polynomials modulo $x^k + 1$ over $\mathbb{F}_2$, allowing an alternative description.

Another useful class of codes is the low-density parity-check code (LDPC) defined as a linear code that admits a sparse parity-check matrix $\mathbf{H}$, where sparsity means that each row of $\mathbf{H}$ has at most w ones, for some small w. This sparse matrix can be represented in the form of a bipartite graph, that consists of $n-k$ upper nodes (named "check node") representing the $n-k$ parity equations and n lower nodes (named "variable node") representing the n codeword bits. A variable node is connected to a check node if the variable is present in that parity check. Each check node is then connected to w variable nodes. We call this graph representation a "Tanner" graph, which is a frequently used term in work on iterative decoding algorithms.

2.1 McEliece Cryptosystem

In 1978 McEliece showed how a public key cryptosystem (PKC) could be constructed using tools from coding theory. We shortly describe the original McEliece PKC here. This scheme uses three matrices $\mathbf{G}, \mathbf{S}, \mathbf{P}$, where $\mathbf{G}$ is a $k \times n$ generator matrix of a binary $[n, k, 2t+1]$ linear code. The original and still secure proposal in [28] is to use Goppa codes (see [13,23]). Then $\mathbf{S}$ a $k \times k$ random binary non-singular matrix (called the scrambling matrix), and $\mathbf{P}$ is an $n \times n$ random permutation matrix (called the permutation matrix). As designers we compute the new $k \times n$ matrix $\mathbf{G}' = \mathbf{SGP}$. The scheme works as follows:

- Private Key: $(\mathbf{G}, \mathbf{S}, \mathbf{P})$.
- Public Key: $(\mathbf{G}', t)$
- Encryption: A message $\mathbf{m}$ is mapped to a ciphertext $\mathbf{c}$ by $\mathbf{c} = \mathbf{mG}' + \mathbf{e}$, where $\mathbf{c}$ is the n-bit ciphertext, $\mathbf{m}$ is the k-bit plaintext and $\mathbf{e}$ an n-bit error vector with (Hamming) weight t.

– Decryption: Use an efficient decoding algorithm for Goppa codes to decode $\mathbf{c}$ to find the error $\mathbf{eP}^{-1}$, recover $\mathbf{mS}$ and thus $\mathbf{m}$.

Knowing the description of the selected Goppa code allows efficient decoding, as there are many decoding algorithms for this problem running in polynomial time. But knowing only the public key, the attacker is facing a decoding problem for a code that looks like a random code, a presumably difficult problem.

3 The QC-MDPC Public Key Encryption Scheme

In [29] a new version of the McEliece PKC was proposed. It has a surprisingly simple description and does not use permutation and scrambling matrices as in the original McEliece construction, as well as in other generalizations [22,2] proposed. The idea is to use codes that allow iterative decoding. In coding theory, this usually involves low-density parity check codes, but for an encryption scheme this is not secure. The reason is that LDPC codes have parity-checks with very small Hamming weight (like 3-5) and these parity-checks in a given LDPC code correspond to codewords in the dual code. Since a basis of the dual code can be computed, it is computationally easy to find low-weight codewords in the dual code and hence the low-weight parity checks. The solution proposed in [29] is to increase the weight of the parity checks to a larger value, which is still small in comparison with the dimension of the code. This makes the task of finding low-weight codewords in the dual code much more costly. In this way, key-recovery attacks by algorithms searching for low weight codewords can be avoided.

The family of such codes with increased parity-check weight is called *Moderate-Density Parity-Check* codes (MDPC codes), and they can be decoded with the same decoding algorithms used to decode LDPC codes. The quasi-cyclic variant of MDPC codes are called QC-MDPC codes. These are of special interest, since the quasi-cyclic property allows us to represent the code to be used, by a single row of the generator matrix. Since the public key is the generator matrix, this gives us very compact keys. We will go through the different steps of the QC-MDPC public key cryptosystem as proposed in [29].

Let $r = n - k$.

3.1 Generation of Public-key

1. Choose an $[n, n-r]$ code in the QC-MDPC family described by the parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{r \times n}$, $n = n_0 r$, such that

$$\mathbf{H} = \left(\mathbf{H}_0 \; \mathbf{H}_1 \; \cdots \; \mathbf{H}_{n_0-1}\right),$$

   where each $\mathbf{H}_i$ is a circulant $r \times r$ matrix with weight w_i in each row and with $\hat{w} = \sum w_i$.
2. Generate the public key $\mathbf{G} \in \mathbb{F}_2^{(n-r) \times n}$ from $\mathbf{H}$ as,

$$\mathbf{G} = \left(\mathbf{I} \; \mathbf{P}\right),$$

where

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_{n_0-2} \end{pmatrix} = \begin{pmatrix} \left(\mathbf{H}_{n_0-1}^{-1}\mathbf{H}_0\right)^{\mathrm{T}} \\ \left(\mathbf{H}_{n_0-1}^{-1}\mathbf{H}_1\right)^{\mathrm{T}} \\ \vdots \\ \left(\mathbf{H}_{n_0-1}^{-1}\mathbf{H}_{n_0-2}\right)^{\mathrm{T}} \end{pmatrix}.$$

Again, the QC-MDPC construction has no need for permutation or scrambling matrices.

Encryption Let $\mathbf{m} \in \mathbb{F}_2^{(n-r)}$ be the plaintext. Multiply $\mathbf{m}$ with the public key $\mathbf{G}$ and add noise within the correction radius t of the code, i.e., $\mathbf{c} = \mathbf{mG} + \mathbf{e}$, where $w_{\mathrm{H}}(\mathbf{e}) \leq t$. The parameter t is obtained from the the error correcting capability of the decoding algorithm for the MDPC code [29]. The error vector is uniformly chosen among all binary n-tuples with $w_{\mathrm{H}}(\mathbf{e}) \leq t$.

3.2 Decryption

Let $\mathbf{c} \in \mathbb{F}_2^n$ be a received ciphertext. Given the secret low-weight parity check matrix $\mathbf{H}$, a low-complexity decoding procedure is used to obtain the plaintext $\mathbf{m}$.

The authors of [29] propose a variant of Gallager's bit-flipping algorithm [12] as the decoding procedure of MDPC codes. Here some details on this bit-flipping procedure are presented, which are vital to the proposed key recovery attack in the next section. The decoding algorithm works as follows:

1. Compute the syndrome, $\mathbf{s} = \mathbf{cH}^T$. Since $\mathbf{mH}^T = \mathbf{0}$, the syndrome is equivalently expressed as $\mathbf{s} = \mathbf{eH}^T$. Now consider the Tanner graph for $\mathbf{H}$ and set the initial value in each variable node to 0. Create a counter with an initial value 0 for each variable node.
2. Run through all parity-check equations (rows of $\mathbf{H}$ and check nodes in the graph) and for every variable node connected to an unsatisfied check node, increase its corresponding counter by one.
3. Run through all variable nodes and flip its value if its counter satisfies a certain constraint—which usually is that the counter surpasses a threshold.
4. Check if all the equations are satisfied; if not, reset all the counters to 0 and go to Step 2. The procedure will stop if all the parity-checks are satisfied or if the limit on the maximum number of iterations is reached.

