A New Decryption Failure Attack Against HQC

Qian Guo[1,2(✉)] and Thomas Johansson[1(✉)]

[1] Department of Electrical and Information Technology, Lund University, P.O. Box 118, 221 00 Lund, Sweden
{qian.guo,thomas.johansson}@eit.lth.se
[2] Department of Informatics, University of Bergen, P.O. Box 7803, 5020 Bergen, Norway

Abstract. HQC is an IND-CCA2 KEM running for standardization in NIST's post-quantum cryptography project and has advanced to the second round. It is a code-based scheme in the class of public key encryptions, with given sets of parameters spanning NIST security strength 1, 3 and 5, corresponding to 128, 192 and 256 bits of classic security.

In this paper we present an attack recovering the secret key of an HQC instance named hqc-256-1. The attack requires a single precomputation performed once and then never again. The online attack on an HQC instance then submits about 2^{64} special ciphertexts for decryption (obtained from the precomputation) and a phase of analysis studies the subset of ciphertexts that are not correctly decrypted. In this phase, the secret key of the HQC instance is determined.

The overall complexity is estimated to be 2^{246} if the attacker balances the costs of precomputation and post-processing, thereby claiming a successful attack on hqc-256-1 in the NIST setting. If we allow the precomputation cost to be 2^{254}, which is below exhaustive key search on a 256 bit secret key, the computational complexity of the later parts can be no more than 2^{64}. This is a setting relevant to practical security since the large precomputation needs to be done only once. Also, we note that the complexity of the precomputation can be lower if the online attack is allowed to submit more than 2^{64} ciphertexts for decryption.

Keywords: Code-based cryptography · IND-CCA · NIST post-quantum standardization · Decryption errors · HQC · Reaction attack

1 Introduction

Integer factorization and the discrete logarithm problem have been cornerstone problems for asymmetric cryptography, but this is changing due to quantum computers, as their ability to efficiently solve such mathematical problems through Shor's algorithm [42] compromises the security of currently used asymmetric primitives. These developments have created the emergence of the area of

S. Moriai and H. Wang (Eds.): ASIACRYPT 2020, LNCS 12491, pp. 353–382, 2020.
https://doi.org/10.1007/978-3-030-64837-4_12

post-quantum cryptography and it motivated NIST to organize a post-quantum cryptography standardization project, with the ultimate goal of standardizing new quantum-resistant public-key crypto primitives. Submissions rely on problems from various fields within post-quantum cryptography, such as lattice-based, code-based and multivariate cryptography.

We specifically consider code-based cryptosystems. By this term, we mean cryptosystems where the algorithmic primitive uses an error correcting code. The primitive typically add an error to a codeword of the code and the primitive relies on the difficulty of decoding the received word back to the codeword. The first of those systems was a public key encryption scheme proposed by McEliece in 1978 [35]. The private key is a random binary irreducible Goppa code and the public key is a generator matrix of a randomly permuted and scrambled version of the original generator matrix for that code. The ciphertext is a codeword with some errors added, and only the knowledge of the private key (the Goppa code) can efficiently remove those errors. More formally, it is based on the difficulty of the syndrome decoding problem[1], which was proved to be NP-hard in [12]. After 40 years, some parameters have been adjusted, but no serious attack is known, even when using a quantum computer.

The birth of post-quantum cryptography made code-based cryptography a very interesting and the second most research-intense area after lattice-based crypto. Let us mention some recent code-based public key cryptosystems. The landmark paper presenting QC-MDPC [37] showed how the size of the public key could be made small, compared to the original McEliece scheme.

The different code-based proposals in the NIST process like NIST PQC candidates BIKE [5], LEDAcrypt [9], HQC [2], and others, showed that fully IND-CCA2 secure schemes could be built, using Fujisaki-Okamoto transform [22] or some similar conversion. They could also provide provable security in the sense that a proof of security related to a difficult decoding problem was given. The above mentioned schemes rely on decoding problems in the Hamming metric, whereas the schemes ROLLO [6] and RQC [3] rely on problems using the rank metric.

There are 17 remaining second-round candidates for public-key encryption and key-establishment algorithms in the NIST PQC project, among them six code-based schemes. The HQC submission [2] considered in this paper is such an IND-CCA2 KEM proposal running for standardization in NIST's post-quantum cryptography project and has advanced to the second round[2]. It is a code-based scheme in the class of public key encryptions, with given sets of parameters spanning NIST security strength 1, 3 and 5, corresponding to 128, 192 and 256 bits of classic security.

As for many of the code-based schemes (as well as for lattice-based schemes), there is no absolute guarantee that the decryption algorithm will succeed to decrypt to the transmitted message. Rather, there is a small probability of error

[1] A stronger hardness assumption in the average case is required.

[2] NIST announced the round-3 candidates in July 2020 and HQC is one of the eight alternate candidates.

for the decryption process, which for HQC is $<2^{-128}$ or even smaller. This work studies an attack that uses the possibility of having decryption errors as a part of the cryptanalysis process to finally determine the secret key.

1.1 Related Works

On code-based schemes without CCA2 conversion we have a few attacks in literature that require more than the CPA assumption. Using a *partially known plaintext attack* [16], one can reduce the code dimension in the decoding and thus achieve a lower complexity. In a *resend attack*, Alice resends the same message twice, or possible two related messages. The message can then be efficiently found [14]. A *reaction attack* [29] is a weaker version of a chosen ciphertext attack. The attacker sends a ciphertext or modifies an intercepted one and observes the reaction of the recipient (correct decryption or failure, but not the result of decryption). Again, one can in certain cases efficiently find the message corresponding to an intercepted ciphertext. Note that all these attacks are message recovery attacks.

In [26], an attack in the form of a reaction attack was given on the QC-MDPC scheme. The interesting fact for this attack was that it could be applied on the CCA version of the QC-MDPC scheme and still be successful; and it was a *key-recovery attack*. The journal version [27] expanded some details of the attack, e.g. the secret key recovery.

Following this work, many attacks on similar schemes were reported, for example on QC-LDPC [20], and attacks on LRPC [7,41]. All these attacks require that the decryption failure rate is fairly large, and subsequently the new schemes were designed with a much lower failure probability.

