




A Timing Attack on the HQC Encryption Scheme

Thales Bandiera Paiva^(✉)  and Routo Terada

Universidade de São Paulo, São Paulo, Brazil
{tpaiva,rt}@ime.usp.br

Abstract. The HQC public-key encryption scheme is a promising code-based submission to NIST's post-quantum cryptography standardization process. The scheme is based on the decisional decoding problem for random quasi-cyclic codes. One problem of the HQC's reference implementation submitted to NIST in the first round of the standardization process is that the decryption operation is not constant-time. In particular, the decryption time depends on the number of errors decoded by a BCH decoder. We use this to present the first timing attack against HQC. The attack is practical, requiring the attacker to record the decryption time of around 400 million ciphertexts for a set of HQC parameters corresponding to 128 bits of security. This makes the use of constant-time decoders mandatory for the scheme to be considered secure.

Keywords: HQC · Post-quantum cryptography · Timing attack · BCH decoding

1 Introduction

Hamming Quasi-Cyclic (HQC) [18] is a code-based public-key encryption scheme. It is based on the hardness of the quasi-cyclic syndrome decoding problem, a conjectured hard problem from Coding Theory. It offers reasonably good parameters, with better key sizes than the classical McEliece scheme [2, 5, 17], but without relying on codes with a secret sparse structure, such as QC-MDPC [19] and QC-LDPC [3].

One of the most interesting features HQC provides is a detailed analysis of the decryption failure probability, which makes it possible to choose parameters that provably avoid reaction attacks [9, 12] that compromise the security of QC-LDPC and QC-MDPC encryption schemes. This makes it one of the most promising code-based candidates in NIST's Post-Quantum standardization process. However, the negligible probability of decoding failure comes at the expense of low encryption rates.

The scheme uses an error correction code \mathcal{C} as a public parameter. The secret key is a sparse vector, while the public key is its syndrome with respect to a

T. B. Paiva is supported by CAPES. R. Terada is supported by CNPq grant number 442014/2014-7.

systematic quasi-cyclic matrix chosen at random, together with the description of this matrix. To encrypt a message, the sender first encodes it with respect to the public code \mathcal{C} , then adds to it a binary error vector which appears to be random for anyone who is not the intended receiver. The receiver, using the sparseness of her secret key, is able transform the ciphertext in such a way to significantly reduce the weight of the error vector. Then, the receiver can use the efficient decoding procedure for \mathcal{C} to correct the remaining errors of the transformed ciphertext to recover the message.

The code \mathcal{C} proposed by Aguilar-Melchor et al. [18] is a tensor code between a BCH code and a repetition code. One drawback of the HQC implementation submitted to NIST is that the decoder [14] for the BCH code is not constant-time, and depends on the weight of the error it corrects. This makes the decryption operation vulnerable to timing attacks.

The use of non-constant-time decoders has been exploited to attack code-based schemes such as QC-MDPC [8], and recently, RQC [1], which is a variant of HQC in the rank metric that uses Gabidulin codes [10], was shown vulnerable to timing attacks [6]. However, timing attacks exploiting non-constant-time decoders are not exclusive to code-base schemes, and the use of BCH codes in LAC [16] has been shown to leak secret information from timing [7].

Contributions. We present the first timing attack on HQC. The attack follows Guo et al. [12] idea: first we show how to obtain information, which is called the spectrum, on the secret key by timing a large number of decryptions, and then use the information gathered to reconstruct the key. We analyze in detail the reason behind the information leakage. As a minor contribution, we show that a randomized variant of Guo's et al. algorithm for key reconstruction is better than their recursive algorithm when the attacker has partial information on the secret key's spectrum. This is useful to reduce the number of decryption timings the attacker needs to perform.

Shortly after this paper was accepted for publication, Wafo-Tapa et al. [24] published a preprint in the Cryptology ePrint Archive in which they also present a timing attack against HQC. Our attack is stronger in the sense that we target the CCA secure version of HQC, while they target only the CPA secure version. However, their paper comes with a countermeasure, which consists of a constant-time BCH decoder with a low overhead.

Paper Organization. In Sect. 2, we review some background concepts for understanding HQC and our attack. The HQC is described in Sect. 3. The attack is presented in Sect. 4. Some mathematical and algorithmic aspects of the attack are analyzed in detail in Sect. 5. In Sect. 6, we analyze the practical performance of the attack against concrete HQC parameters. We conclude in Sect. 8.

2 Background

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

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

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

Definition 4 (Support). The support of a vector \mathbf{v} , denoted by $\text{supp}(\mathbf{v})$, is the set of indexes of its non-null entries.

We use zero-based numbering for the vectors indexes as we believe it allows more concise descriptions in some of the algorithms and analysis.

Definition 5 (Cyclic matrix). The cyclic matrix defined by a vector $\mathbf{v} = [v_0, \dots, v_{n-1}]$, is the matrix

$$\text{rot}(\mathbf{v}) = \begin{bmatrix} v_0 & v_{n-1} & \dots & v_1 \\ v_1 & v_0 & \dots & v_2 \\ \vdots & \vdots & \ddots & \vdots \\ v_{n-1} & v_{n-2} & \dots & v_0 \end{bmatrix}.$$

Definition 6 (Vector product). The product of two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}_2^n$ is given as

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{u} \text{rot}(\mathbf{v})^T = (\text{rot}(\mathbf{v})\mathbf{u}^T)^T = \mathbf{v} \text{rot}(\mathbf{u})^T = \mathbf{v} \cdot \mathbf{u}.$$

Definition 7 (Syndrome decoding problem). Consider the following input: a random binary matrix $\mathbf{H} \in \mathbb{F}_2^{k \times n}$, a random vector $\mathbf{s} \in \mathbb{F}_2^k$, and an integer $w > 0$. The syndrome decoding problem asks for a \mathbf{v} of weight w such that $\mathbf{v}\mathbf{H}^T = \mathbf{s}$.

The quasi-cyclic syndrome decoding problem is a restriction of the syndrome decoding problem, in which \mathbf{H} is a block matrix consisting of cyclic blocks.

The syndrome decoding problem is proven to be NP-hard [4]. Despite no complexity result on the quasi-cyclic variant, it is considered hard since all known decoding algorithms that exploit the cyclic structure have only a small advantage over general decoding algorithms for the non-cyclic case.

Definition 8 (Circular distance). The circular distance between the indexes i and j in a vector of length n is

$$\text{dist}_n(i, j) = \begin{cases} |i - j| & \text{if } |i - j| \leq \lfloor n/2 \rfloor, \\ n - |i - j| & \text{otherwise.} \end{cases}$$

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

Definition 9 (Spectrum of a vector). Let $\mathbf{v} = [v_0, v_1, \dots, v_{n-1}]$ be an element of \mathbb{F}_2^n . Then the spectrum of \mathbf{v} is the set

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

In some cases, it is important to consider the multiplicity of each distance d , that is the number of pairs of non-null entries that are at distance d apart. In such cases, we abuse notation and write $(d : m) \in \sigma(\mathbf{v})$ to denote that d appears with multiplicity m in vector \mathbf{v} .

Definition 10 (Mirror of a vector). Let $\mathbf{v} = [v_0, v_1, \dots, v_{n-1}]$ be an element of \mathbb{F}_2^n . Then the mirror of \mathbf{v} is the vector

$$\text{mirror}(\mathbf{v}) = [v_{n-1}, v_{n-2}, \dots, v_0].$$

We sometimes abuse notation and write $\text{mirror}(V)$, where V is the support of a vector \mathbf{v} , to represent the support of the mirror of \mathbf{v} .

Notice that the spectrum of a vector is invariant with respect to its circular shifts and its mirror.

Guo et al. [12] showed that it is possible to reconstruct a sparse vector from its spectrum. To solve this problem, they propose an algorithm that consists of a simple pruned depth-first search. Its description is given as Algorithm 1.¹ The main argument by Guo et al. for the efficiency of their algorithm is that unfruitful branches are pruned relatively early in the search.

Let α be the fraction of the $\lfloor n/2 \rfloor$ possible distances that are not in D , that is $\alpha = 1 - |D|/\lfloor n/2 \rfloor$. For each new level in the search tree, it is expected that a fraction α of the possible positions in the previous level survive the sieve imposed by line 10. Let MAXPATHS be the total number of paths that Guo’s et al. [12] algorithm can explore. Then

$$\text{MAXPATHS} = \prod_{\ell=2}^{w-1} \max(1, \lfloor n/2 \rfloor \alpha^\ell) = \lfloor n/2 \rfloor^\phi \alpha^{\phi(\phi+3)/2},$$

where ℓ represents the level in the search tree, and ϕ is the level at which each node has an expected number of child nodes lower than or equal to 1. Notice that, on average, the mirror test in line 5 cuts in half the number of paths the algorithm needs to explore until it finds the key. From the remaining paths, we expect that half of them have to be taken until the key is found. Therefore, considering \mathbf{WF}_{GJS} to be the average number of paths the algorithm explores until a key is found, we have

$$\mathbf{WF}_{\text{GJS}} = \frac{1}{4} \text{MAXPATHS} = \frac{1}{4} \lfloor n/2 \rfloor^\phi \alpha^{\phi(\phi+3)/2}.$$

¹ Here we present a slightly more general version of Guo’s et al. reconstruction algorithm that does not require the key’s spectrum to be completely determined, but the idea is the same.