This iterative decoding algorithm commonly used with LDPC codes has an error-correction capability that increases linearly with the length of the code. The good performance of LDPC codes is due to the low-weight parities as the error-correction capability also decreases linearly with the weight of the parity-checks. MDPC codes have slightly higher parity-check weight than LDPC codes and one should anticipate that this influences the error-correction capability.

As expected, the actual performance of this procedure on MDPC codes is relatively poor compared with that on LDPC codes. Along the path of the work [29],

researchers also proposed other variants [15,27] that reduce the decoding error probability further via changing the flipping threshold or introducing more rounds to handle the detected decoding errors. The reduced error probability, however, is still large compared with the corresponding security level[1].

3.3 Proposed Parameters

The authors of [29] proposed the parameters found in Table 1 for a QC-MDPC scheme with 80-bit, 128-bit and 256-bit security level.

Table 1: Some proposed QC-MDPC instances with key size and security level.

Parameters						
n	r	$\hat{w}$	t	n_0	Key size	Security
9602	4801	90	84	2	4801	80
19714	9857	142	134	2	9857	128
65542	32771	274	264	2	32771	256

Results from actual implementations of the QC-MDPC scheme [15,27] and also a QC-MDPC Niederreiter variant [26] were recently published. They all demonstrated excellent efficiency in terms of computational complexity and key sizes for encryption and decryption on constrained platforms such as embedded micro-controllers and FPGAs using the proposed parameters.

A European initiative, PQCRYPTO, sponsored by the European Commission under its Horizon 2020 Program ICT-645622, is »developing cryptology that resists the unmatched power of quantum computers«. In September 2015 this group of researchers published a report entitled "Initial Recommendation of long-term secure post-quantum systems", where they recommended several algorithms as being ready in 2015 and several others that warrant further study and may be recommended in coming years. This report recommends the QC-MDPC scheme for further study, confirming its competitiveness as a post-quantum candidate.

4 A Key-Recovery Attack

In this section we describe our new attack against the plain QC-MDPC scheme as it has been proposed in [29] and described in the previous section.

[1] As in NTRUEncrypt [17,16], a secure approach is to require the decoding error probability to be less than $2^{-\kappa}$ for the κ-bit security.

4.1 Attack Model

The basic scenario for the attack is the following. Alice continuously sends messages to Bob using the QC-MDPC scheme and Bob's public key. Occasionally, a decoding error will occur and Bob will show a different reaction to report this decoding failure. The information will then be detected and collected. After repeating the procedure a number of times, Alice will be capable of recovering Bob's secret key using our proposed attack.

In terms of a security model definition, the attack is called a *reaction attack*. In previous work, resend and reaction attacks on McEliece PKC have appeared [14]. However, they have targeted message recovery only and there has been no key recovery attack in this model before.

The McEliece PKCs in their plain form have computational security against *chosen plaintext attacks* (CPAs), but are known to be insecure against *chosen ciphertext attacks* (CCAs). The reaction attack is an attack model in-between since it only requires the reaction of the decryption device (whether there was a decryption error) and not the result of decryption.

4.2 Attack Description

Continuing, we assume that the rate of the code is $R = k/n = 1/2$, corresponding to $n_0 = 2$. Also, let $w_0 = w_1 = w$. Attacks for other parameters follow in a similar fashion.

The key-recovery attack on QC-MDPC aims at finding the secret matrix $\mathbf{H}_0$, given only the public-key matrix $\mathbf{P}$. From $\mathbf{H}_0$, the remaining part of $\mathbf{H}$ can easily be recovered from $\mathbf{P}$ using basic linear algebra. Being a cyclic matrix, recovering $\mathbf{H}_0$ is equivalent to recovering its first row vector, denoted $\mathbf{h}_0$.

The key idea is to examine the decoding procedure for different error patterns. In particular, we will be interested in having Alice pick error patterns from special subsets. Let Ψ_d be the set of all binary vectors of length $n = 2r$ having exactly t ones, where all the t ones are placed as random pairs[2] with distance d in the first half of the vector. The second half of the vector is an all-0 vector. Formally, we select from the set Ψ_d, which guarantees repeated ones at distance d at least $t/2$ times, where

$$\Psi_d = \{\mathbf{v} = (\mathbf{e}, \mathbf{f}) \mid w_{\mathrm{H}}(\mathbf{f}) = 0, \text{ and } \exists \text{ distinct } s_1, s_2, \ldots, s_t, \text{ s.t. } \mathbf{e}_{s_i} = 1, \text{ and } s_{2i} = (s_{2i-1} + d) \bmod r \text{ for } i = 1, \ldots, \frac{t}{2}\}.$$

Alice will now send M messages to Bob, using QC-MDPC with the error selected from the subset Ψ_d of all possible error vectors of weight t. When there is a decoding error with Bob, she will record this and after M messages she will be able to compute an empirical decoding error probability for the subset Ψ_d. Furthermore she will do this for $d = 1, 2, \ldots, U$ for some suitable upper bound U.

[2] We assume that t is an even number for the ease of description; otherwise, we just pick $\frac{t-1}{2}$ random pairs and randomly choose another position to fulfill the constraint on the error weight.

Algorithm 1 – Computing the distance spectrum

Input: parameters n, r, w and t of the underlying QC-MDPC code, number of decoding trials M per distance.
Output: distance spectrum $D(\mathbf{h}_0)$.

for *all distances* d **do**
  Try M decoding trials using the designed error pattern
  Perform statistical test to decide multiplicity $\mu(d)$
  if $\mu(d) > 0$ **then**
    Add d with multiplicity $\mu(d)$ to distance spectrum $D(\mathbf{h}_0)$

The main observation of the paper is that there is a strong correlation between the decoding error probability for error vectors from Ψ_d and the existence of a distance d between two ones in the secret vector $\mathbf{h}_0$. Namely, if there exists two ones in $\mathbf{h}_0$ at distance d, the decoding error probability is much smaller than if distance d does not exist between two ones. We will give an explanation to this in the next section.

So after sending $M \times U$ messages, we look at the decoding error probability for each Ψ_d and classify each d, $d = 1, 2, \ldots, U$ as "does not exist in $\mathbf{h}_0$" (called CASE-0) or "existing in $\mathbf{h}_0$" (called CASE-1). This gives us what we call a distance spectrum for $\mathbf{h}_0$, denoted $D(\mathbf{h}_0)$. It is given as

$$D(\mathbf{h}_0) = \{d : 1 \leq d \leq U, d \text{ classified as existing in } \mathbf{h}_0\}.$$

Also, since a distance d can appear many times in the distance spectrum of a given bit pattern $\mathbf{c}$, we will let the multiplicity of d in $\mathbf{c}$ be denoted $\mu_{\mathbf{c}}(d)$ in the sequel, or simply $\mu(d)$ when $\mathbf{c}$ is clearly defined from the context.

As an example, for the bit pattern $\mathbf{c} = 0011001$ we have $U = 3$ and

$$D(\mathbf{c}) = \{1, 3\},$$

with distance multiplicities $\mu(1) = 1, \mu(2) = 0$ and $\mu(3) = 2$.

The procedure for computing the distance spectrum is specified in Algorithm 1.