A similar development can be found for lattice-based schemes. For the lattice-based scheme NTRU (NTRUEncrypt) some problems due to decryption failures were identified already in [32,33]. More recently, several CCA type attacks using decryption failures on lattice-based schemes without CCA transforms has been reported, Fluhrer [21], Bernstein [13], on New-hope in CT-RSA 2019 [10], and mis-use attacks found in [8].

Attacking CCA secure lattice-based schemes through decryption failures in the spirit of [26] has been less successful. However, recently, CCA attacks based on failures were modeled and some initial attack attempts on an NTRU version were presented [17,25]. The most recent work in this direction is the attack on the LAC proposal [34] given in [28].

The proposed attack in this paper shares some similarities with these attacks in the sense that it uses a first precomputation phase to generate a set of encrypted messages for which the corresponding error vectors are causing the decryption failure probability to be much larger that the average case. The online attack then makes a statistical analysis of the information obtained from the ciphertexts that failed to decrypt and extracts enough information to recover the secret key.

There are also major differences, one being that HQC is a code-based scheme. Another major difference is that the LAC attack can only target weak secret keys

(e.g. one out of 2^{64} key pairs), whereas this attack on HQC targets any public key.

A relevant research direction is to investigate failure amplification tricks, including [38] in the code-based regime and [18] in the lattice-based regime. These techniques seem not directly to apply to attacking HQC.

1.2 Contributions

In this paper we present a CCA attack recovering the secret key of an HQC instance named hqc-256-1. The attack requires a single precomputation performed once and then never again. The online attack then submits about 2^{64} special ciphertexts for decryption (obtained from the precomputation) and a statistical analysis step processes information from the subset of ciphertexts that are not correctly decrypted. In this phase, the secret key of the HQC instance is determined. The overall complexity is estimated to be 2^{246} when the online decryption calls are limited to 2^{64}. With the given attack, this parameter choice hqc-256-1 can be attacked faster than exhaustive key search for a single key. One could also allow a large precomputation to reduce the post-processing cost since the single precomputation is performed only once. One example is to perform a precomputation of about 2^{254}, which is still below exhaustive key search on a 256 bit secret key. The computational complexity of the online and the post-processing steps is no more than the cost of submitting 2^{64} ciphertexts for decryption. Also, the complexity of the precomputation can be lower if the online attack is allowed to submit more than 2^{64} ciphertexts for decryption.

Last, we should note that once the precomputation is completed, the attack can be mounted on any HQC public keys when this cryptosystem is deployed. In this case, the precomputation complexity of the attack can be amortized. Therefore, several other parameter choices of HQC can be successfully targeted as well, in the sense that *all* the attacked keys can be recovered with complexity below the claimed security level. The amortized complexity is similar to Hellman's "cost per solution" [30], and the vulnerability comes from that the attacked HQC parameter sets cannot provide sufficient security compared with that of time-memory trade-off attacks on their block cipher counterpart such as AES. This attack model is not considered in the attacking framework in the NIST PQC standardization project. However, it could have high practical relevance since a scheme will be widely deployed if it becomes a standard. Time-memory trade-off attacks are relevant in practice and this attack model should be discussed also in the PKC area.

1.3 Organizations

The remaining of the paper is organized as follows. We briefly describe the HQC scheme in Sect. 2. Then we give a high-level description of the attack, explaining the basic underlying ideas in Sect. 3. In Sect. 4 we then give a more formal and detailed description of the attack and provide a theoretical basis for estimating the required complexity. Section 5 specifically considers the security of several

other HQC parameter sets. Section 6 discusses some aspects of the HQC scheme that are to the advantage of the attacker and related countermeasures. Section 7 concludes the paper.

2 Description of HQC

We briefly describe the HQC proposal [2] as submitted to the second round of the NIST post-quantum cryptography standardization process. HQC stands for Hamming Quasi-Cyclic and the underlying scheme was published in [36]. For more details we refer to the original design document [2] as we give only a brief description of the scheme. In particular, no description of the underlying difficult problems or the proofs of security are given in this section.

2.1 Notation

The scheme processes binary vectors of some length n, so such a vector $\mathbf{y} = (y_0, y_1, \dots y_{n-1}) \in \mathbb{F}_2^n$, where $y_i \in \mathbb{F}_2$, for $i = 0, 1, \dots, n-1$. By $\omega(\mathbf{y})$ we mean the Hamming weight of the vector $\mathbf{y}$, i.e., the number of nonzero coordinates. Since the field is $\mathbb{F}_2$, one can replace the operation of $-$ by $+$. Given a set $\mathcal{S}$, we use $\#\mathcal{S}$ to denote its cardinality.

Let $\mathcal{R} = \mathbb{F}_2[X]/(X^n - 1)$. An element $y(x) \in \mathcal{R}$ is a polynomial of degree at most $n-1$ expressed through the coefficients $y(x) = y_0 + y_1 x + \cdots + y_{n-1}x^{n-1}$. We will interchange between the expression of a vector $\mathbf{y}$ as a row vector and the corresponding polynomial $y(x)$. We may also write $\mathbf{y} \xleftarrow{\$} \mathcal{R}$, meaning that we randomly select a binary vector $\mathbf{y}$ also considered as a polynomial.

For two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}_2^n$ we define their product $\mathbf{u}\cdot\mathbf{v}$ as the coefficients of the polynomial $u(x)v(x) \in \mathcal{R}$. This product can also be expressed using circulant matrices. For a vector $\mathbf{y} \in \mathbb{F}_2^n$, the circulant matrix induced by $\mathbf{y}$ is denoted $\mathbf{rot}(\mathbf{y})$ and defined as

$$\mathbf{rot}(\mathbf{y}) = \begin{pmatrix} y_0 & y_{n-1} & \cdots & y_1 \\ y_1 & y_0 & \cdots & y_2 \\ \vdots & \vdots & \ddots & \vdots \\ y_{n-1} & y_{n-2} & \cdots & y_0 \end{pmatrix}.$$

The multiplication $\mathbf{uv}$ can now alternatively be written as

$$\mathbf{uv} = \mathbf{u} \cdot \mathbf{rot}(\mathbf{v})^T = \mathbf{v} \cdot \mathbf{rot}(\mathbf{u})^T,$$

where $(\cdot)^T$ denotes transpose.