Algorithm 1. GJS key reconstruction algorithm [12]

Data: n, w the length and weight of the secret vector \mathbf{y}
 D a set of distances outside $\sigma(\mathbf{y})$
 s a distance inside $\sigma(\mathbf{y})$
 V the partially recovered support of a shift of \mathbf{y} (initially set to $\{0, s\}$, where $s \in \sigma(\mathbf{y})$) is known
Result: V the support of some shift of \mathbf{y} , or \perp if $\sigma(\mathbf{y})$ is an invalid spectrum

```

1 begin
2   if  $|V| = w$  then
3     if  $V$  is the support of a shift of  $\mathbf{y}$  then
4       return  $V$ 
5     else if mirror( $V$ ) is the support of a shift of  $\mathbf{y}$  then
6       return mirror( $V$ )
7     else
8       return  $\perp$ 
9   for each position  $j = 1, \dots, n - 1$  which are not in  $V$  do
10    if  $\text{dist}_n(v, j) \notin D$  for all  $v$  in  $V$  then
11      Add  $j$  to  $V$ 
12       $\text{ret} \leftarrow$  recursive call with the updated set  $V$ 
13      if  $\text{ret} \neq \perp$  then
14        return  $V$ 
15      Remove  $j$  from  $V$ 
16  return  $\perp$ 

```

3 The HQC Encryption Scheme

3.1 Setup

On input 1^λ , where λ is the security parameter, the setup algorithm returns the public parameters $n, k, \delta, w, w_r, w_e$, from parameters table such as Table 1. For these parameters, an $[n, k]$ linear code \mathcal{C} , with an efficient decoding algorithm Ψ capable of correcting random errors of weight up to δ with overwhelming probability, is fixed. Parameters w, w_r and w_e correspond to the weights of the sparse vectors defined and used in the next sections.

3.2 Key Generation

Let $\mathbf{H} \in \mathbb{F}_2^{n \times 2n}$ be a quasi-cyclic matrix selected at random, in systematic form, that is $\mathbf{H} = [\mathbf{I} \mid \text{rot}(\mathbf{h})]$, for some vector \mathbf{h} . Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ be sparse vectors with weight $w(\mathbf{x}) = w(\mathbf{y}) = w$. Compute

$$\mathbf{s} = [\mathbf{x}|\mathbf{y}]\mathbf{H}^T = \mathbf{x} + \mathbf{y} \cdot \text{rot}(\mathbf{h})^T = \mathbf{x} + \mathbf{y} \cdot \mathbf{h}.$$

The public and secret key are $K_{\text{PUB}} = [\mathbf{s}|\mathbf{h}]$ and $K_{\text{SEC}} = [\mathbf{x}|\mathbf{y}]$, correspondingly.

From this construction, it is easy to see the relation between recovering the secret key from the public key and the quasi-cyclic syndrome decoding problem.

Table 1. Suggested parameters for some security levels [1].

Instance	Security	n_1	n_2	$n \approx n_1 n_2^a$	$k = k_1$	w	$w_r = w_e$	p_{fail}
Basic-I	128	766	29	22,229	256	67	77	2^{-64}
Basic-II	128	766	31	23,747	256	67	77	2^{-96}
Basic-III	128	796	31	24,677	256	67	77	2^{-128}
Advanced-I	192	796	51	40,597	256	101	117	2^{-64}
Advanced-II	192	766	57	43,669	256	101	117	2^{-128}
Advanced-III	192	766	61	46,747	256	101	117	2^{-192}
Paranoiac-I	256	766	77	59,011	256	133	153	2^{-64}
Paranoiac-II	256	766	83	63,587	256	133	153	2^{-128}
Paranoiac-III	256	796	85	67,699	256	133	153	2^{-192}
Paranoiac-IV	256	796	89	70,853	256	133	153	2^{-256}

^a The value of n is the smallest prime number greater than $n_1 n_2$.

3.3 Encryption

Let $\mathbf{m} \in \mathbb{F}_2^k$ be the message to be encrypted. First, choose two random sparse vectors $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{F}_2^n$ such that $w(\mathbf{r}_1) = w(\mathbf{r}_2) = w_r$. Then choose a random sparse vector $\mathbf{e} \in \mathbb{F}_2^n$ such that $w(\mathbf{e}) = w_e$. Let

$$\mathbf{u} = [\mathbf{r}_1 | \mathbf{r}_2] \mathbf{H}^T = \mathbf{r}_1 + \mathbf{r}_2 \cdot \mathbf{h}, \text{ and } \mathbf{v} = \mathbf{mG} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e}.$$

Return the ciphertext $\mathbf{c} = [\mathbf{u} | \mathbf{v}]$.

3.4 Decryption

Compute $\mathbf{c}' = \mathbf{v} + \mathbf{u} \cdot \mathbf{y}$. Notice that

$$\begin{aligned} \mathbf{c}' &= \mathbf{mG} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e} + (\mathbf{r}_1 + \mathbf{r}_2 \cdot \mathbf{h}) \cdot \mathbf{y} \\ &= \mathbf{mG} + (\mathbf{x} + \mathbf{y} \cdot \mathbf{h}) \cdot \mathbf{r}_2 + \mathbf{e} + (\mathbf{r}_1 + \mathbf{r}_2 \cdot \mathbf{h}) \cdot \mathbf{y} \\ &= \mathbf{mG} + \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{r}_1 \cdot \mathbf{y} + \mathbf{e}. \end{aligned}$$

Intuitively, since $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2$, and \mathbf{e} all have low weight, we expect $\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{r}_1 \cdot \mathbf{y} + \mathbf{e}$ to have a relatively low weight. This is made precise by Aguilar-Melchor et al. [1], where they propose the public parameters to ensure that $w(\mathbf{e}')$ is sufficiently low for it to be corrected out of \mathbf{c}' with overwhelming probability.

Therefore we can use the decoder Ψ to correct the errors in \mathbf{c}' and obtain $\mathbf{c}'' = \Psi(\mathbf{c}') = \mathbf{mG}$. We finally get \mathbf{m} by solving the overdetermined linear system $\mathbf{mG} = \mathbf{c}''$.

3.5 Security and Instantiation

In general, schemes based on syndrome decoding have to take care to avoid generic attacks based on Information Set Decoding [20, 22, 23]. Furthermore, the

quasi-cyclic structure of the code used to secure the secret key can make the scheme vulnerable to DOOM [21], or other structural attacks [11, 15].

To instantiate the scheme, the authors propose parameters for which they prove very low decryption error probability and resistance to the attacks mentioned. This error analysis allows the HQC to achieve IND-CCA2 security using the transformation of Hofheinz et al. [13].

Of particular interest for our timing attack, is the way that code \mathcal{C} is chosen. Their proposal is to build the tensor code $\mathcal{C} = \mathcal{C}_1 \otimes \mathcal{C}_2$, where the auxiliary codes are chosen as follows. \mathcal{C}_1 is a BCH(n_1, k_1, δ_1) code of length n_1 , dimension k_1 . \mathcal{C}_2 is a repetition code of length n_2 and dimension 1, that can decode up to $\delta_2 = \lfloor \frac{n_2-1}{2} \rfloor$. Therefore, to encode a message \mathbf{m} with respect to \mathcal{C} is equivalent to first encode it using the BCH code \mathcal{C}_1 , and then encode *each bit* of the resulting codeword with the repetition code \mathcal{C}_2 .

The suggested parameters for this instantiation are shown in Table 1. In this table, column p_{fail} contains an upper bound for the probability of a decryption failure for each instance of the scheme. The size of the public keys and ciphertexts correspond to $2n$ bits.

4 Timing Attack Against HQC

In the decryption algorithm, the decoder Ψ is used to correct the errors in the word

$$\mathbf{c}' = \mathbf{m}\mathbf{G} + \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{r}_1 \cdot \mathbf{y} + \mathbf{e},$$

where the attacker knows every element, except for the secret key consisting of \mathbf{x} and \mathbf{y} . For the original instantiation, where \mathcal{C} is the tensor product of a BCH code and a repetition code, the decoder Ψ consists of a sequence of two operations: first apply a repetition code decoder Ψ_2 , and then apply the BCH code decoder Ψ_1 . That is $\Psi(\mathbf{c}') = \Psi_1(\Psi_2(\mathbf{c}'))$.