The final step is to do a reconstruction of $\mathbf{h}_0$ from knowing the distance spectrum $D(\mathbf{h}_0)$. This is done through an iterative procedure. Start by assigning the first two ones in a length i_0 vector in position 0 and i_0, where i_0 is the smallest value in $D(\mathbf{h}_0)$. Then put the third one in a position and test if the two distances between this third one and the previous two ones both appear in the distance spectrum. If they do not, we test the next position for the third bit. If they do, we move to test the fourth bit and its distances to the previous three ones, etc. After reconstruction, we have restored $\mathbf{h}_0$. The reconstruction procedure is illustrated in Figure 1 and detailed in Algorithm 2.

Algorithm 2 – Key recovery from distance spectrum

Input: distance spectrum $D(\mathbf{h}_0)$, partial secret key $\mathbf{h}_0$, current depth l.
Output: recovered secret key $\mathbf{h}_0$ or message "No such secret key exists".
Initial recursion parameters: distance spectrum $D(\mathbf{h}_0)$, empty set for secret key, current depth 0.

```
if l = w then
    return h_0 /* secret key found */
for all potential key bits i do
    for all distances to key bit i exist in D(h_0) do
        Add key bit i to secret key h_0
        Make recursive call with parameters D(h_0), h_0 and l + 1
        if recursive call finds solution h_0 then
            if h_0 is the secret key then
                return h_0 /* secret key found */
        Remove key bit i from secret key h_0
return "No such secret key exists"
```

For the above example with bit pattern $\mathbf{c} = 0011001$, the careful reader will note that this algorithm will reconstruct $\mathbf{c}$ as 1100100 – with a rotation. However, this rotation is a non-issue in practice in our application.

In addition, the reconstruction procedure may also find some key pattern $\mathbf{h}'$ with the same distance spectrum $D(\mathbf{h}_0)$ as $\mathbf{h}_0$. The algorithm will then discard it and recursively try other key patterns, which provides an exhaustive search process.

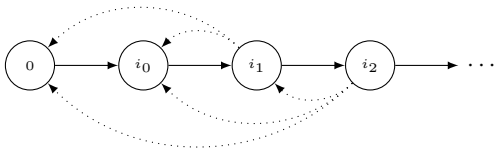

Fig. 1: The reconstruction process. Vertices represent nonzero bit positions in the bit pattern, solid arrows show the search order from left to right, dotted arrows show that for a newly determined bit position, its distances to all previous nonzero bit positions should all be in the distance spectrum.

5 Analysis

In this section we present an intuitive explanation why the proposed attack can recover the secret key from the decoding errors of a bit-flipping-type iterative decoder, and we also give some theoretical analysis on this new algorithm.

5.1 An Explanation for the Distinguishing Procedure

The authors in [27] pointed out that the employed iterative bit-flipping variants will stop in quite a small number of iterations (i.e., around 3 to 5 iterations on average), and further iterations have little effect on improving the success probability. Therefore, the behavior of the error variables in the first iteration plays a vital role in the decoding process: if almost all the variables flip from a wrong to a right value, the decoder will correct the errors quickly, otherwise it is more probable to fail.

We thus focus on the flipping behavior of the error bits in the first iteration for different input error patterns from the sets Ψ_d, containing random pairs of ones with distance d.

Table 2: The relation between the number of nonzero $h_{ij}e_i$'s and that of correctly changed counters in the first decoding iteration.

# $(h_{ij}e_i = 1)$	#(right change)	#(wrong change)
0	w	0
1	1	$w-1$
2	$w-2$	2
3	3	$w-3$
$\vdots$	$\vdots$	$\vdots$

First, we present more observations on the first round of the bit-flipping process. Given the j^{th} parity-check equation, i.e.,

$$\sum_{i=1}^{n} h_{ij}e_i = s_j,$$

for $1 \leq j \leq r$, this equation will affect w counters corresponding to the error variables with a nonzero coefficient h_{ij}. The value of the syndrome bit s_j determines if the equation is satisfied or not, since value for all the error variables e_i's are initially set to 0. That is, if $s_j = 0$, then the parity-check equation holds and no counters are increased for this check node. On the other hand, if $s_j = 1$, all the w counters for variable nodes included in this parity check are incremented.

Obviously, in the iterative decoding we do not want the counter for an error variable e_i to increment if $e_i = 0$, and vice versa; we do want it to increment

if $e_i = 1$. So we can consider whether the counter is correctly or erroneously changed.[3]

As a result, the number of nonzero terms of the form $h_{ij}e_i$'s in a parity-check equation determines the number of correctly changed counters in the first iteration, and the numbers are shown in Table 2. For example, in an equation, if there is no nonzero terms $h_{ij}e_i$, then $s_j = 0$ and the initial values of the e_i's in this check are all correct; But since $s_j = 0$ none of their counters are incremented and hence all counters are correctly changed.

If there is only one nonzero term $h_{ij}e_i$ in the parity check, then $s_j = 1$ and the equation is unsatisfied. Every counter corresponding to an error variable in this equation will be increased, but only one variable is actually in error. Hence we are changing $w - 1$ counters erroneously and only one correctly. For two nonzero term $h_{ij}e_i$ in the parity check, $s_j = 0$ and it follows in the same way as before that $w - 2$ counters are correctly changed and two of them erroneously, etc.

According to the above observation, it is desirable to have a small even number (like 0, 2, ...) of nonzero terms $h_{ij}e_i$ when evaluating parity-check equations for having the best chances of success in decoding. We can observe that if we look at all the r parity checks in $\mathbf{H}$, we will create a total of exactly $t \cdot w$ nonzero terms $h_{ij}e_i$ in the parity checks all together. For a randomly selected weight t error, we can view this as putting $t \cdot w$ different objects in r buckets and counting the number of objects in each bucket. An even number of objects in a bucket will be helpful in decoding, while an odd number of objects will act in opposite.

Now let us consider errors selected from our special error set Ψ_d. If the secret vector $\mathbf{h}_0$ contains two ones with distance d inbetween (CASE-1), then, due to the many inserted pairs of distance d in the error vector, we have "artificially" created a number of $(\geq \frac{t}{2})$ check equations where we know that we have at least two nonzero terms $h_{ij}e_i$ in the parity check. This "artificial" creation of pairs of nonzero terms $h_{ij}e_i$ in the same check equation changes the distribution of the number of nonzero terms $h_{ij}e_i$ in parity checks. If the secret vector $\mathbf{h}_0$ does not contains two ones with distance d inbetween (CASE-0), then the same phenomenon does not appear.

In Table 3, we present a precise evaluation of the corresponding distributions of an instance using the suggested QC-MDPC parameters for 80-bit security where the weight of $\mathbf{h}_0$ is assumed to be exactly 45. These results are obtained by a heavy simulation using 1000 different random keys and 480100 valid error patterns for each key. In CASE-1, the probability of being 0 is higher and that of being 1 lower, which are both preferred for the decoding purpose. Also owing to that the probabilities of being other values larger than 1 are of a similar magnitude for the both cases, this table verifies the influences of the "artificially" created pairs.

Since this algorithm iterates further and many quite short (e.g., length-4) cycles[4] appear in the corresponding Tanner graph, it is challenging to determine

[3] Here "change" means increasing by 1 or preserving the value.

[4] See Table 4 for more details.