We give some basic definitions and properties from coding theory and refer to [2] or any textbook on the subject. A linear code $\mathcal{C}$ of length n and dimension k (an $[n, k]$ code) is a subspace of $\mathbb{F}_2^n$ of dimension k. A matrix $\mathbf{G} \in \mathbb{F}_2^{\mathbf{k}\times\mathbf{n}}$ is a generator matrix of the code if

$$\mathcal{C} = \{\mathbf{mG}, \mathbf{m} \in \mathbb{F}_2^k\}.$$

A matrix $\mathbf{H} \in \mathbb{F}_{\mathbf{2}}^{(\mathbf{n}-\mathbf{k})\times\mathbf{n}}$ is a parity check matrix of the code if

$$\mathcal{C} = \{\mathbf{v} \in \mathbb{F}_2^n, \text{such that } \mathbf{vH^T} = \mathbf{0}\}.$$

Fix a parity check matrix and let $\mathbf{v} \in \mathbb{F}_{\mathbf{2}}^{\mathbf{n}}$. Then the syndrome of $\mathbf{v}$ is the value $\mathbf{vH^T}$. If $\mathbf{v} \in \mathcal{C}$ then the syndrome is $\mathbf{0}$. The minimum distance d of the code is the minimum weight taken over all nonzero codewords in the code. Such a code is capable of correcting $\delta = \lfloor \frac{d-1}{2} \rfloor$ errors. This means that if a codeword is disturbed by adding a binary vector known to be of weight at most δ, then there is an algorithm that can find and remove this noise and return the codeword.

A repetition code, denoted $\mathbb{1}_n$, is an $[n, 1]$ code that has a generator matrix of the form $\mathbf{G} = [\mathbf{1}]$. It means transmitting a single bit and repeating it n times. Such a code can then correct up to $\delta_r = \lfloor \frac{n-1}{2} \rfloor$ errors. BCH codes are a very common class of codes as they achieve a good error correction capability. We do not need to define them, but rather just note that there is an efficient algorithm for correcting errors. By $BCH(n_1, k_1, \delta)$ we denote a BCH code that is capable of correcting up to δ errors.

Finally, a tensor product code $\mathcal{C}$, denoted $\mathcal{C}_1 \otimes \mathcal{C}_2$, is a code built from two codes $\mathcal{C}_1$ and $\mathcal{C}_2$. If $\mathcal{C}_1$ is an $[n_1, k_1, d_1]$ code and $\mathcal{C}_2$ is an $[n_2, k_2, d_2]$ code then the tensor product code $\mathcal{C}$ is an $[n_1 n_2, k_1 k_2, d_1 d_2]$ code. You can view the length $n_1 n_2$ codewords in $\mathcal{C}_1 \otimes \mathcal{C}_2$ as a $n_1 \times n_2$ array, where every row is a codeword in $\mathcal{C}_1$ and every column is a codeword in $\mathcal{C}_1$. We will only be concerned with the construction $\text{BCH}(n_1, k, \delta) \otimes \mathbb{1}_{n_2}$. It means that every position in the BCH codeword is repeated n_2 times. The decoding procedure first decodes every repetition code by simply counting the number of zeros (or ones). Then the output is used as the value for each position in the BCH code, which is then decoded. In particular, if we want to find an error that will not decode correctly, we need to have at least $\delta_r = \lfloor \frac{n_2+1}{2} \rfloor$ errors in each of at least $\delta + 1$ different columns (repetition codes).

2.2 The HQC Scheme

The public key encryption (PKE) version of HQC is shown in Fig. 1. HQC makes use of the tensor product code of two different codes, one being a BCH code and the other being a simple repetition code. The code is denoted by $\mathcal{C}$ and has a corresponding generator matrix $\mathbf{G} \in \mathbb{F}_2^{k\times n}$. We return to the details of the tensor product code later.

The scheme follows the steps of many lattice-based schemes, but here errors are considered in the Hamming metric. The key generation randomly selects a public $\mathbf{h} \in \mathcal{R}$ and two private vectors $\mathbf{x}, \mathbf{y} \in \mathcal{R}$ with very low Hamming weight. It computes $\mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y}$ as the second part of the public key, which presumably looks like a randomly chosen vector.

In the encryption one generates noise $\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2 \in \mathcal{R}$ with very low Hamming weight and computes $\mathbf{u} = \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2$ and $\mathbf{v} = \mathbf{mG} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e}$, which is returned as the ciphertext.

In decryption, one has access to the secret key $\mathbf{y}$ and computes $\mathbf{v} - \mathbf{u} \cdot \mathbf{y}$. The decoder for the code then finally removes all the noise. An expansion of the expression shows

$$\mathbf{v} - \mathbf{u} \cdot \mathbf{y} = \mathbf{mG} + (\mathbf{x} + \mathbf{h} \cdot \mathbf{y}) \cdot \mathbf{r}_2 + \mathbf{e} - (\mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2) \cdot \mathbf{y} = \mathbf{mG} + \mathbf{e}',$$

where

$$\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{e}. \tag{1}$$

Throughout the paper, we denote the i-th entry of $\mathbf{e}$ ($\mathbf{e}'$) by e_i (e'_i). If the Hamming weight of $\mathbf{e}'$ is not too large, the decoder will be able to decode and return the correct message $\mathbf{m}$. As all parts of the expression for $\mathbf{e}'$ are of very low weight, also $\mathbf{e}'$ will be of somewhat low weight.

- $\mathsf{Setup}(1^\lambda)$: generates the global parameters $\mathsf{param} = (n, k, \delta, \omega, \omega_\mathbf{r}, \omega_\mathbf{e})$.
- $\mathsf{KeyGen}(\mathsf{param})$: sample $\mathbf{h} \xleftarrow{\$} \mathcal{R}$, the generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ of $\mathcal{C}$, $\mathsf{sk} = (\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{R}^2$ such that $\omega(\mathbf{x}) = \omega(\mathbf{y}) = \omega$, sets $\mathsf{pk} = (\mathbf{h}, \mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y})$, and returns $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Encrypt}(\mathsf{pk}, \mathbf{m})$: generates $\mathbf{e} \xleftarrow{\$} \mathcal{R}$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathcal{R}^2$ such that $\omega(\mathbf{e}) = \omega_\mathbf{e}$ and $\omega(\mathbf{r}_1) = \omega(\mathbf{r}_2) = \omega_\mathbf{r}$, sets $\mathbf{u} = \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2$ and $\mathbf{v} = \mathbf{mG} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e}$, returns $\mathbf{c} = (\mathbf{u}, \mathbf{v})$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$: returns $\mathcal{C}.\mathsf{Decode}(\mathbf{v} - \mathbf{u} \cdot \mathbf{y})$.