The timing attack is based on the fact the BCH decoder implemented by Aguilar-Melchor et al. [18] is not constant-time, and is slower when there are more errors to be corrected. In other words, the decryption time leaks the number of errors that the repetition code (RC) decoder Ψ_2 was not able to correct.

Figure 1 shows the essentially linear relation between the decryption time and the number of errors corrected by the BCH decoder. We emphasize that the time considered is for complete decryption, not only the BCH decoding step. The weight distribution is centered between 9 and 10, thus error weights larger than 22 are rare (around 1%).

Let \mathbf{e}' be the error vector that Ψ_2 will try to correct, that is $\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{r}_1 \cdot \mathbf{y} + \mathbf{e}$. We note that it is useful to consider Guo's et al. [12] observation, used in their attack on QC-MDPC, that the weight of the product of two binary sparse vectors $\mathbf{a} \cdot \mathbf{b}$ is lower when the spectrums of \mathbf{a} and \mathbf{b} share more entries. However, this observation is not sufficient to enable us to accurately distinguish between distances in and out of the spectrum, because we are not just interested in the weight of \mathbf{e}' , but mainly in the probability that it has enough non-null entries in the same repetition blocks to cause Ψ_2 to leave decoding errors.

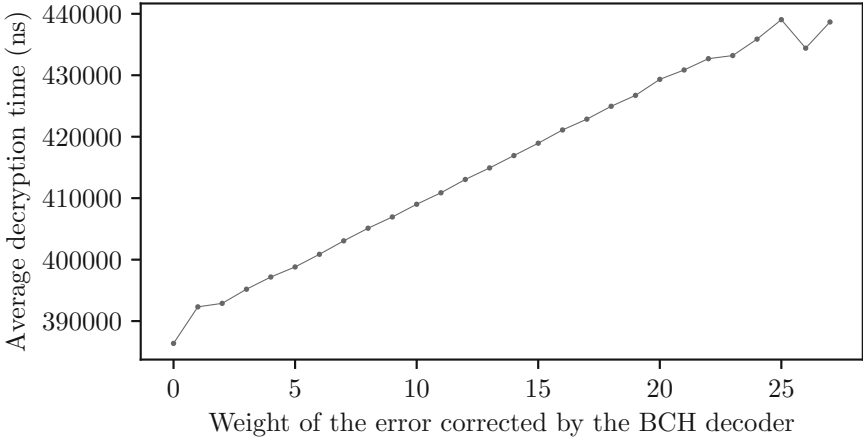


Fig. 1. The average decryption time for different weights of the errors corrected by the BCH decoder, considering 10 million decryption operations.

When the number of RC decoding errors for $\Psi_2(\mathbf{mG} + \mathbf{e}')$ is high, it means that \mathbf{e}' has a lot of non-null entries that are at a distance lower than the repetition block size n_2 . Therefore, if we understand how \mathbf{r}_1 and \mathbf{y} influence the number of entries lower than n_2 in $\sigma(\mathbf{r}_1 \cdot \mathbf{y})$, we can use our knowledge on \mathbf{r}_1 together with the decryption time to obtain information on \mathbf{y} .²

We make three observations that relate the spectrum of \mathbf{e}' to the spectrums of \mathbf{r}_1 and \mathbf{y} , (alternatively \mathbf{r}_2 and \mathbf{x}). These are presented in the next section based on empirical data, and their mathematical nature is explained in Sect. 5.1.

The timing attack then consists of two parts. In the first part, called the spectrum recovery, the attacker sends Alice a great number of ciphertexts and records the decryption times for each one. This step runs until it is gathered sufficient information on the spectrums of \mathbf{y} (or \mathbf{x}) for him to build a large set D of distances outside the spectrum, and to obtain a distance $s \in \sigma(\mathbf{y})$ (respectively, $\sigma(\mathbf{x})$). In the second part, the set D and distance s are passed to the key reconstruction algorithm.

It is important to notice that the the attacker needs only to recover one of \mathbf{x} or \mathbf{y} , because he can use the linear relation $\mathbf{s} = \mathbf{x} + \mathbf{y} \cdot \mathbf{h}$ to easily recover one from the other.

In the next sections, the two parts are presented in detail.

4.1 Spectrum Recovery

This is the part where timing information is used. Let Alice be the target secret key holder. The attacker sends Alice valid ciphertexts, and records the time she takes to decrypt each challenge. Since the attacker generated all ciphertexts, then, for each one of them, he knows \mathbf{r}_1 and \mathbf{r}_2 . The idea is that the attacker

² This sentence remains valid if we substitute \mathbf{y} and \mathbf{r}_1 by \mathbf{x} and \mathbf{r}_2 , respectively.

iteratively builds two arrays, \mathbf{T}_x and \mathbf{T}_y , such that $\mathbf{T}_x[d]$ ($\mathbf{T}_y[d]$) is the average of the decryption time when d is in the spectrum of \mathbf{r}_2 (resp. \mathbf{r}_1).

The algorithm for spectrum recovery is given as Algorithm 2. Notice that we are not choosing the vectors $\mathbf{r}_1, \mathbf{r}_2$, which are assumed to be random. Therefore CCA2 conversions [13] do not protect the scheme against this attack.

Algorithm 2. Estimating the decryption time for each possible distance in $\sigma(\mathbf{x})$ and $\sigma(\mathbf{y})$

Data: The HQC public parameters and Alice’s public key
 \mathcal{T} returns the target’s decryption time for the challenge passed as argument
 M number of decoding challenges
Result: $\mathbf{T}_x, \mathbf{T}_y$ average decryption time for candidate distances in $\sigma(\mathbf{x})$ and $\sigma(\mathbf{y})$, respectively

```

1 begin
2    $\mathbf{a}_y, \mathbf{b}_y, \mathbf{a}_x, \mathbf{b}_x \leftarrow$  zero-initialized arrays with  $\lfloor n/2 \rfloor$  entries each
3   for each decoding trial  $i = 1, 2, \dots, M$  do
4      $\mathbf{m} \leftarrow$  a random message in  $\mathbb{F}_2^k$ 
5      $\mathbf{c} \leftarrow$  encryption of  $\mathbf{m}$  using vectors  $\mathbf{r}_1$  and  $\mathbf{r}_2$  randomly chosen
6      $t = \mathcal{T}(\mathbf{c})$ 
7     for each distance  $d$  in  $\sigma(\mathbf{r}_1)$  do
8        $\mathbf{a}_y[d] \leftarrow \mathbf{a}_y[d] + t$ 
9        $\mathbf{b}_y[d] \leftarrow \mathbf{b}_y[d] + 1$ 
10    for each distance  $d$  in  $\sigma(\mathbf{r}_2)$  do
11       $\mathbf{a}_x[d] \leftarrow \mathbf{a}_x[d] + t$ 
12       $\mathbf{b}_x[d] \leftarrow \mathbf{b}_x[d] + 1$ 
13     $\mathbf{T}_x, \mathbf{T}_y \leftarrow$  zero-initialized array with  $\lfloor n/2 \rfloor$  positions
14    for each distance  $d$  in  $\{1, 2, \dots, \lfloor n/2 \rfloor\}$  do
15       $\mathbf{T}_x[d] \leftarrow \mathbf{a}_x[d]/\mathbf{b}_x[d]$ 
16       $\mathbf{T}_y[d] \leftarrow \mathbf{a}_y[d]/\mathbf{b}_y[d]$ 
17    return  $\mathbf{T}_x$  and  $\mathbf{T}_y$ 

```

To maximize the information obtained from each decryption timing, the proposed spectrum recovery procedure targets $\sigma(\mathbf{x})$ and $\sigma(\mathbf{y})$ simultaneously. This is interesting for the attacker since it may be the case that, after a number of challenges, the output \mathbf{T}_x does not have sufficient information on \mathbf{x} for it to be reconstructed, but \mathbf{T}_y is sufficient to recover \mathbf{y} .

Figure 2 shows the output of the spectrum recovery algorithm \mathbf{T}_y for $M = 1$ billion decryption challenges. On the left of the figure, we see that distances lower than n_2 have a significantly higher average decryption time. The figure shows that, in general, distances inside the spectrum of \mathbf{y} appears to have lower average decryption time. However, there is no clear line to classify a distance d as inside or outside $\sigma(\mathbf{y})$, based only on $\mathbf{T}_y[d]$, since this value appears to also depend on the neighbors of d .