Table 3: The distinct distributions of the number of nonzero terms $h_{ij}e_i$'s for the error patterns from Ψ_d using the QC-MDPC parameters for 80-bit security and assuming that the weight of $\mathbf{h}_0$ is exactly 45.

$\#\ (h_{ij}e_i = 1)$	Probability	
	CASE-0	CASE-1
0	0.4485	0.4534
1	0.3663	0.3602
≥ 2	0.1852	0.1864

the variation of the decoding error probability caused by the different distributions in the first round, via presenting some precise theoretical estimations. On the other hand, several thousands of parity-check equations (e.g., 4801 equations in the 80-bit security case) exist, making the overall differences substantial. In addition, more correct values in the initial round will contribute positively in the following iterations. These facts explain why some significant differences can be detected in our experiments, and why they imply a successful key-recovery attack in real time.

5.2 Complexity Analysis

We now derive a complexity estimation for the key-recovery attacks. Making use of the obtained experimental results for certain key parameters, we can then approximate the concrete time complexity (shown in Section 7). This complexity consists of two parts: that of building the distance spectrum and that of reconstructing the secret polynomial. We analyze separately.

The complexity for building the distance spectrum It is shown in experiments that the error rates for the different distances clearly separate into intervals according to multiplicity. When these intervals are disjoint, it is possible to determine the complete distance spectrum of the secret key fully and without error. In general, for a well-designed error pattern, the error probabilities increase with decreasing multiplicities, as sketched in Figure 2a.

The distinguishing procedure involves U groups of decoding tests, each of them consisting of M decoding trails. Thus, overall $U \times M$ decoding data would be collected, implying that the complexity is of order $\mathcal{O}(MU)$. Here M and U are two algorithmic parameters that depend on the targeting security parameters $n, r, \hat{w}, t$. A reasonable upper bound for U is $\lfloor \frac{r}{2} \rfloor$, since this is the number of possible (modular) distances given the block size r.

On the other hand, it is non-trivial to determine the minimal value of M that is sufficient to execute a successful distinguishing. The experimental results suggest that the error rate for the error pattern using a distance d with multiplicity $\mu(d)$ can be approximated by a Gaussian distribution with mean $m_{\mu(d)}$ and

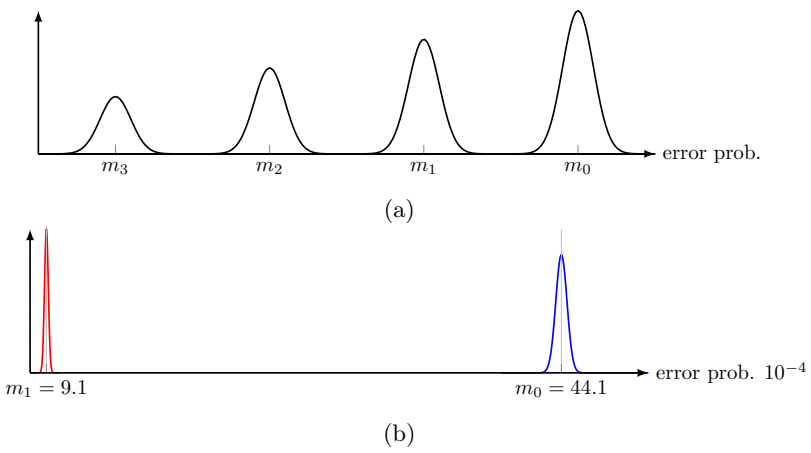

Fig. 2: Classification of distance multiplicities based on decoding error probability. (a): Distribution shape in general. (b): Empirical distribution using $M = 100,000$ decoding trials for each distance (proposed parameters for 80-bit security with $t = 84$).

variance $\sigma^2_{\mu(d)}$; we can thus model this problem as a hypothesis testing problem determining whether $\mu(d)$ is zero or not.

Figure 2, which consists of two sub-figures, describes the approximated distributions of the error probability when performing the proposed reaction attack. With adequate decoding trials to make the widths of these Gaussian distributions "narrow" enough, we draw roughly the shape of the probability density function of the decoding error probability (in Figure 2a). On the contrary, Figure 2b records with the precision in magnitude the empirical distribution when performing the proposed reaction attack on the QC-MDPC parameters for 80-bit security with error weight 84, where $100,000$ decoding trails are exploited for each distance d. In this figure, only the groups of multiplicity 0 and multiplicity 1 are depicted as the remaining groups are of a much smaller magnitude. More data can be found in Table 5.

The complexity for reconstruction We show that the algorithm will return the correct key soon on average. Since this algorithm builds an enumeration tree to search for the possible solutions in a depth-first way, the time complexity can be represented by its paths to the leaves in the tree. Later we present a rough estimation of this number.

Suppose n_s is the size of the distance spectrum $D(\mathbf{h}_0)$, n_t the number of possible distances[5] required to be tested, and α the ratio between them, i.e. n_s/n_t. In the beginning, we chose the smallest distance in the spectrum and

[5] A reasonable setting for n_t is T, the number of distances bounded by $\frac{r}{2}$.

determine two positions 0 and i_0; this can be viewed as the root of the tree. Then we extend the tree to choose another position i_1. We know that the distances i_1 and $i_1 - i_0$ should be both in the distance spectrum; among the n_s possible distances for i_1, thus, we can expect an α fraction of them are valid and there exist $n_s\alpha$ nodes in the first level. Similarly for one node in the first level, there are $n_s\alpha^2$ child nodes on average in the second level since the distances i_2, $i_2 - i_0$ and $i_2 - i_1$ should be all in the distance spectrum. Etc.

Since the average child number of a node after quite few steps[6] (denoting this number $\phi + 1$) will be less than 1, we can deduce a loose estimation on the average number of possible paths as

$$\prod_{i=1}^{\phi} n_s\alpha^i = n_t^{\phi}\alpha^{\frac{\phi(\phi+3)}{2}} \leq \left(\frac{r}{2}\right)^{\phi}\alpha^{\frac{\phi(\phi+3)}{2}}. \tag{1}$$

The above results state that in expectation, the number of paths tested can be bounded by Equation (1). In reality, the algorithm may terminate soon if we are lucky.

6 Debunking the CCA Security Claim

When targeting the CPA security of the MDPC scheme, we were free to choose the injected error patterns. When we now turn to attack its CCA-secure version, this freedom of choice is severely limited.

The CCA-secure version of the MDPC scheme is of more importance as in real applications the error vector will be protected by cryptographic hash functions after conversions (e.g., [21]) for making the MDPC scheme semantically secure.

The fundamental idea of the attack is as follows. We randomly generate T plaintext-ciphertext pairs. We then form subsets of those with desired error patterns. In particular, we will be interested in error patterns that contain occurrences of distance d between error bits, where d is a length in the distance spectrum to be tested. Our simulations show that these error patterns can be used to efficiently distinguish whether a certain distance d appears in the distance spectrum of the targeted secret polynomial.

We present the algorithm in two versions to match different levels of detail. The high level description is presented as Algorithm 3.

The description seems to suggest that we need lots of storage for handling ciphertexts, but this is not the case. An efficient implementation requires virtually no storage. To see this, consider the alternative description in Algorithm 4.

In Algorithm 4 we successively check the decryptability of ciphertexts and use these observations to obtain better and better estimates of decoding error probabilities related to all possible distances in the distance spectrum of the secret key.