Fig. 1. Description of the proposal HQC.PKE [2].

A transform [31] is then applied to HQC.PKE to achieve IND-CCA2 for the KEM/DEM version of HQC (see HQC.KEM in Fig. 2). The KEM version can be converted to an IND-CCA2 PKE using generic transforms.

In the description, $\mathcal{G}, \mathcal{H}$ and $\mathcal{K}$ are different hash functions, e.g. SHA512. Also, $\mathcal{E}$ denotes the IND-CPA secure HQC.PKE primitive including randomness input.

The noise term written as $\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{e}$ is a sparse vector, but not extremely sparse. The decryption is guaranteed to succeed if $\omega(\mathbf{e}')$ is an error that is within the decoding capability of the employed error-correcting code. So the code must be very powerful and be able to correct many errors. In the HQC scheme, the error correcting code with generator matrix $\mathbf{G}$ is the tensor product code $\mathcal{C} = \mathrm{BCH}(n_1, k, \delta) \otimes \mathbb{1}_{n_2}$. This is a powerful code employed to reduce the decryption failure probability. It guarantees to correct any error of weight up to $\delta \cdot (n_2 + 1)/2$, but will most likely also correct errors of somewhat higher weight.

2.3 Parameter Settings

The proposed parameters of different instances of HQC are shown in Table 1. The parameter n is a prime number slightly larger than $n_1 \times n_2$, ω is the weight of the secret and $\omega_\mathbf{r}$, $\omega_\mathbf{e}$ is the weight of the noise vectors.

- $\mathsf{Setup}(1^\lambda)$: generates the global parameters $\mathsf{param} = (n, k, \delta, \omega, \omega_{\mathbf{r}}, \omega_{\mathbf{e}})$. here k will be the length of the symmetric key being exchanged, typically $k = 256$.
- $\mathsf{KeyGen}(\mathsf{param})$: exactly as in HQC.PKE.
- $\mathsf{Encapsulate}(\mathsf{pk})$: generate $\mathbf{m} \xleftarrow{\$} \mathbb{F}_2^k$, which will serve as the seed to derive the shared key. Derive the randomness $\theta \xleftarrow{\$} \mathcal{G}(\mathbf{m})$. Generate the ciphertext $\mathbf{c} \leftarrow (\mathbf{u}, \mathbf{v}) = \mathcal{E}.\mathsf{Encrypt}(\mathsf{pk}, \mathbf{m}, \theta)$, and derive the symmetric key $K \leftarrow \mathcal{K}(\mathbf{m}, \mathbf{c})$. Let $\mathbf{d} \leftarrow \mathcal{H}(\mathbf{m})$, and send $(\mathbf{c}, \mathbf{d})$.
- $\mathsf{Decapsulate}(\mathsf{sk}, \mathbf{c}, \mathbf{d})$: decrypt $\mathbf{m}' \leftarrow \mathcal{E}.\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$, compute $\theta' \leftarrow \mathcal{G}(\mathbf{m}')$, and (re-)encrypt $\mathbf{m}'$ to get $\mathbf{c}' \leftarrow \mathcal{E}.\mathsf{Encrypt}(\mathsf{pk}, \mathbf{m}', \theta')$. If $\mathbf{c} \neq \mathbf{c}'$ or $\mathbf{d} \neq \mathcal{H}(\mathbf{m}')$ then abort. Otherwise, derive the shared key $K \leftarrow \mathcal{K}(\mathbf{m}, \mathbf{c})$.

Fig. 2. Description of the proposal HQC.KEM [2].

Table 1. Proposed parameters for HQC.

	n_1	n_2	n	k	δ	ω	$\omega_{\mathbf{r}} = \omega_{\mathbf{e}}$	Security	p_{fail}
hqc-128-1	796	31	24,677	256	60	67	77	128	$<2^{-128}$
hqc-192-1	766	57	43,669	256	57	101	117	192	$<2^{-128}$
hqc-192-2	766	61	46,747	256	57	101	117	192	$<2^{-192}$
hqc-256-1	766	83	63,587	256	57	133	153	256	$<2^{-128}$
hqc-256-2	796	85	67,699	256	60	133	153	256	$<2^{-192}$
hqc-256-3	796	89	70,853	256	60	133	153	256	$<2^{-256}$

3 Basic Ideas for the Attack

In this section we try to describe the underlying ideas behind the attack and the detailed analysis is done in the following sections. The first step is to find out how we can produce decryption failures and to do that we need to study the details of the decoding procedure. The scheme uses the tensor product code $\mathcal{C} = \text{BCH}(n_1, k, \delta) \otimes \mathbb{1}_{n_2}$ which means that the received vector can be split in n_1 parts each of length n_2. The decoding is done in two steps. First, each subvector of length n_2 corresponding to a repetition code is decoded to a single bit $\{0, 1\}$. This leaves a length n_1 vector which is decoded through a decoder for the BCH code to correct up to δ errors. This means that in order to have an overall decoding error, one has to get at least $\delta + 1$ of the repetition codes to make an individual decoding error. Such an individual error appears if the noise $\mathbf{e}'$ has more ones than zeros in the n_2 positions corresponding to that particular repetition code.

Let us now look at a typical error pattern $\mathbf{e}'$ to be decoded. Since the noise term can be written as $\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{e}$, almost all nonzero contribution comes from the two product terms. If we for example use the parameters for hqc-256-1 then $\mathbf{x}$ has weight 133, $\mathbf{r}_2$ has weight 153, and the product between them will have weight close to $133 \cdot 153$, say about weight 20000. Adding another

Fig. 3. How decoding is split in two parts.

such error contribution from $\mathbf{y} \cdot \mathbf{r}_1$ of the similar weight results in a typical error vector of weight around 27000 and length 63587. The ones in $\mathbf{e}'$ are distributed roughly with the same probability for all positions thus also roughly with the same probability in each repetition code (Fig. 3).