Figure 3 shows another interval of the same data, but with one vertical line for each distance in the spectrum. This enables us to see that regions where there are more distances inside the spectrum appear to have higher average decryption time.

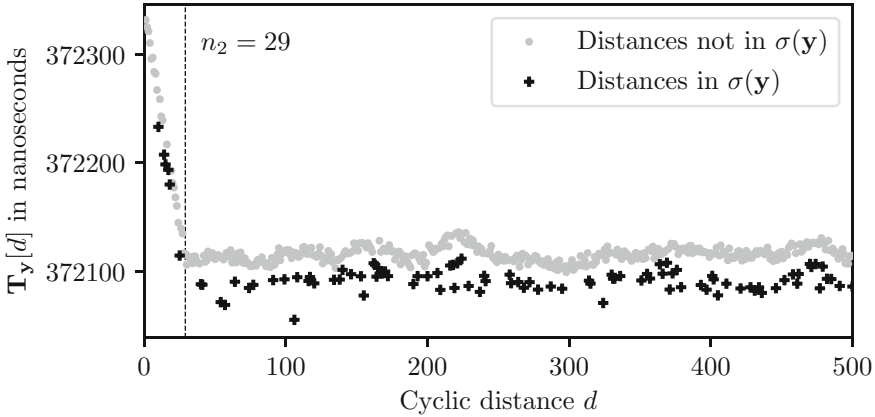


Fig. 2. The average decryption time $\mathbf{T}_y[d]$ for each distance d that can occur in \mathbf{r}_1 , for $M = 1$ billion.

Summarizing the analysis of the figures, we make the following three informal observations that allow us to distinguish between distances inside and outside the spectrums.

1. When d decreases from $d = n_2 - 1$ to $d = 1$, the value of $\mathbf{T}_y[d]$ increases, getting significantly higher than the rest of the values in \mathbf{T}_y .
2. When $d \in \sigma(\mathbf{y})$, the value of $\mathbf{T}_y[d]$ is lower than the average in the neighborhood of d .
3. When d has a large number of neighbors in $\sigma(\mathbf{y})$, the value of $\mathbf{T}_y[d]$ tends to be higher.

The reasons why we observe such behavior are analyzed in detail in Sect. 5.1.

Similarly to the GJS algorithm (Algorithm 1), our key reconstruction algorithm for the next part of the attack works with two inputs: a set D of distances outside the spectrum, and a distance s inside the spectrum. Figure 2 suggests that, when a sufficiently large number of decryption challenges are timed, it is easy to get a distance inside the spectrum with high probability by just taking the distance s such that $\mathbf{T}_y[s]$ is the minimum value in the array. However, it is not trivial to find a sufficiently large set D from \mathbf{T}_y . For this, we propose a routine called BUILD D , which is describe next.

BuildD: Building the Set of Distances not in $\sigma(\mathbf{y})$ from \mathbf{T}_y . We propose to use the following simple algorithm, that takes as input a value μ and the decryption times estimation \mathbf{T}_y , and outputs μ distances which it classifies as

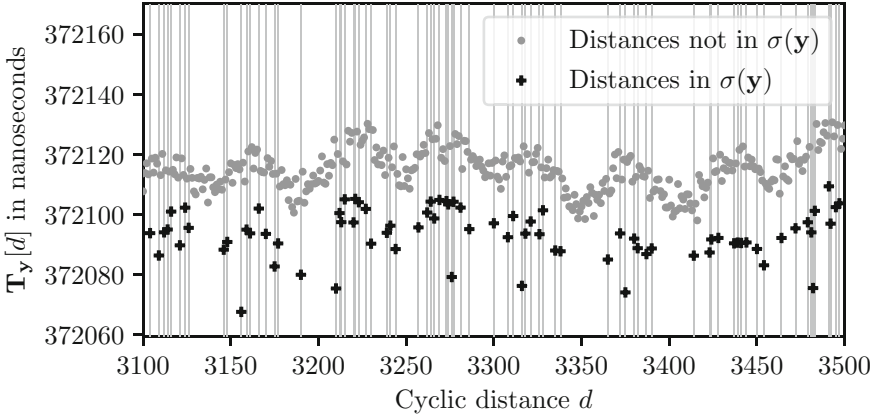


Fig. 3. A closer look at the behavior of $\mathbf{T}_y[d]$ for each distance d . The gray vertical lines represent distances inside $\sigma(\mathbf{y})$.

out of $\sigma(\mathbf{y})$. The idea is to select the μ values of d such that $\mathbf{T}_y[d]$ are among the highest of their corresponding neighborhood.

Let η be some small positive integer for which the probability that $\{d, d + 1, \dots, d + \eta - 1\} \subset \sigma(\mathbf{y})$ is negligible for all values of d . The value η will be the size of the neighborhood, which must contain at least one distance outside the spectrum. This value can be estimated by generating N random vectors, then computing the minimum value η for which η consecutive distances always contain at least one distance not in the vectors corresponding spectrums. For the Basic-I parameters, we obtained $\eta = 11$ for $N = 10000$.

For each d , we compute the difference between $\mathbf{T}_y[d]$ and the highest value of \mathbf{T}_y in the window $\{d - \lfloor \eta/2 \rfloor, \dots, d + \lfloor \eta/2 \rfloor - 1\}$. If the window contains invalid distances, we just truncate it to exclude them. In other words

$$\rho(d) = \left(\max_{i \in W_d} \mathbf{T}_y[i] \right) - \mathbf{T}_y[d],$$

where W_d is the intersection between $\{d - \lfloor \eta/2 \rfloor, \dots, d + \lfloor \eta/2 \rfloor - 1\}$ and the set of possible distances. The algorithm sorts the possible distances with respect to $\rho(d)$, and returns the μ values of d such that $\rho(d)$ are the lowest ones.

Let $\text{BUILDD}(\mathbf{T}_y, \mu)$ be the output of the algorithm just described for the given inputs. Since the key reconstruction only works if D is a large set of distances not in the spectrum, it is natural to define the quality of the input \mathbf{T}_y as

$$\text{QUALITY}(\mathbf{T}_y) = \max \{ \mu : \text{BUILDD}(\mathbf{T}_y, \mu) \cap \sigma(\mathbf{y}) = \emptyset \}.$$

Figure 4 helps us visualize why this algorithm works. For $M = 1$ billion decryptions, it is easy to see that the distances between $\mathbf{T}_y[d]$ and $\max_{i \in W_d} \mathbf{T}_y[i]$ should be smaller when d is not in the spectrum of \mathbf{y} . However, it is not clear yet how many decryptions are necessary for the algorithm to be able to build

sufficiently large sets D , that is, to obtain high values for $\text{QUALITY}(\mathbf{T}_y)$. This is considered in the experimental analysis in Sect. 6.2.

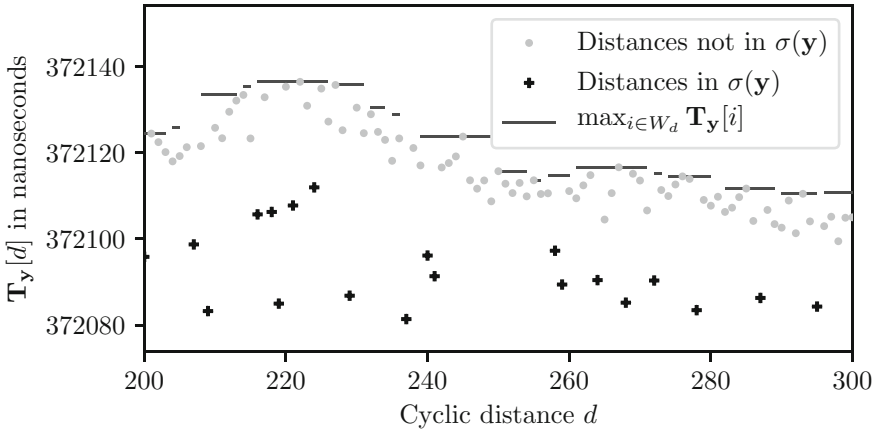


Fig. 4. Illustration of the values $\max_{i \in W_d} T_y[i]$ for each distance d , considering windows of size $\eta = 11$, after $M = 1$ billion decryptions.

4.2 Reconstructing \mathbf{y} from Partial Information on Its Spectrum

We propose the key reconstruction algorithm given as Algorithm 3, which is a simple randomized extension of Guo’s et al. algorithm. Instead of performing a depth-first search for the key, at each level of the search tree, the algorithm chooses the next node at random.