The vector slots of $\mathbf{a}$ and $\mathbf{b}$ are used to represent the decoding error probabilities, so that $\frac{\mathbf{a}[d]}{\mathbf{b}[d]}$ is an approximation of the decoding error probability over

[6] The average child number of a node in the l^{th} level drops exponentially in l.

Algorithm 3 – Breaking the CCA security of the converted MDPC scheme.

Input: number T of ciphertexts to generate.
Output: distance spectrum $\mathbf{s}$ for the secret key K.

```
Generate a collection Σ of T ciphertexts
Record decryptability for each c in Σ
s ← storage for distance spectrum of secret key
for all distances d do
    Σ_d ← {c ∈ Σ | μ_c(d) ≥ 1}
    s[d] ← multiplicity classification from decryptability rate in Σ_d
return s
```

all error patterns with distance spectrums containing distance d. This subset of error patterns is denoted Σ_d in Algorithm 3.

Each ciphertext updates several entries in $\mathbf{a}$ and $\mathbf{b}$, and we need to observe the decryptability of sufficiently many ciphertexts in order to obtain probability estimates that are reliable enough for correct multiplicity classification.

For each ciphertext we utilize the nonzero (other thresholds are also possible) entries in the corresponding distance spectrum. Letting α denote the average fraction of nonzero entries in such a distance spectrum, one can see that the total number of iterations in the inner loop (per ciphertext) is about $\frac{\alpha r}{2}$.

The output of Algorithm 4 is the distance spectrum of the secret key, so the careful reader will note that the key recovery method described in Algorithm 2 needs to be applied as a final step for full key recovery. However, in terms of complexities, this additional step comes for free.

The time complexity of Algorithm 4 is precisely T if we count the number of observed ciphertexts. If we count low-level operations, as defined by the inner loop of Algorithm 4, the time complexity is $T \times \frac{r}{2}$.

6.1 An Explanation of How Sample Collection Works

The precise nature of Algorithm 4 can easily and very conveniently be understood by modeling the sampling procedure as a generalized version of the coupon collector's problem. In the original coupon collector's problem, using the balls-and-bins paradigm, we randomly throw balls into u bins until all bins are nonempty. We need to throw around $u \log u$ balls before we achieve this goal.

In the generalized problem, we keep throwing balls until all bins each contain at least b balls. The time complexity for this (see [11]) is

$$J(u,b) = u\left(\log u + (b-1)\log\log u + \gamma - \log(b-1)! + o(1)\right). \tag{2}$$

It is even possible to arbitrarily bound the probability of failure by adding a linear number of samples (balls) according to

$$\lim_{t\to\infty} \mathbf{Pr}\left[\mathcal{X}_{(u,b)} < u\log u + (b-1)\,u\log\log u + tu\right] = e^{-\frac{e^{-t}}{(b-1)!}},$$

Algorithm 4 – Breaking the CCA security of the converted MDPC scheme. Detailed description.

Input: number T of ciphertexts to generate.
Output: distance spectrum for the secret key K.

```
a ← zero-initialized vector of length r/2 /* count decoding failures per distance */
b ← zero-initialized vector of length r/2 /* count total samples per distance */
i ← 0
while i < T do
    Generate ciphertext c
    s_err ← distance spectrum of ciphertext error
    ℓ ← decryptability of c /* 0 for successful decryption, 1 for decryption failure */
    for all distances d do
        if s_err[d] ≥ 1 then
            a[d] ← a[d] + ℓ
            b[d] ← b[d] + 1
    i ← i + 1
s_key ← vector of length r/2 /* distance spectrum of secret key */
for all distances d do
    s_key[d] ← multiplicity classification from estimated error rate a[d]/b[d]
return s_key
```

where $\mathcal{X}_{(u,b)}$ is a statistical variable that represents the number of throws needed to fill up u bins so that all of them contain at least b balls.

In the CCA case we collect error patterns, but not all error patterns are useful. Instead, we form different subsets of useful error patterns denoted Σ_d in Algorithm 3. We successively check the decryptability of ciphertexts and use these observations to obtain better and better estimates of decoding error probabilities related to all possible distances in the distance spectrum of the secret key.

For each ciphertext we then utilize the nonzero (other thresholds are also possible) entries in the corresponding distance spectrum, and each such nonzero entry corresponds to a ball. With α denoting the average fraction of nonzero entries in such a distance spectrum, one can see that the total number of balls we collect per ciphertext is about $\frac{\alpha r}{2}$.

In Algorithm 4, the bins are represented by the vector slots of $\mathbf{a}$ and $\mathbf{b}$, so there are $u = \frac{r}{2}$ bins. Each observed error pattern generates α balls, and each ball updates an entry in $\mathbf{a}$ and $\mathbf{b}$. We need at least b balls in each bin in order to obtain probability estimates that are reliable enough for computing the distance spectrum of the secret key. The value b determines the number T of ciphertexts that we need to generate, since b and T are strongly related according to

$$\frac{\alpha r T}{2} \approx J\left(\frac{r}{2}, b\right). \tag{3}$$

It may also be noted that it is not immediately clear how to analytically derive b or T directly from the security parameters. For our results, we have determined T explicitly by simulation, as described in Section 7.

7 Implementations and Numerical Results

We have conducted several simulation tests to verify the behaviors of the error rates related to different multiplicities and different error shapes. The following implementation results are all obtained by employing QC-MDPC with the proposed parameters for 80-bit security [29] and the original Gallager's bit-flipping algorithm [12], i.e., Decoder $\mathcal{B}$ in [27].

In the CPA case, we consider two different error weights. Error weight $t = 84$ is what is proposed for 80-bit security, but we also consider the case $t = 90$ here. This is motivated by security models that allow injection of more errors, where additional errors are not explicitly detected. For the CCA case, only results with $t = 84$ are stated.

Results for the CPA case are presented in Section 7.1, and the results for the CCA case are presented in Section 7.2. A discussion on the employment of other decoders follows in Section 7.3.

Before introducing the main implementation results, we show the probability distributions for distance multiplicities in the first polynomial when considering the QC-MDPC scheme with $n_0 = 2$ (see Table 4).

Table 4: Probability distributions for distance multiplicities in the first polynomial (of two), generated uniformly with total weight $t = 84$ and $t = 90$. The polynomial length is 4801, while the total vector length is 9602.

	$t = 84$			$t = 90$ / key with $\hat{w} = 90$		
multiplicity	probability	accumulated	accumulated	probability	accumulated	accumulated
0	0.6955724	0.6955724	1.0000000	0.6589889	0.6589889	1.0000000
1	0.2524958	0.9480683	0.3044275	0.2748075	0.9337965	0.3410106
2	0.0458487	0.9939170	0.0519316	0.0573330	0.9911295	0.0662031
3	0.0055425	0.9994596	0.0060829	0.0079677	0.9990972	0.0088701
4	0.0005018	0.9999614	0.0005403	0.0008287	0.9999260	0.0009024
5	0.0000362	0.9999977	0.0000385	0.0000688	0.9999949	0.0000737
6	0.0000021	0.9999998	0.0000022	0.0000047	0.9999997	0.0000049
7	0.0000001	1.0000000	0.0000001	0.0000002	1.0000000	0.0000002

The vector is of length 9602, and is generated uniformly with weight 84 (or 90). These probability distributions are mainly of importance for the following two reasons.