So how can we increase the overall probability of a decoding error? The idea is that we use a lot of precomputation to test many different messages and look at what error vectors they create (through the hash functions that gives the error vectors in the CCA version). The problem is to figure out what kind of error vectors will increase the overall probability of a decoding error. One answer to this problem is to consider the set of vectors where we have a lot of ones close together in the vector. More specifically, we keep and store only messages for which the generated $\mathbf{r}_2$ vector contains an interval (chunk) of length l_1 containing many ones, say at least l_0 ones. An example of a parameter choice might be $l_1 = 55$ and $l_0 = 38$. Now, let us look at the contribution of $\mathbf{x} \cdot \mathbf{r}_2$ to the error $\mathbf{e}'$. So $\mathbf{x}$ has weight 133, meaning that the result of $\mathbf{x} \cdot \mathbf{r}_2$ is the sum of 133 different rotated versions of $\mathbf{r}_2$. The average distance between ones in $\mathbf{x}$ is almost 500. This means that length l_1 intervals of many ones in the 133 rotated versions of $\mathbf{r}_2$ almost never coincide in positions, but leave (almost) 133 intervals of many ones in the result of $\mathbf{x} \cdot \mathbf{r}_2$. In more detail, a single interval of many ones may either end up completely inside a single repetition code or it will contribute in two adjacent repetition codes. This depends on the exact starting position of the interval of many ones as well as the positions of the ones in the secret $\mathbf{x}$. This is all depicted in Fig. 4, where we can see the top interval of many ones (illustrated as a box) affecting two adjacent repetition codes whereas the second top interval in the figure affects only a single repetition code.

In any case, the result is that the ones in $\mathbf{e}'$ are no longer uniformly distributed, but some repetition codes will have many errors and others very few. Concentrating the errors to a subset of the repetition codes drastically changes the overall probability of a decoding error. Whereas an average error vector has probability much smaller than 2^{-128} of not being correctly decoded, errors of the above form with $l_1 = 55$ and $l_0 = 38$ show an overall decoding error probability of 2^{-14}!

We now assume that we have observed a number of decryption errors by feeding the decryption oracle these special messages and recording their corresponding error vectors generated in the encryption. The second task in the attack is to recover some part of the secret key $\mathbf{x}, \mathbf{y}$. We do that by the following observation. We consider a position i and look at whether it is likely that

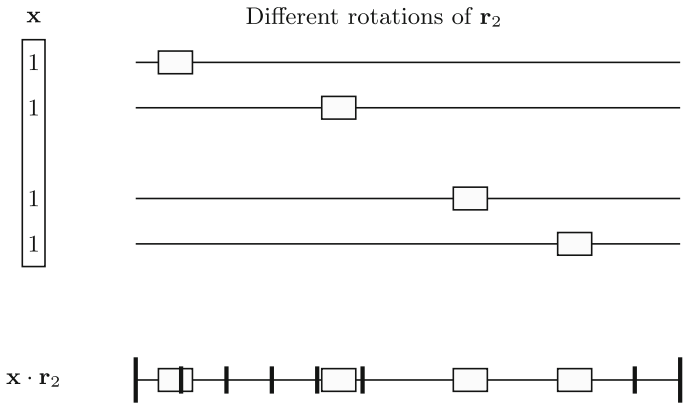

Fig. 4. The appearance of ones in $\mathbf{x} \cdot \mathbf{r}_2$ when $\mathbf{r}_2$ contains an interval of many ones.

$x_i = 1$. If this is the case and an interval of many ones starts at position j for a particular $\mathbf{r}_2$, then we will have a contribution of many ones starting at position $i + j$. Since we got a decryption error it is much more likely that the repetition codes corresponding to position $i + j$ and l_1 positions onwards, will decode in error, compared to the general case.

Now, since the overall error is $\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{e}$ another observation is that if a repetition code is not decoding correctly, it is likely that the independent noise part $\mathbf{e}$ is "helping out to make a decoding error". If the contribution from the $\mathbf{x} \cdot \mathbf{r}_2$ term for $x_i = 1$ gives a chunk of many ones in a repetition code that is assumed not to decoding correctly, then the corresponding $\mathbf{e}$ values will be more likely to be zero if there is already a one contributed from the interval of many ones. Similarly, the corresponding $\mathbf{e}$ values will be more likely to be one if there is a zero contribution from the interval of many ones, or if the position is outside the interval.

These two observations put together gives us a strategy of examining the Hamming weight of the given $\mathbf{e}$ vector in the repetition codes corresponding to position $i + j$. Basically, if the Hamming weight of these parts taken over many different such $\mathbf{e}$ vectors is following the above observation, then we can come to the conclusion that $x_i = 1$, otherwise we set $x_i = 0$. We come back to the details of this procedure in the next section.

There is some dependence for positions that are closely located, so it is actually a good approach to only establish many positions for which we have $x_i = 0$. Then we can use an information set decoding algorithm to solve for $\mathbf{x}, \mathbf{y}$ using the knowledge from the public key. The complexity of this procedure has very limited complexity.

4 Attack Model and Detailed Steps

We follow the newly-proposed attack using decryption failures adopting the weak-ciphertext attack model from [17], although no assumption on weak keys is used. The attack consists of three main steps. Firstly, we prepare 'weak' ciphertexts with the noise tuple $(\mathbf{r_1}, \mathbf{r_2}, \mathbf{e})$ of a specific form. We then submit these ciphertexts to the decryption oracle and collect the decryption errors that occur. The last step is to perform a statistical analysis (e.g., hypothesis testing) to extract the key information. We will also include classical solving algorithms in code-based cryptography like Information Set Decoding (ISD) to improve the key-recovery efficiency.