We give a brief description of the algorithm. Parameters s , which must be a distance inside the spectrum of \mathbf{y} , and D , which is a set of distances outside the spectrum of \mathbf{y} , are obtained in the first part of the attack. At each iteration, the algorithm starts with the set $V = \{0, s\}$ and tries to complete it with $w - 2$ indexes. To complete the support V , the algorithm chooses at random an index inside the auxiliary set Γ_ℓ , which contains, for each level ℓ , the possible positions to complete the support. That is, Γ_ℓ consists of all the elements from $\{0, \dots, n - 1\}$ which are not in V , and whose circular distance to any index in V is not in D .

Notice that it is easy to perform the tests in lines 12 and 14 without knowing the secret key. Let $\bar{\mathbf{y}}$ be the vector with support V found in the algorithm’s main loop. Consider all possible cyclic shifts of $\bar{\mathbf{y}}$, denoted by $\mathbf{y}^0, \dots, \mathbf{y}^{n-1}$. To test if $\bar{\mathbf{y}}$ is a shift of \mathbf{y} , we look for a shift \mathbf{y}^i such that the weight of the vector $\bar{\mathbf{x}} = \mathbf{s} + \mathbf{y}^i \cdot \mathbf{h}$ is $w(\bar{\mathbf{x}}) = w$. If we find one, then $\bar{\mathbf{y}}$ is a shift of \mathbf{y} (high probability), or we have found an equivalent secret key for the given public key (\mathbf{h}, \mathbf{s}) . If we do not find one, then we start a new iteration.

The complexity of the algorithm is analyzed in Sect. 5.2, while its practical performance is shown in Sect. 6.1.

Algorithm 3. Randomized key reconstruction algorithm

```

Data: The HQC public parameters and Alice's public key
           $s$  a distance inside  $\sigma(\mathbf{y})$ 
           $D$  a set of distances which are not in  $\sigma(\mathbf{y})$ 
Result:  $V$  the support of a rotation of  $\mathbf{y}$ 
1 begin
2   do
3      $V \leftarrow \{0, s\}$ 
4      $\Gamma_2 \leftarrow \{i \in \{1, \dots, \lfloor n/2 \rfloor\} - V : \text{dist}_n(i, v) \notin D \text{ for all } v \in V\}$ 
5      $\ell \leftarrow 2$ 
6     while  $|V| < w$  and  $|\Gamma_\ell| > 0$  do
7        $p \leftarrow$  a random element from  $\Gamma_\ell$ 
8        $V \leftarrow V \cup \{p\}$ 
9        $\Gamma_{\ell+1} \leftarrow \{i \in \Gamma_\ell : \text{dist}_n(i, v) \notin D \text{ for all } v \in V\}$ 
10       $\ell \leftarrow \ell + 1$ 
11   while Both  $V$  and  $\text{mirror}(V)$  are not the support of a rotation of  $\mathbf{y}$ ;
12   if  $V$  is the support of a shift of  $\mathbf{y}$  then
13     return  $V$ 
14   else if  $\text{mirror}(V)$  is the support of a shift of  $\mathbf{y}$  then
15     return  $\text{mirror}(V)$ 

```

5 Analysis

In this section we analyze two aspects of the attack. First we explain why it is possible to distinguish between distances inside and outside the spectrum based on decryption time. Then we analyze the complexity of the randomized key reconstruction algorithm, and how it compares to the one presented by Guo et al. [12].

5.1 Distinguishing Distances Inside and Outside the Spectrum

We know that the decryption time is related to the number of errors left by the repetition code (RC) decoder. Our main observation is that the number of RC decoding errors depends on how the spectrums of \mathbf{r}_1 and \mathbf{r}_2 relate to those of \mathbf{y} and \mathbf{x} , respectively.

Consider the error to be corrected by the RC decoder, given by

$$\mathbf{e}' = \mathbf{r}_1 \cdot \mathbf{y} + \mathbf{r}_2 \cdot \mathbf{x} + \mathbf{e}.$$

An RC decoding error occurs when \mathbf{e}' contains more than $(n_2 - 1)/2$ nonzero errors in the same repetition block. Therefore, an RC decoding error has higher probability of occurring when the spectrum of \mathbf{e}' contains small distances with high multiplicity, and in particular, when $\sigma(\mathbf{e}')$ contains a lot of distances lower than the repetition block length n_2 . We also expect that $\sigma(\mathbf{e}')$ contains small distances when $\sigma(\mathbf{r}_2 \cdot \mathbf{x})$ and $\sigma(\mathbf{r}_1 \cdot \mathbf{y})$ also contain small distances. In the following discussion, we focus on $\mathbf{r}_1 \cdot \mathbf{y}$, but we could have used $\mathbf{r}_2 \cdot \mathbf{x}$ without any difference.

The above paragraph motivates us to better understand what causes the spectrum of $\mathbf{r}_1 \cdot \mathbf{y}$ to contain small distances. Unfortunately, the strong dependency between the rows of $\text{rot}(\mathbf{y})^T$ can make it very hard to perform a satisfactory statistical analysis on the product $\mathbf{r}_1 \cdot \mathbf{y}$.

Therefore, we study a simpler problem, namely to describe $\sigma(\mathbf{r}_1 \cdot \mathbf{y})$ as a function of $\sigma(\mathbf{r}_1)$ and $\sigma(\mathbf{y})$, but restricted to the case where $w(\mathbf{r}_1) = w(\mathbf{y}) = 2$. Even though it is not the general case, it can give us a good intuition on why the attack works. The analysis is given in the following lemma. First we discuss the implications of the lemma and how it can be used to distinguish between distances inside and outside the spectrums of the secret key, and then we prove it.

Lemma 1. *Let $\mathbf{y}, \mathbf{r} \in \mathbb{F}_2^n$ be two binary vectors of weight 2, where n is an odd prime. Let α and β be the only distances in $\sigma(\mathbf{y})$ and $\sigma(\mathbf{r})$, respectively. Then, we have the following possibilities.³*

If $\alpha = \beta$, then

$$\sigma(\mathbf{r} \cdot \mathbf{y}) = \{\text{dist}_n(0, 2\alpha) : 1\} = \{\text{dist}_n(0, 2\beta) : 1\}. \tag{1}$$

If $\alpha \neq \beta$, then

$$\sigma(\mathbf{r} \cdot \mathbf{y}) = \{\alpha : 2, \tag{2}$$

$$\beta : 2, \tag{3}$$

$$|\beta - \alpha| : 1, \tag{4}$$

$$\text{dist}_n(0, \beta + \alpha) : 1\}. \tag{5}$$

Interpreting Lemma 1. Intuitively, α represents distances inside the spectrum of the secret vector \mathbf{y} , while β represents distances inside the spectrum of \mathbf{r}_1 . We now restate the observations from Sect. 4.1 with brief discussions on why they happen, using the lemma to help us.

1. When β decreases from $\beta = n_2 - 1$ to $\beta = 1$, the value of $\mathbf{T}_y[\beta]$ increases, getting significantly higher than the rest of the values in \mathbf{T}_y .

From (3), distance β in $\sigma(\mathbf{r}_1)$ can cause $\sigma(\mathbf{r}_1 \cdot \mathbf{x})$ to contain β with multiplicity 2. Therefore when $\beta < n_2$, it can be responsible for more RC errors than values of $\beta \geq n_2$. The reason why $\mathbf{T}_y[\beta]$ gets increasingly higher when β approaches 1 is that, we get an increasing incidence of $\beta + \alpha < n_2$, where $\alpha \in \sigma(\mathbf{y})$. Therefore, from (5), these values of β tend to cause more distances lower than n_2 in $\sigma(\mathbf{r}_1 \cdot \mathbf{y})$.

2. When $\beta \in \sigma(\mathbf{y})$, the value of $\mathbf{T}_y[\beta]$ is lower than the average in the neighborhood of β .

Comparing both cases considered by the lemma, we see that values of $\beta = \alpha$ for some $\alpha \in \sigma(\mathbf{y})$ (Case 1) are expected to produce a lower number of small distances in $\sigma(\mathbf{r}_1 \cdot \mathbf{x})$ than values of $\beta \neq \alpha$ for all $\alpha \in \sigma(\mathbf{y})$ (Case 2).

³ Recall that we use $(\gamma : m) \in \sigma(\mathbf{y})$ to denote that cyclic distance γ occurs m times between non-null entries of \mathbf{y} .

3. When β has a large number of neighbors in $\sigma(\mathbf{y})$, the value of $\mathbf{T}_y[\beta]$ tends to be higher.

Using (4), we have that $\mathbf{T}_y[\beta]$ tends to be higher when more values of $\alpha \in \sigma(\mathbf{y})$ satisfy $|\beta - \alpha| < n_2$. In fact, the lemma even helps us formalize the neighborhood of β as the distances d between $\beta - n_2 < d < \beta + n_2$.

We now proceed with the proof of Lemma 1.