– When the vector is viewed as a key vector, the data in the right part (corresponding to $t = 90$) show the distance multiplicity distributions of a random key, from which not only the size of its distance spectrum can be estimated,

Table 5: Decoding error rates when using the original Gallager's bit-flipping algorithm (Decoder $\mathcal{B}$ in [27]) and the designed error pattern Ψ_d with $t = 84$ and $t = 90$. The number of decoding trials in a group is $M = 100,000$ and $M = 10,000$, respectively.

	$t = 84$		$t = 90$	
multiplicity	error rate	σ	error rate	σ
0	0.0044099	0.00003868	0.415395	0.000830
1	0.0009116	0.00001304	0.248642	0.000729
2	0.0001418	0.00000475	0.121623	0.000529
3	0.0000134	0.00000112	0.048330	0.000299

but some other vital information may also be revealed. For example, since about 6.6 percent of the distances are of multiplicity 2 or more when $t = 90$, quite a few length-4 cycles[7] will appear in the Tanner graph corresponding to the secret key.

– When the vector is viewed as an error vector, these data can be utilized to simulate the random error obtained from a CCA2-secure QC-MDPC scheme. We will explain this further in Section 7.2.

7.1 CPA Case

As described in Section 5.2, the time complexity of attacking the CPA-secure version consists of two parts: that of constructing the distance spectrum and of key reconstruction. From Table 5, we can see that for the MDPC parameters targeting 80-bit security, it is sufficient to choose M to be $100,000$ to make the decoding error rates well-separated according to the multiplicity; this value can be even reduce to $10,000$ if an error with weight $t = 90$ is allowed to be used. Setting the number of different groups for decoding test as 2400, we derive that the time complexity for Alice to know the distance spectrum of the secret key is bounded by that of calling the decoder about $240,000,000$ (or $24,000,000$) times for solely the information whether the decoding succeeds, when the error weight t is 84 (or 90). In the security model of a reaction attack, the decoding results (success or fail) are presumably provided to the adversary; therefore, the decoding cost is excluded from the time complexity, implying that the time complexity for constructing the distance spectrum can be estimated as 2^{28} (or 2^{25}) operations for $t = 84$ (or 90).

For the MDPC parameters targeting 80-bit security, the weight of the secret key is set to be 90. By checking Table 4, therefore, the empirical ratio α can be approximated as 0.341 and thus ϕ is 6. We on average test no more than $2^{25.5}$ paths, costing less than 2^{35} operations since most of the invalid paths will be detected and removed soon (less than 20 steps). We implemented this algorithm,

[7] A distance with multiplicity of 2 or more implies that there exists at least one length-4 cycle.

which performed quite well in practice — for most of the instances, the algorithm succeeded in minutes.

7.2 CCA Case

Next in turn is the CCA case and truly uniform error patterns with a certain weight. We have used Algorithm 4 for our simulation runs. One such simulation for the QC-MDPC scheme with the proposed parameters for 80-bit security (with $t = 84$) can be seen in Figure 3. Here we plot the number of utilized ciphertexts vs. the fraction of correctly classified distance spectrum entries, resulting in a simple visualization of the algorithm efficiency.

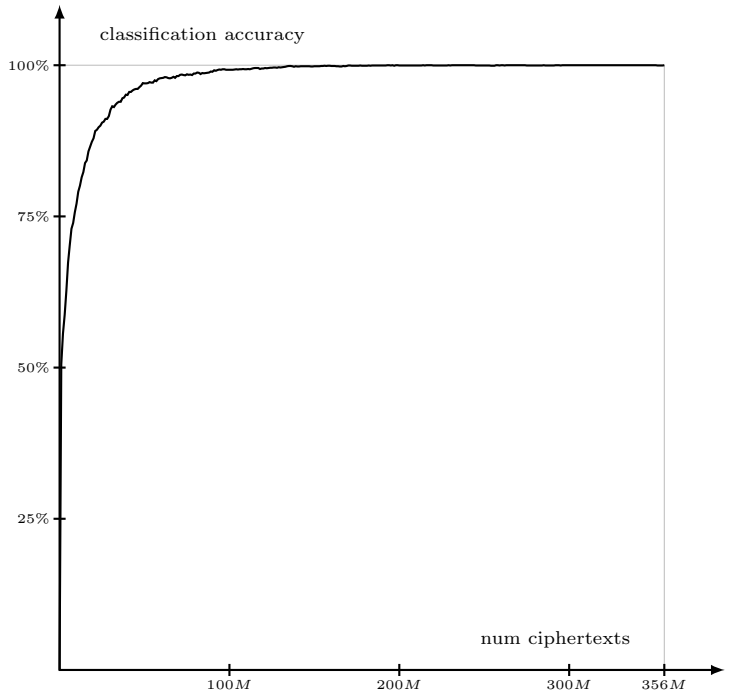

Fig. 3: CCA algorithm for a QC-MDPC McEliece instance for 80-bit security with $t = 84$. The distance spectrum of the key is fully recovered (no errors) after observing 356M ciphertexts. The graph shows the worst case out of ten full simulations.

The simulations suggest that $T = 356$ million observed ciphertexts are sufficient for fully determining the entire distance spectrum without any errors. That is, after we have observed 356 million ciphertexts, the distance spectrum remains stable and correct.

It should be noted that we ran ten independent simulation runs, and the result presented in Figure 3 was the worst case simulation result. The 356 million ciphertexts estimate is therefore a conservative high probability estimate. That is, in all simulations, the multiplicity classifications were 100% correct and stable after 356 million ciphertexts. For comparison, the best case yielded full distance spectrum recovery after 203 million ciphertexts.

The same simulation is shown in Figure 4, providing a more detailed view of how the algorithm works. Each dot represents the estimated decoding error probability for one particular distance, and every possible distance has been plotted in the same graph.

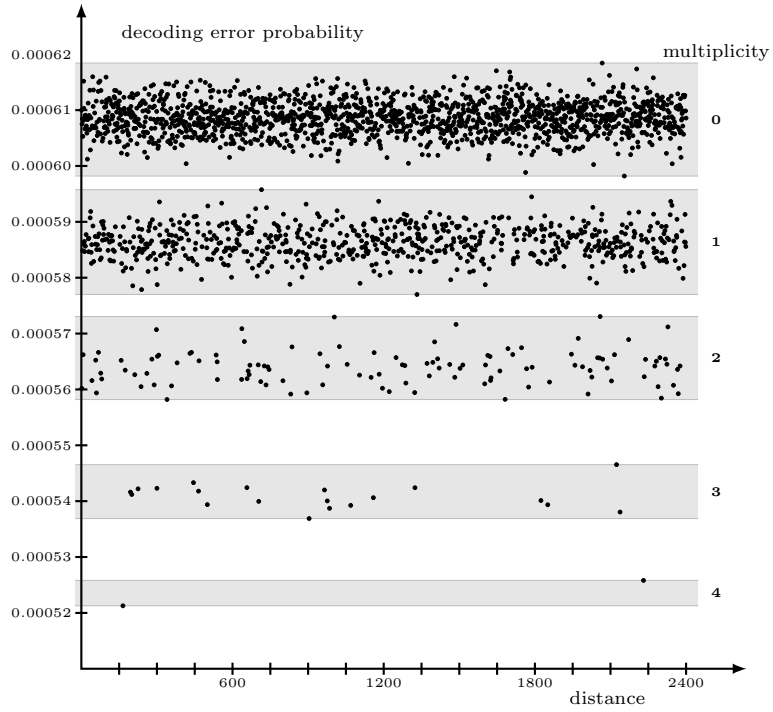

Fig. 4: Classification intervals for the $t = 84$ worst-case simulation after 356M ciphertexts. All 2400 data points plotted.