4.1 Weak-Ciphertext Preparation

We select a set (denoted $\mathcal{A}$) of 'weak' ciphertexts and the corresponding noise tuples $(\mathbf{r_1}, \mathbf{r_2}, \mathbf{e})$ with $\mathbf{r_2}$ having a consecutive l_1 positions with at least l_0 ones in this chunk. Let A_i denote the event that this consecutive chunk exists and starts from the i-th position. We estimate the probability of this event as

$$\Pr[A_i] = \frac{\binom{w_r}{l_0} \cdot \binom{n-w_r}{l_1-l_0}}{\binom{n}{l_1}}.$$

The overall probability of finding such a chunk in a length n vector, denoted as p, can then be estimated as

$$p = \Pr[\cup_i A_i] \approx \sum_i \Pr[A_i] - \sum_{i,j} \Pr[A_i \cap A_j]. \tag{2}$$

Thus, we expect that we need p^{-1} computations (hash calls to $\mathcal{G}$) to generate one 'weak' ciphertext with the chosen form. For hqc-256-1, as shown in the second column of Table 2, the precomputation costs differ for different choices of l_1 and l_0. This cost is bounded by 2^{191} if we set $l_1 = 53$ and $l_0 = 29$. Note that even though the precomputation is searching a larger space than allowed in the scheme, if we set $l_1 = 55$ and $l_0 \geq 36$, we include and discuss these parameter choices for the purpose of simulation of parts of the attack.

4.2 Collecting Errors

We send the selected ciphertexts in set $\mathcal{A}$ to the decryption oracle for decryption and store the tuple $(\mathbf{r_1}, \mathbf{r_2}, \mathbf{e})$ for ciphertexts leading to a decryption error. An important task is to estimate the decryption error rate for the chosen parameters.

The Convolution of Probability Distributions. Let $X_i = 1$ be the event that decoding output of the i-th repetition code is erroneous, for $i \in \{0, \ldots, n_1 - 1\}$. We denote this event by p_i, i.e., $p_i = \Pr[X_i = 1]$, and thus, the probability $\Pr[X_i = 0]$ is $(1 - p_i)$. Let $\mathcal{D}$ denote the event that the tuple $(\mathbf{r_1}, \mathbf{r_2}, \mathbf{e})$ leads to

Table 2. The estimated decryption error rate (DER) for hqc-256-1. The starting position is 0.

l_1	l_0	$\log_2(p^{-1})$	The DER (in $\log_2(\cdot)$) in estimation
55	38	276	−14.72
55	37	266	−17.65
55	36	256	−20.84
53	29	191	−48.61
63	30	191	−48.03
66	30	188	−49.17
55	26	163	−63.91
45	16	86	−112.6

a decryption error. If we assume that all the X_i's are independent[3], then we can recursively estimate the probability of $\mathcal{D}$ as

$$\Pr[\sum_{i=0}^{n_1-1} X_i > \delta] = p_{n_1-1} \cdot \Pr[\sum_{i=0}^{n_1-2} X_i \geq \delta] + (1 - p_{n_1-1}) \cdot \Pr[\sum_{i=0}^{n_1-2} X_i > \delta]. \quad (3)$$

This method is referred to as the convolution of probability distributions.

The computed decryption error probabilities (DER) for different choices of l_1 and l_0 are shown in the third column of Table 2. We see that for the simulation purpose, the decryption error probability can be as low as 2^{-14} if $l_1 = 55$ and $l_0 = 38$; for a theoretical attack, the decryption error probability is estimated to be $2^{-48.61}$ if $l_1 = 53$ and $l_0 = 29$. The p_i's can be computed and also tested in simulation. The exact calculation of the p_i's for a given error pattern in $\mathcal{A}$ is considered later in the section. In practice, we run a large number of trials to empirically test the values of p_i and then use the convolution of probability distributions to compute the decryption error probabilities (DER). In a later part (see Table 5) we report on testing the accuracy of this estimation approach and obtained simulation results are close to the estimation. This estimation approach is actually a bit conservative from the attacker's viewpoint.

4.3 Statistical Analysis

After collecting all the tuples $(\mathbf{r_1}, \mathbf{r_2}, \mathbf{e})$ that lead to a decryption error, we attempt to recover partial secret key information on $\mathbf{x}$. In this part, we first present the empirical statistical dependence observed for a key recovery attack

[3] The independence assumption will lead to a conservative estimation from the attacker's viewpoint. The reason is this assumption can cause a DER estimation smaller than the true value, which fits the results [19,28] on LAC and has been verified by simulation in Table 5.

and develop theoretical estimations on this dependence. The developed theory nicely explains the observed distinguishing property and shows that the property is even stronger when the DER drops. We also discuss techniques and tricks employed in this statistical test step.

Observation. Assume for simplicity that all chunks of many ones in $\mathbf{r_2}$ start in position 0. Our approach to recover partial secret key information is to investigate and compute the frequency of $\{e_i = 1|\mathcal{D}\}$, where $0 \leq i \leq n_1 n_2 - 1$ and $\mathcal{D}$ means that the tuple $(\mathbf{r_1}, \mathbf{r_2}, \mathbf{e})$ leads to a decryption error.

The basic observation is that if $x_{i_0} = 1$, then with high probability (due to the sparsity of $\mathbf{x}$), only one nonzero entry is in the interval $[n_2 \cdot t, n_2 \cdot (t+1))$, for $t = \lfloor i_0/n_2 \rfloor$. Then $\#\{e_i = 1|\mathcal{D}\}$ for $i \in [i_0, i_0 + l_1]$ is smaller than for the values corresponding to the other positions in the interval $[n_2 \cdot t, n_2 \cdot (t+1))$. We can observe no difference (i.e., behave like the random) for an interval $\mathcal{I}$ corresponding to a repetition code without an i_0 such that $i_0 \in \mathcal{I}$ and $x_{i_0} = 1$. We refer to the prior as *CASE I* and the latter as *CASE II*.

The controlled window of length l_1 can be divided into two halves and contributes to two consecutive but different repetition decoding intervals. In this case we can still observe these (possibly weaker) frequency differences.

Visual Illustration from Experiments. This phenomenon is visually illustrated in Fig. 5, where the simulation results for targeting the hqc-256-1 parameters by setting $l_1 = 55$ and $l_0 = 38$ are provided. The DER is simulated to be $2^{-14.3}$. We select at random two CASE I repetition code intervals and two CASE II repetition code intervals, which are plotted in the top two sub-figures and the bottom two sub-figures in Fig. 5, respectively. In the first two plots we re-order the positions by putting the length l_1 controlled window at the beginning. We then derive four length $n_2 - l_1$ vectors with entry i being the summation of l_1 consecutive positions starting from i in the re-ordered repetition code intervals. One can in this way visually observe the differences of CASE I and CASE II in the repetition code intervals.

This figure shows that one can recover key information if a sufficient number of decryption failures are provided. The above presented distinguisher, however, is far from optimal. We will now present a better maximum likelihood distinguisher that will also give an estimate of the required number of decryption failures.