Proof (Lemma 1). Let α_1, α_2 and β_1, β_2 be the positions of the two ones in \mathbf{y} and \mathbf{r} , respectively. We can suppose without loss of generality that

$$\alpha_2 = \alpha_1 + \alpha \bmod n, \text{ and } \beta_2 = \beta_1 + \beta \bmod n,$$

since if this is not the case, we can just swap the corresponding values.

The product $\mathbf{r} \cdot \mathbf{y}$ consists of the sum of two circular shifts of \mathbf{y} : one by β_1 , and the other of β_2 positions, denoted by $\text{shift}_{\beta_1}(\mathbf{y})$ and $\text{shift}_{\beta_2}(\mathbf{y})$, respectively. More formally

$$\mathbf{r} \cdot \mathbf{y} = \mathbf{r} \text{rot}(\mathbf{y})^T = \text{shift}_{\beta_1}(\mathbf{y}) + \text{shift}_{\beta_2}(\mathbf{y}),$$

where

$$\begin{aligned} \text{supp}(\text{shift}_{\beta_1}(\mathbf{y})) &= \{\alpha_1 + \beta_1 \bmod n, \alpha_2 + \beta_1 \bmod n\} \\ &= \{\alpha_1 + \beta_1 \bmod n, \alpha_1 + \alpha + \beta_1 \bmod n\}, \end{aligned}$$

and

$$\begin{aligned} \text{supp}(\text{shift}_{\beta_2}(\mathbf{y})) &= \{\alpha_1 + \beta_2 \bmod n, \alpha_2 + \beta_2 \bmod n\} \\ &= \{\alpha_1 + \beta_1 + \beta \bmod n, \alpha_1 + \alpha + \beta_1 + \beta \bmod n\}. \end{aligned}$$

Therefore the weight of $\mathbf{r} \cdot \mathbf{y}$ is at most 4, but can be lower if the supports above share some of their entries. We consider separately the cases when $\alpha = \beta$ and $\alpha \neq \beta$. These cases are illustrated in Fig. 5.

Case $\alpha = \beta$. In this case, we have:

$$\text{supp}(\text{shift}_{\beta_1}(\mathbf{y})) = \{\alpha_1 + \beta_1 \bmod n, \alpha_1 + \alpha + \beta_1 \bmod n\},$$

and

$$\text{supp}(\text{shift}_{\beta_2}(\mathbf{y})) = \{\alpha_1 + \beta_1 + \alpha \bmod n, \alpha_1 + 2\alpha + \beta_1 \bmod n\}.$$

The supports of the shifts share the entry $\alpha_1 + \beta_1 + \alpha \bmod n$. But notice that this is the only shared entry, since the fact that n is odd implies $\alpha_1 + \beta_1 \not\equiv \alpha_1 + \beta_1 + 2\alpha \bmod n$. Then, summing the shifts of \mathbf{y} we get

$$\text{supp}(\mathbf{r} \cdot \mathbf{y}) = \{\alpha_1 + \beta_1 \bmod n, \alpha_1 + 2\alpha + \beta_1 \bmod n\}.$$

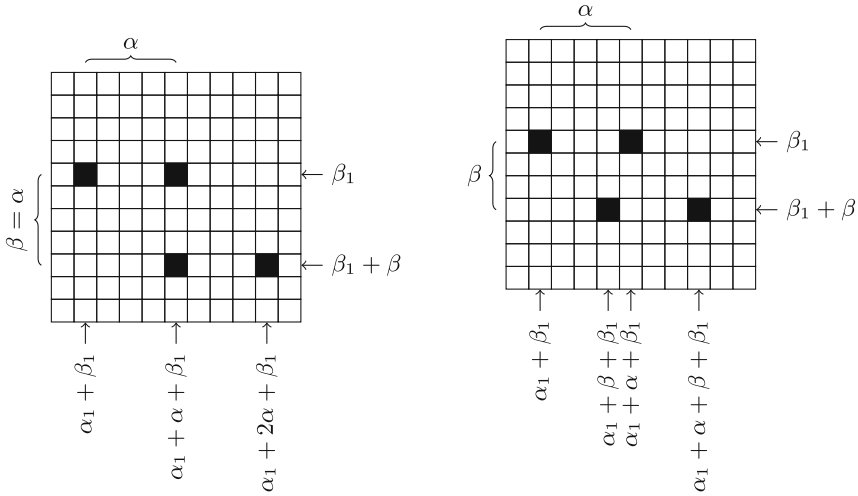


Fig. 5. The cases when $\alpha = \beta$ (left), and $\alpha \neq \beta$ (right).

Therefore, using the facts that $\alpha \leq \lfloor n/2 \rfloor$ and n is odd, we get

$$\begin{aligned} \sigma(\mathbf{r} \cdot \mathbf{y}) &= \{\text{dist}_n(\alpha_1 + \beta_1 \bmod n, \alpha_1 + 2\alpha + \beta_1 \bmod n) : 1\} \\ &= \{\text{dist}_n(0, 2\alpha \bmod n) : 1\} \\ &= \{\text{dist}_n(0, 2\alpha) : 1\}. \end{aligned}$$

Case $\alpha \neq \beta$. We begin by showing that the supports of the shifts do not share any entry. It is clear that $\alpha_1 + \beta_1$ is not equivalent to $\alpha_1 + \beta_1 + \beta$ nor $\alpha_1 + \alpha + \beta_1 + \beta \pmod n$ since $1 < \alpha, \beta \leq (n - 1)/2$. The same can easily be seen for $\alpha_1 + \alpha + \beta_1$.

Therefore, spectrum of $\mathbf{r} \cdot \mathbf{y}$ consists of the following distances.

1. $\text{dist}_n(\alpha_1 + \beta_1, \alpha_1 + \beta_1 + \beta) = \text{dist}_n(0, \beta) = \beta$.
2. $\text{dist}_n(\alpha_1 + \beta_1, \alpha_1 + \alpha + \beta_1) = \text{dist}_n(0, \alpha) = \alpha$.
3. $\text{dist}_n(\alpha_1 + \beta_1, \alpha_1 + \alpha + \beta_1 + \beta) = \text{dist}_n(0, \alpha + \beta)$.
4. $\text{dist}_n(\alpha_1 + \beta_1 + \beta, \alpha_1 + \alpha + \beta_1) = \text{dist}_n(\beta, \alpha) = \text{dist}_n(0, \alpha - \beta) = |\alpha - \beta|$.
5. $\text{dist}_n(\alpha_1 + \beta_1 + \beta, \alpha_1 + \alpha + \beta_1 + \beta) = \text{dist}_n(0, \alpha) = \alpha$.
6. $\text{dist}_n(\alpha_1 + \alpha + \beta_1, \alpha_1 + \alpha + \beta_1 + \beta) = \text{dist}_n(0, \beta) = \beta$.

Counting the multiplicities of these distances, we get the desired result. □

5.2 Probabilistic Analysis of the Key Reconstruction Algorithm

In this section, we first analyze our randomized variant of the key reconstruction algorithm, given as Algorithm 3 in Sect. 4.2. We then compare it to Guo’s et al. recursive algorithm, described as Algorithm 1 in the end of Sect. 2.

In each iteration, the algorithm performs a random walk down the search tree, starting from the root $\{0, s\}$, corresponding to $\ell = 2$, and ending in one of

its leaves. Therefore, for the algorithm to succeed in finding \mathbf{y} , it has to choose, in each level of the search, an element in $\text{supp}(\mathbf{y})$.

Let s be a distance in $\sigma(\mathbf{y})$. Suppose the search is at level ℓ , and the algorithm has chosen, until now, the elements $V_\ell = \{v_1 = 0, v_2 = s, \dots, v_\ell\}$, all in the support of \mathbf{y} . Let Γ_ℓ be the set of possible choices at level ℓ , then

$$\Gamma_\ell = \{p \in (\{0, \dots, n - 1\} - V_\ell) : \text{dist}_n(p, v) \notin D \text{ for all } v \in V_\ell\}.$$

We now have exactly $w - |V_\ell|$ good choices for the next level, which gives us

$$\Pr(v_{\ell+1} \in \text{supp}(\mathbf{y}) \mid V_\ell \subset \text{supp}(\mathbf{y})) = \frac{w - |V_\ell|}{|\Gamma_\ell|} = \frac{w - \ell}{|\Gamma_\ell|}.$$

Remember that the spectrum recovery algorithm can find either \mathbf{y} or $\text{mirror}(\mathbf{y})$, and both are of interest to the attacker. Therefore, we can write the probability that the algorithm successfully finds the key as

$$\Pr(\text{Success}) = 2 \prod_{\ell=2}^{w-1} \frac{w - \ell}{|\Gamma_\ell|},$$

where the product starts at level $\ell = 2$ since the search begins with $V_2 = \{0, s\}$, and it ends at level $\ell = w - 1$ because this is the last level in which a choice is made. The factor 2 comes from the mirror test.