Now, have a closer look at the classification intervals that have been highlighted with a grey background. These intervals span from the minimum to the maximum estimated decoding error probability *per multiplicity*.

As Figure 4 shows the state of affairs at the end of the simulation, we can see that the distance dots are clearly separated into different and fully disjoint classification intervals depending on their multiplicity. The classification intervals are generally overlapping during the earlier parts of the simulation.

We can use the classification intervals and analyze their successive widths in the simulation. This is very useful, because we are only able to compute a perfect error-free distance spectrum when all of these intervals are mutually disjoint. In this way we can derive a reliable estimate for the value T for different problem instances. The reader may note that there is some (little) room for improvement here. In the worst case simulation out of the ten we have performed, the classification intervals became disjoint after 268 million ciphertexts, which sets a lower bound for error-free distance spectrum recovery. However, this bound can be lowered even further if we allow errors in the recovered distance spectrum, or if we tweak the algorithm in other ways, and so on, but such improvements are out of scope here.

In a live scenario, the distance multiplicities are unknown, so it is not possible to compute the successive classification intervals and check when they are disjoint. Predetermined probability values cannot be used, since the decoding error probabilities differ significantly between instances (different keys). However, the general shape of the probability distribution in Figure 2a is known and represents a "side view" of Figure 4, so it can be seen that the multiplicity classification problem is not very difficult in practice.

The classification procedure we have used in our experiments is quite simplistic. In our simulations, we computed the current (estimated) decoding error probabilities $m_0, m_1, \ldots$ per multiplicity from the simulation values, and then computed boundary mid-points $\frac{m_0+m_1}{2}, \frac{m_1+m_2}{2}, \ldots$ and checked when the intervals were fully separated into these mid-point regions. In a live scenario one could efficiently achieve the same effect by using a simple clustering technique (counting dots in small intervals) to first accurately estimate $m_0, m_1, \ldots$, and then continue as we have done.

To conclude the simulation results, the total time complexity of Algorithm 4 is at most 356 million observed ciphertexts. If we count low-level operations instead, as specified in Section 6, then the total time complexity is about $T \times \frac{r}{2} = 2^{39.7}$ for the proposed security parameters for 80-bit security using the Gallager's original bit-flipping decoder. Compared with this complexity figure, the key reconstruction part is negligible.

7.3 Some Discussions

We discuss more about the decoding procedure employed in the implementation.

Using other decoding techniques The employed decoding algorithm in implementation is Gallager's bit-flipping algorithm, which is chosen not only because its relatively higher error-probability makes the implementation easier, but also because it is the original iterative decoding algorithm for LDPC settling the framework and the principle for the later improved decoders [15,27]. Hence, it is reasonable to assume that replacing the Gallager's bit-flipping decoder by another more advanced decoder may increase the attack complexity by a factor of around 2^e, if the error probability is reduced with a factor of 2^{-e}. However, the attack complexity is still far less than the claimed security level.

For example, the best implementations of bit-flipping-type decoders found in literature with respect to the decoding performance are the ones from [10] and [8] both claiming a decoding error probability less than 10^{-8} for the 80-bit secure QC-MDPC parameter set. These decoders improve upon the original Gallager's decoder by a factor of about $2^{15.6}$. Therefore, we might estimate the time complexity for attacking the 80-bit CPA (or CCA2) -secure version as $2^{43.6}$ (or $2^{55.3}$) operations, if these two decoders are instead implemented.

Moreover, some decoders (including the one in [29]) decrease the error probability by restarting the decoding process in the same decoding framework but only employing different thresholds, when an error is detected. On one side, since each calling of the bit-flipping algorithm might contribute to the variation between CASE-0 and CASE-1, the effects on the distinct decoding error probabilities may accumulate after more and more decoding rounds, implying that the estimation in the last paragraph is conservative. On the other hand, via some side-channel attacks, an adversary might get the information of the initial errors occurred, thereby reducing the problem to that of using a less powerful decoder like the one being implemented. It is definitely beneficial to design a countermeasure to withstand this type of attack.

We conjecture that this attack also works for the MDPC scheme employing a soft-decision decoding implementation.

Moving to a higher security level The QC-MDPC scheme using the suggested 80-bit secure parameter set is frequently discussed and implemented in literature due to its applications in power-constraint devices, for which we choose it as a study case. However, for the long-term security purpose, the bottom line nowadays is to achieve 128-bit security. There is no evidence that the scheme with the suggested 128-bit secure parameters will be invulnerable to the proposed reaction attack, and frankly speaking, the situation is even worse due to the larger gap between the current state-of-the-art implementation[8] of bit-flipping-type decoders and the required decoding performance[9] with respect to security.

Higher error probability Another meaningful observation is that when utilizing the designed highly unbalanced error pattern, the error probability is higher than when harnessing a uniform distribution in the valid set of errors. This increased error probability jeopardizes the security of the MDPC scheme by boosting the proposed reaction attack further. Since we employ the same implementation of Gallager's bit-flipping decoder as in [15], the enlarged failure probability is mainly due to the specific error pattern used: all the t error po-

[8] With respect to the decoding performance, the best known implementation using the suggested 80-bit secure parameter set outperforms the one using the suggested 128-bit secure parameters (10^{-8} in [8,10] vs. 10^{-7} in [29]).

[9] One should decrease the decoding error probability for thwarting the proposed reaction attack within 2^{128} operations.

sitions are gathered together in the first part of the error vector. We show the numerical results in Table 6.

Table 6: The comparison of failure rates among different error patterns using the QC-MDPC parameters for the 80-bit security.

Error weight	all valid errors [15]	this work	
		multiplicity 0	multiplicity 1
84	0.00051	0.00441	0.00091
90	0.24080	0.41539	0.24864

8 Conclusions and Future Work

In this paper, we have presented a reaction-type attack against the QC-MDPC public key encryption scheme. This novel attack exploits the strong correlation between certain structures in the secret key and the decoding error probability when errors with arranged patterns are employed. It then rebuilds the secret polynomial efficiently by executing a reconstruction procedure, therefore breaking the QC-MDPC scheme. With a slight modification, it can also be applied to the CCA2 converted version of the scheme to break its claimed security level against CCA2 attack. This (weaker) reaction attack can break the proved (stronger) CCA2 security because the decoding error probability is excluded in the proof models.

There are several research directions to be further investigated. A natural one is to design a countermeasure to protect the QC-MDPC scheme against this new attack. The most secure approach is to amend the employed iterative decoder to reduce the decoding error probability to be less than $2^{-\kappa}$ for κ-bit security. This is a challenging task due to the large gap between the state-of-the-art and the desired error levels, and also due to the lack of a precise theoretical error bound on these iterative algorithms. That is, changing to a more powerful decoder can enhance its security, but it is doubtful to claim that it can reach a quite high security level. Moreover, when moving to a decoder with improved performance via running itself more times with different algorithmic parameters if an error is detected, some types of side-channel attacks — like timing attacks — should be useful to know the original errors in the initial round, which can be used for a faster reaction attack.