Theoretical Estimation on $\Pr[e_j = 1|\mathcal{D}]$**.** Let us examine $\Pr[e_j = 1|\mathcal{D}]$ for $0 \leq j \leq n_1 n_2 - 1$ through

$$\begin{aligned}\Pr[e_j = 1|\mathcal{D}] =& \Pr[X_i = 1|\mathcal{D}]\Pr[e_j = 1|X_i = 1, \mathcal{D}] \\ &+ \Pr[X_i = 0|\mathcal{D}]\Pr[e_j = 1|X_i = 0, \mathcal{D}],\end{aligned}$$

where $X_i = 1$ represents the event that the i-th repetition decoding is erroneous, and vice versa, for $i = \lfloor j/n_2 \rfloor$. We denote the i-th repetition code interval as $\mathcal{I}_i$, where $\mathcal{I}_i = [i \cdot n_2, (i+1) \cdot n_2)$.

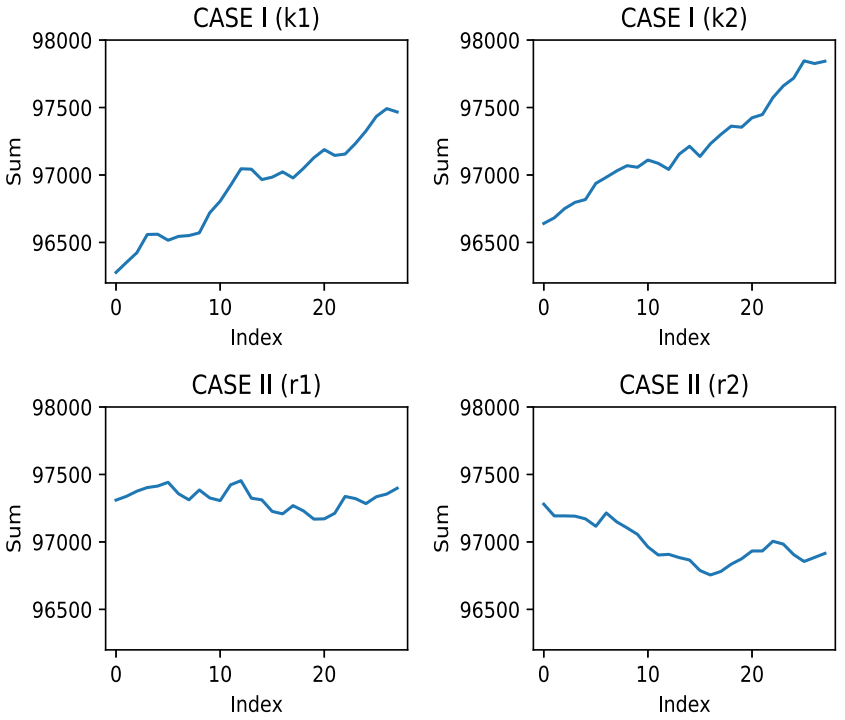

Fig. 5. The sum of l_1 consecutive $\#\{e_i = 1|\mathcal{D}\}$ in a repetition decoding interval with re-ordering. The top two sub-figures plot CASE I intervals and the last two figures plot the opposite case. 735,000 decryption errors are collected with DER $2^{-14.3}$. The CASE I and CASE II intervals could be distinguished.

We assume that the events $(e_j = 1|X_i = 1)$ and $(\mathcal{D}|X_i = 1)$ are independent, as e_j roughly only depends on X_i and not any other X_j for $j \neq i$. Then we derive that

$$\Pr[e_j = 1|X_i = 1, \mathcal{D}] = \frac{\Pr[e_j = 1, \mathcal{D}|X_i = 1]}{\Pr[\mathcal{D}|X_i = 1]} \approx \Pr[e_j = 1|X_i = 1],$$

Also, $\Pr[e_j = 1|X_i = 0, \mathcal{D}] \approx \Pr[e_j = 1|X_i = 0]$. Then we can rewrite as

$$\begin{aligned}\Pr[e_j = 1|\mathcal{D}] \approx & \Pr[X_i = 1|\mathcal{D}]\Pr[e_j = 1|X_i = 1] \\ & + \Pr[X_i = 0|\mathcal{D}]\Pr[e_j = 1|X_i = 0] \\ = & \Pr[X_i = 1|\mathcal{D}](\Pr[e_j = 1|X_i = 1] - \Pr[e_j = 1|X_i = 0]) \\ & + \Pr[e_j = 1|X_i = 0].\end{aligned}$$

We note that

$$\Pr[X_i = 1|\mathcal{D}] = \frac{\Pr[X_i = 1, \mathcal{D}]}{\Pr[\mathcal{D}]} = \frac{p_i \cdot \Pr[\sum_{j \neq i} X_j \geq \delta]}{\Pr[\mathcal{D}]}.$$

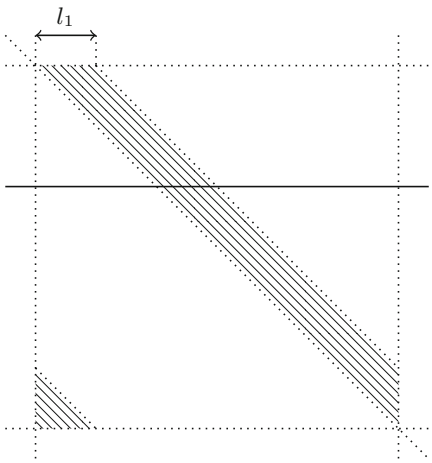

Fig. 6. The matrix representation of $\mathbf{r}_1$ in case that the controlled interval starts from Position 0.

We also know that

$$\begin{aligned}\Pr[e_j = 1, X_i = 1] &= \Pr[e_j = 1, \sum_{k\in\mathcal{I}_i} e'_k > \delta_r] \\ &= \Pr[e_j = 1, e'_j = 1] \cdot \Pr[\sum_{k\neq j, k\in\mathcal{I}_i} e'_k > \delta_r - 1] \\ &\quad + \Pr[e_j = 1, e'_j = 0] \cdot \Pr[\sum_{k\neq j, k\in\mathcal{I}_i} e'_k > \delta_r]\end{aligned}$$

and

$$\Pr[e_j = 1 | X_i = 1] = \frac{\Pr[e_j = 1, X_i = 1]}{\Pr[e_j = 1, X_i = 1] + \Pr[e_j = 0, X_i = 1]}. \tag{4}$$

We can analogously compute $\Pr[e_j = 1 | X_i = 0]$ as

$$\Pr[e_j = 1 | X_i = 0] = \frac{\Pr[e_j = 1] - \Pr[e_j = 1, X_i = 1]}{1 - \Pr[X_i = 1]}, \tag{5}$$

where $\Pr[e_j = 1] = \omega_{\mathbf{e}}/n$.