Unfortunately, it is not easy to compute the distribution of $|\Gamma_\ell|$, because of the dependency between distances in D and elements in V_ℓ . However, we can approximate its expected value using an argument similar to the one used by Guo et al. [12]. Let α be the probability that a distance is not in D , that is $\alpha = 1 - |D|/\lfloor n/2 \rfloor$. At level ℓ , there are $w - \ell$ choices that are in $\text{supp}(\mathbf{y})$, and ℓ positions already in V_ℓ . For the other $n - w$ positions that are not in the support of \mathbf{y} , we expect a fraction of α^ℓ of them to have survived the sieves of each level. Therefore, we have

$$\mathbb{E} |\Gamma_\ell| \approx (n - w)\alpha^\ell + w - \ell.$$

We define the work factor $\mathbf{WF}_{\text{RAND}}$ of this algorithm as the expected number of paths it needs to explore until it finds the secret key. Then, using the approximation above, its value is

$$\begin{aligned} \mathbf{WF}_{\text{RAND}} &= \frac{1}{\Pr(\text{Success})} \\ &\approx \frac{1}{2} \prod_{\ell=2}^{w-1} \frac{(n - w)\alpha^\ell + w - \ell}{w - \ell} = \frac{1}{2} \prod_{\ell=2}^{w-1} \left(\frac{(n - w)\alpha^\ell}{w - \ell} + 1 \right). \end{aligned}$$

Looking at the term in each level ℓ , they appear to be lower than the corresponding ones for Guo's et al. algorithm. However, just looking at the expressions, it is not clear how they compare.

To better understand how they compare, consider Fig. 6, which shows a concrete comparison of the work factors for both algorithms when the input D has an increasing number of distances outside the spectrum. We considered parameters for three HQC variants. Since the range of $|D|$ varies according to the parameters n and w , we normalized its value with respect to the average of the total number of distances outside the spectrum, denoted by Δ . To estimate Δ for each pair (n, w) , we generated 1000 different random vectors and computed the average number of distances outside the spectrums. We can see that the work factor of the randomized algorithm is typically more than 3 orders of magnitude lower than Guo’s et al. [12] recursive one.

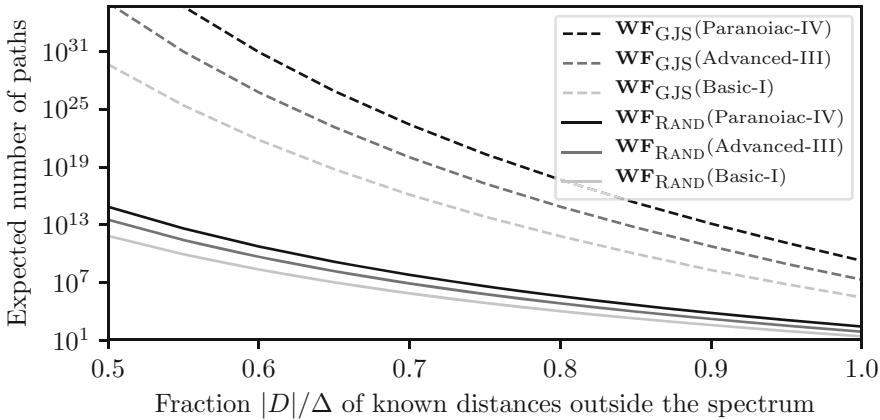


Fig. 6. Comparison between Guo’s et al. [12] key reconstruction algorithm and our randomized variant with respect to the expected number of paths until the secret key is found. For each set of parameters (n, w) the value of $|D|$ is normalized using the average number of distances outside the spectrum, denoted by Δ .

6 Experimental Results

In this section, we present our results for the timing attack against the Basic-I parameters of the HQC. We consider the two parts of the attack separately. First we run experiments on the key reconstruction algorithm to find out how much information on the spectrum it needs to run efficiently. We then run simulations to estimate how many decryptions timings an attacker needs to perform to be able to reconstruct the key. The source code and data are available at www.ime.usp.br/tpaiva.

6.1 Performance of the Key Reconstruction Algorithm

We want to determine how many entries outside the spectrum of the secret vector y the attacker needs to know for the key reconstruction algorithm to efficiently

reconstruct the vector. In other words, we are interested in how large the set D needs to be. Figure 6 gives us a hint on this matter, but it does not give us a concrete estimation of the key reconstruction algorithm’s performance.

Table 2 shows the performance of both key reconstruction algorithms, the GJS and our randomized variant, when given sets D of different sizes for the Basic-I HQC parameters. For each considered size for the set D , we generated 10 random secret keys and considered D as a random set of $|D|$ distances outside the spectrum. The distance s was selected at random from the secret key spectrum. For a more clear interpretation of the results, we considered, in the second column, the approximate average number $\Delta = 9104$ of distances not in the spectrum, to normalize the values $|D|$. We then ran C implementations of the algorithms with parameters D and s . This experiment was performed on an Intel i7-8700 CPU at 3.20 GHz, using its 12 hyperthreads.

Table 2. Performance of the key reconstruction algorithms when input D has different sizes, for the Basic-I HQC parameters.

$ D $	$ D /\Delta$	Randomized variant of GJS reconstruction algorithm			GJS reconstruction algorithm
		$\mathbf{WF}_{\text{RAND}}$	Median of the number of paths	Median of the CPU time (s)	Median of the CPU time (s)
9104	100%	28	63	0.51	0.98
8648	95%	99	80	0.51	10.78
8192	90%	407	232	0.50	772.64
7736	85%	1957	1714	0.75	6801.10
7280	80%	11394	9995	1.96	–
6824	75%	83670	54721	10.02	–
6368	70%	816671	365604	75.63	–
5912	65%	11355108	8472060	2767.90	–
5456	60%	246873607	–	–	–

We can see that our randomized algorithm performs much better than Guo’s et al. [12] one. This not only implies that the randomized algorithm allows faster key reconstruction, but also that it allows the attacker to recover the key with less interaction with the secret key holder. The estimates for the number of paths $\mathbf{WF}_{\text{RAND}}$ appear to be sufficiently accurate for our purposes, with only a minor discrepancy when $D/\Delta = 100\%$ that happens because of the concurrent hyperthreads. From $D/\Delta = 60\%$ down, the randomized algorithm starts taking too long to finish. Therefore, we consider that we are able to efficiently reconstruct the key when $D/\Delta \geq 65\%$.

6.2 Communication Cost

We now analyze how many decryption challenges an attacker needs to send to the secret key holder for a successful attack. In this paper, we only considered

the Basic-I HQC parameters, but this experiment can easily be extended for the other parameters.

For the analysis, 10 secret keys were generated at random, and for each of them we ran the spectrum recovery algorithm for $M = 700$ million challenges. For each number challenges i , consider the quality of the decryption time estimates \mathbf{T}_x^i and \mathbf{T}_y^i , given by

$$\begin{aligned} \text{QUALITY}(\mathbf{T}_x^i) &= \max \{ \mu : \text{BUILDD}(\mathbf{T}_x^i, \mu) \cap \sigma(\mathbf{x}) = \emptyset \}, \text{ and} \\ \text{QUALITY}(\mathbf{T}_y^i) &= \max \{ \mu : \text{BUILDD}(\mathbf{T}_y^i, \mu) \cap \sigma(\mathbf{y}) = \emptyset \}, \end{aligned}$$

where BUILDD is the algorithm described in the end of Sect. 4.1. For the BUILDD procedure, we considered the window size $\eta = 11$, which was obtained by the independent simulation described in the end of Sect. 4.1.

Based on the results from the previous section, we consider that the key reconstruction algorithm can efficiently recover \mathbf{x} or \mathbf{y} , when either $\text{QUALITY}(\mathbf{T}_x^i)$ or $\text{QUALITY}(\mathbf{T}_y^i)$ is greater than 5912, correspondingly.

Figure 7 shows the result of the experiment. We can see that with about 400 million of challenges, efficient key reconstruction is possible. After 600 million challenges, almost all distances outside the spectrum can be correctly identified.

7 Discussion on Countermeasures

The most obvious countermeasure against this timing attack is to use constant-time BCH decoders [24,25]. However, these decoders were proposed recently and they are not well studied yet. As such, their security against other types of side-channel attacks needs further investigation.

Walters and Sinha Roy [25] studied constant-time BCH decoders in the context of LAC [16]. Their decoder yields overheads between 10% and 40% when used in LAC. The optimized decoder proposed by Wafo-Tapa et al. [24] can yield reasonable overheads, between 3% and 11%, for the different security levels provided by the HQC instances. Hopefully, with further study on constant-time BCH decoders, lower overheads can be achieved.