Other directions include characterizing the strong and weak keys to resist this attack, deriving precise bounds on the decoding error probability for various error patterns given a security parameter, and designing more advanced reconstruction algorithms to handle more errors in the distance spectrum, e.t.c.. It would be

fascinating if one can extend this attack to break the CCA-secure version of cryptosystems based on the LPN and LWE problems.

References

1. Augot, D., Batina, L., Bernstein, D.J., Bos, J., Buchmann, J., Castryck, W., Dunkelman, O., Güneysu, T., Gueron, S., Hülsing, A., et al.: Initial recommendations of long-term secure post-quantum systems. Available at `http://pqcrypto.eu.org/docs/initial-recommendations.pdf` (2015)
2. Baldi, M., Chiaraluce, F., Garello, R., Mininni, F.: Quasi-Cyclic Low-Density Parity-Check Codes in the McEliece Cryptosystem. In: Proceedings of IEEE International Conference on Communications, ICC 2007, Glasgow, Scotland, 24-28 June 2007. pp. 951–956. IEEE (2007), `http://dx.doi.org/10.1109/ICC.2007.161`
3. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{\frac{n}{20}}$: How $1 + 1 = 0$ improves information set decoding. In: Advances in Cryptology—EUROCRYPT'12, pp. 520–536. Springer (2012)
4. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Advances in Cryptology—CRYPTO'98. pp. 26–45. Springer (1998)
5. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post-quantum cryptography. Springer Science & Business Media (2009)
6. Berson, T.A.: Failure of the McEliece public-key cryptosystem under message-resend and related-message attack. In: Advances in Cryptology—CRYPTO'97, pp. 213–220. Springer (1997)
7. Canteaut, A., Sendrier, N.: Cryptanalysis of the original McEliece cryptosystem. In: Advances in Cryptology—ASIACRYPT'98. pp. 187–199. Springer (1998)
8. Chaulet, J., Sendrier, N.: Worst case QC-MDPC decoder for McEliece cryptosystem. In: IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016. pp. 1366–1370. IEEE (2016), `http://dx.doi.org/10.1109/ISIT.2016.7541522`
9. Chen, L., Jordan, S., Liu, Y.K., Moody, D., Peralta, R., Perlner, R., Smith-Tone, D.: Report on post-quantum cryptography. National Institute of Standards and Technology Internal Report 8105 (2016)
10. Chou, T.: QcBits: Constant-Time Small-Key Code-Based Cryptography. In: Gierlichs, B., Poschmann, A.Y. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2016: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings. pp. 280–300. Springer Berlin Heidelberg, Berlin, Heidelberg (2016), `http://dx.doi.org/10.1007/978-3-662-53140-2_14`
11. Flajolet, P., Sedgewick, R.: Analytic Combinatorics. Cambridge University Press (2009)
12. Gallager, R.G.: Low-Density Parity-Check Codes. Ph.D. thesis, MIT Press, Cambridge (1963)
13. Goppa, V.D.: A New Class of Linear Correcting Codes. In: Probl. Peredachi Inf. vol. 6, pp. 24–30) (1970)
14. Hall, C., Goldberg, I., Schneier, B.: Reaction attacks against several public-key cryptosystem. In: Information and Communication Security, pp. 2–12. Springer (1999)
15. Heyse, S., Von Maurich, I., Güneysu, T.: Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices. In: Cryptographic Hardware and Embedded Systems-CHES 2013, pp. 273–292. Springer (2013)

16. Hoffstein, J., Pipher, J., Schanck, J.M., Silverman, J.H., Whyte, W., Zhang, Z.: Choosing Parameters for NTRUEncrypt. Cryptology ePrint Archive, Report 2015/708 (2015), http://eprint.iacr.org/
17. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Algorithmic number theory, pp. 267–288. Springer (1998)
18. Howgrave-Graham, N., Nguyen, P.Q., Pointcheval, D., Proos, J., Silverman, J.H., Singer, A., Whyte, W.: The impact of decryption failures on the security of NTRU encryption. In: Advances in Cryptology—CRYPTO'03, pp. 226–246. Springer (2003)
19. Howgrave-Graham, N., Silverman, J.H., Singer, A., Whyte, W., NTRU Cryptosystems: NAEP: Provable Security in the Presence of Decryption Failures. IACR Cryptology ePrint Archive 2003, 172 (2003)
20. Johansson, T., Jönsson, F.: On the complexity of some cryptographic problems based on the general decoding problem. IEEE Transactions on Information Theory 48(10), 2669–2678 (2002)
21. Kobara, K., Imai, H.: Semantically secure McEliece public-key cryptosystems-conversions for McEliece PKC. In: Public Key Cryptography. pp. 19–35. Springer (2001)
22. Löndahl, C., Johansson, T.: A New Version of McEliece PKC Based on Convolutional Codes. In: Chim, T.W., Yuen, T.H. (eds.) Information and Communications Security - 14th International Conference, ICICS 2012, Hong Kong, China, October 29-31, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7618, pp. 461–470. Springer (2012), http://dx.doi.org/10.1007/978-3-642-34129-8_45
23. MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error Correcting Codes, vol. 16. Elsevier (1977)
24. von Maurich, I., Güneysu, T.: Lightweight code-based cryptography: QC-MDPC McEliece encryption on reconfigurable devices. In: Proceedings of the conference on Design, Automation & Test in Europe. p. 38. European Design and Automation Association (2014)
25. von Maurich, I., Güneysu, T.: Towards side-channel resistant implementations of QC-MDPC McEliece encryption on constrained devices. In: Post-Quantum Cryptography, pp. 266–282. Springer (2014)
26. von Maurich, I., Heberle, L., Güneysu, T.: IND-CCA Secure Hybrid Encryption from QC-MDPC Niederreiter. In: Post-Quantum Cryptography, pp. 1–17. Springer (2016)
27. Maurich, I.V., Oder, T., Güneysu, T.: Implementing QC-MDPC McEliece Encryption. ACM Transactions on Embedded Computing Systems (TECS) 14(3), 44 (2015)
28. McEliece, R.J.: A Public-Key Cryptosystem Based On Algebraic Coding Theory. DSN Progress Report 42–44 pp. 114–116 (1978)
29. Misoczki, R., Tillich, J.P., Sendrier, N., Barreto, P.S.: MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In: Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on. pp. 2069–2073. IEEE (2013)
30. Overbeck, R., Sendrier, N.: Code-based cryptography. In: Post-quantum cryptography, pp. 95–145. Springer (2009)
31. Repka, M., Zajac, P.: Overview of the Mceliece Cryptosystem and its Security. Tatra Mountains Mathematical Publications 60(1), 57–83 (2014)
32. Sendrier, N.: Decoding one out of many. In: Post-quantum cryptography, pp. 51–67. Springer (2011)

33. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science, 20-22 November 1994, Santa Fe, New Mexico, USA. pp. 124–134. IEEE Press (1994)