If the slowdown factor is a problem, one could try to add a number of errors to the partially decoded vector right before the BCH decoding procedure. Next, we explain the rationale behind this idea. Consider the vector $\mathbf{c}' = \mathbf{m}\mathbf{G} + \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{r}_1 \cdot \mathbf{y} + \mathbf{e}$. When applying the repetition code decoder to each block of n_2 elements of \mathbf{c}' , we can estimate the probability of a repetition decoding error from the number of ones (or zeros) in the block. For example, if the number of 1's and 0's in a block are similar (both close to $n_2/2$), then the probability of a decoding error to occur is high. This might make it possible to estimate, within some statistical margin, the number of errors that the repetition code decoder has left for the BCH decoder. Then, one can add intentional errors to the partially decoded vector $\mathbf{c}'' = \Psi_2(\mathbf{c}')$, for it to have a weight W , where W is a constant error weight which the BCH decoder can correct. Further study and a careful probabilistic analysis is needed to understand if a decoder using this idea is secure.

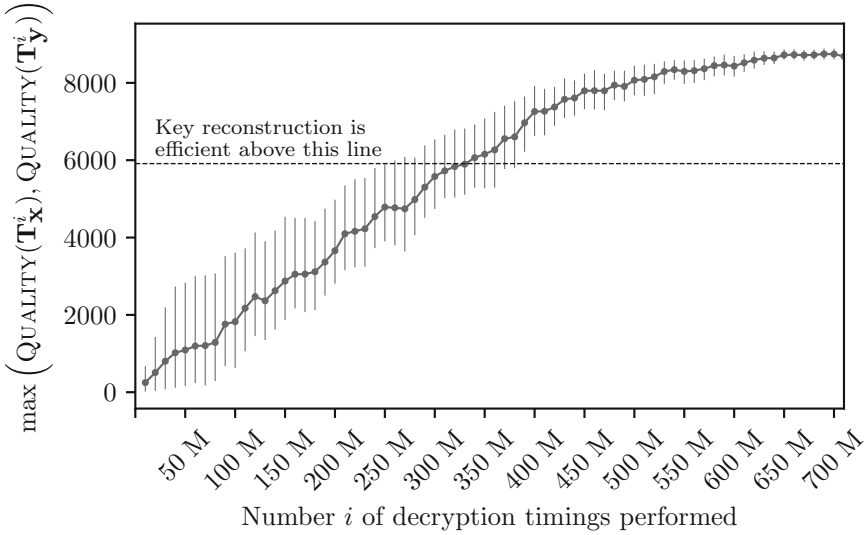


Fig. 7. Number of decryption timings an attacker needs to perform before the key can be successfully reconstructed. A confidence level of 95% was considered for the error bars.

8 Conclusion

In this paper, we present the first timing attack on the HQC encryption scheme. The attack depends on the choice of the parameter code \mathcal{C} and its decoder implementation. We show that the attack is practical, requiring about 400 million decryption timings to be performed. This makes the use of constant-time decoders for \mathcal{C} mandatory.

We discuss possible countermeasures against this timing attack, with the preferred one being to use constant-time BCH decoders [25]. However, further study is needed for the secure and efficient adoption of these decoders. Other solution would be to use codes for which efficient constant-time decoders are known. One interesting future work would be to find alternatives for the code \mathcal{C} that admit compact keys and efficient constant-time decoders, and for which we can prove negligible decryption failure probability.

Acknowledgments. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9.

References

1. Aguilar-Melchor, C., Blazy, O., Deneuville, J.C., Gaborit, P., Zémor, G.: Efficient encryption from random quasi-cyclic codes. *IEEE Trans. Inf. Theory* **64**(5), 3927–3943 (2018)
2. Albrecht, M., Cid, C., Paterson, K.G., Tjhai, C.J., Tomlinson, M.: NTS-KEM (2018)
3. Baldi, M.: QC-LDPC code-based cryptosystems. *QC-LDPC Code-Based Cryptography*. SECE, pp. 91–117. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-02556-8_6
4. Berlekamp, E.R., McEliece, R.J., Van Tilborg, H.C.: On the inherent intractability of certain coding problems. *IEEE Trans. Inf. Theory* **24**(3), 384–386 (1978)
5. Bernstein, D.J., et al.: Classic McEliece: conservative code-based cryptography (2019)
6. Bettaieb, S., Bidoux, L., Gaborit, P., Marcatel, E.: Preventing timing attacks against RQC using constant time decoding of Gabidulin codes. In: Ding, J., Steinwandt, R. (eds.) *PQCrypto 2019*. LNCS, vol. 11505, pp. 371–386. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25510-7_20
7. D’Anvers, J.P., Tiepelt, M., Vercauteren, F., Verbauwheide, I.: Timing attacks on error correcting codes in post-quantum secure schemes. *Cryptology ePrint Archive, Report 2019/292* (2019). <https://eprint.iacr.org/2019/292>
8. Eaton, E., Lequesne, M., Parent, A., Sendrier, N.: QC-MDPC: a timing attack and a CCA2 KEM. In: Lange, T., Steinwandt, R. (eds.) *PQCrypto 2018*. LNCS, vol. 10786, pp. 47–76. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-79063-3_3
9. Fabšič, T., Hromada, V., Stankovski, P., Zajac, P., Guo, Q., Johansson, T.: A reaction attack on the QC-LDPC McEliece cryptosystem. In: Lange, T., Takagi, T. (eds.) *PQCrypto 2017*. LNCS, vol. 10346, pp. 51–68. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59879-6_4
10. Gabidulin, E.M.: Theory of codes with maximum rank distance. *Probl. Peredachi Inform.* **21**(1), 3–16 (1985)
11. Guo, Q., Johansson, T., Löndahl, C.: A new algorithm for solving Ring-LPN with a reducible polynomial. *IEEE Trans. Inf. Theory* **61**(11), 6204–6212 (2015)
12. Guo, Q., Johansson, T., Stankovski, P.: A key recovery attack on MDPC with CCA security using decoding errors. In: Cheon, J.H., Takagi, T. (eds.) *ASIACRYPT 2016*. LNCS, vol. 10031, pp. 789–815. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_29
13. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) *TCC 2017*. LNCS, vol. 10677, pp. 341–371. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_12
14. Joiner, L.L., Komo, J.J.: Decoding binary BCH codes. In: *Proceedings IEEE Southeastcon 1995*. Visualize the Future, pp. 67–73. IEEE (1995)
15. Löndahl, C., Johansson, T., Shooshtari, M.K., Ahmadian-Attari, M., Aref, M.R.: Squaring attacks on mceliece public-key cryptosystems using quasi-cyclic codes of even dimension. *Des. Codes Crypt.* **80**(2), 359–377 (2016)
16. Lu, X., et al.: LAC: Practical Ring-LWE based public-key encryption with byte-level modulus. *Cryptology ePrint Archive, Report 2018/1009* (2018). <https://eprint.iacr.org/2018/1009>

17. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. *Deep Space Netw. Prog. Rep.* **44**, 114–116 (1978)
18. Melchor, C.A., et al.: Hamming quasi-cyclic (HQC). Technical report, National Institute of Standards and Technology 2017 (2018)
19. Misoczki, R., Tillich, J.P., Sendrier, N., Barreto, P.S.: MDPC-McEliece: new McEliece variants from moderate density parity-check codes. In: 2013 IEEE International Symposium on Information Theory Proceedings (ISIT), pp. 2069–2073. IEEE (2013)
20. Prange, E.: The use of information sets in decoding cyclic codes. *IRE Trans. Inf. Theory* **8**(5), 5–9 (1962)
21. Sendrier, N.: Decoding one out of many. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 51–67. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_4
22. Stern, J.: A method for finding codewords of small weight. In: Cohen, G., Wolfmann, J. (eds.) *Coding Theory 1988*. LNCS, vol. 388, pp. 106–113. Springer, Heidelberg (1989). <https://doi.org/10.1007/BFb0019850>
23. Canto Torres, R., Sendrier, N.: Analysis of information set decoding for a sub-linear error weight. In: Takagi, T. (ed.) PQCrypto 2016. LNCS, vol. 9606, pp. 144–161. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29360-8_10
24. Wafo-Tapa, G., Bettaieb, S., Bidoux, L., Gaborit, P.: A practicable timing attack against HQC and its countermeasure. *Cryptology ePrint Archive*, Report 2019/909 (2019). <https://eprint.iacr.org/2019/909>
25. Walters, M., Roy, S.S.: Constant-time BCH error-correcting code. *Cryptology ePrint Archive*, Report 2019/155 (2019). <https://eprint.iacr.org/2019/155>