Thus, all the intermediate probability values can be obtained via recursively computing the convolution of probability distributions if p_i, $\Pr[e_j = 1, e'_j = 1]$, $\Pr[e_j = 1, e'_j = 0]$, and $\Pr[e'_k = 1]$ are known. We next show how to estimate p_i, $\Pr[e_j = 1, e'_j = 1]$, $\Pr[e_j = 1, e'_j = 0]$, and $\Pr[e'_k = 1]$.

Computation of Probabilities p_i: Firstly, the probability p_i can also be computed using the convolution of probability distributions of $\Pr[e_j = 1] : j \in \mathcal{I}_i$, but a simpler strategy is to test the values experimentally, since these probability

Table 3. The computed $\Pr[X_i = 1|\mathcal{D}]$ from the simulation results on hqc-256-1 with $l_1 = 53$ and $l_0 = 29$.

l'	Index	$\Pr[X_i = 1\|\mathcal{D}]$
0	25	0.00506
	71	0.00504
	166	0.00504
	250	0.00502
	365	0.00500
	590	0.00506
53	68	0.3089
	127	0.3087
	186	0.3089
	356	0.3089
	482	0.3089
	695	0.3089

values are relatively significant. An important observation shown in Table 3 is that the computed $\Pr[X_i = 1|\mathcal{D}]$ is a function of the length l' of the controlled window in the repetition code interval. Here $l' = 0$ corresponds to the random case and $l' = l_1$ corresponds to having a full length l_1 chunk. Intermediate values appear when a chunk of many ones is split in two different repetition code intervals. In this table, the computed $\Pr[X_i = 1|\mathcal{D}]$ from the simulation results regarding the hqc-256-1 parameters, where $l_1 = 53$ and $l_0 = 29$ and the controlled interval starting form position 0. We choose at random 6 repetition code intervals with controlled length 0 and repetition code intervals with the full controlled length 53. The second column shows the index of the selected repetition code interval and the last column shows the computed probability $\Pr[X_i = 1|\mathcal{D}]$. We see the computed probabilities are close in the same group and have a huge gap between the two groups with different l'.

Computation of Remaining Probabilities: Secondly, with assumptions of certain independence, if the contribution from the marked strip in Fig. 6 is correctly guessed, then one can apply similar arguments as Proposition 1.4.1 in [2] to derive closed formulas to estimate the probabilities $\Pr[e'_k = 1]$, $\Pr[e_j = 1, e'_j = 1]$, and $\Pr[e_j = 1, e'_j = 0]$.

Proposition 1 (Proposition 1.4.1 in [2]). *Let* $\mathbf{x} = (X_0, \ldots, X_{n-1})$ *(resp.* $\mathbf{r} = (R_0, \ldots, R_{n-1})$*) be a random vector where* X_i *(resp.* R_i*) are independent Bernoulli variables of parameters* p *(resp.* $p_{\mathbf{r}}$*),* $\Pr[X_i = 1] = p$ *and* $\Pr[R_i = 1] = p_{\mathbf{r}}$*. Assuming* $\mathbf{x}$ *and* $\mathbf{r}$ *are independent, and denoting* $\mathbf{z} = \mathbf{x} \cdot \mathbf{r} = (Z_0, \ldots, Z_{n-1})$ *as defined as the multiplication of* $\mathbf{x}$ *and* $\mathbf{r}$ *in* $\mathcal{R}$*, we have*

$$\hat{p} = \Pr[Z_k = 1] = \frac{1}{2} - \frac{1}{2}(1 - 2pp_{\mathbf{r}})^n. \tag{6}$$

Table 4. The simulated probabilities v.s. the estimated probabilities on hqc-256-1. The starting position is 0. The distinguishing property is stronger when the DER level drops.

l_1	l_0	In estimation			In simulation			The DER level
		p_{random}	p_{high}	p_{low}	p_{random}	p_{high}	p_{low}	(in $\log_2(\cdot)$)
55	38	0.002406	0.002412	0.002401	0.002404	0.002460	0.002347	-14
55	36	0.002406	0.002419	0.002393	0.002404	0.002468	0.002340	-20
53	29	0.002408	0.002446	0.002369	–	–	–	-48

Thus, the contribution of $\mathbf{y}\cdot\mathbf{r}_2$ can be modeled as a Bernoulli random variable with probability $\hat{p}$. For a position j, if we guess the sub-vector $\mathbf{x}_{\mathsf{part}}$ of $\mathbf{x}$ corresponding to the controlled interval of length l_1 shown in the marked strip, its contribution denoted Υ_{part} to e'_j is then known. Let ω_{part} be the weight of $\mathbf{x}_{\mathsf{part}}$, we then model the position in the remaining sub-vector of $\mathbf{x}$ as a Bernoulli random variable with probability $\frac{\omega-\omega_{\mathsf{part}}}{n-l_1}$. We model the position in the unmarked part of $\mathbf{r}_1$ as a Bernoulli random variable with probability $\frac{\omega_{\mathbf{r}}-l_0}{n-l1}$. The contribution of $\mathbf{x}\cdot\mathbf{r}_1 - \Upsilon_{\mathsf{part}}$ is modeled as a Bernoulli random variable with probability $\tilde{p}$, where

$$\tilde{p} = \frac{1}{2} - \frac{1}{2}(1 - 2\cdot\frac{\omega-\omega_{\mathsf{part}}}{n-l_1}\cdot\frac{\omega_{\mathbf{r}}-l_0}{n-l_1})^{n-l_1}.$$

We derive the following proposition.

Proposition 2. *We have that,*

$$\Pr[e'_k - \Upsilon_{\mathsf{part}} = 0, e_k = 1] = \frac{\omega_{\mathbf{e}}}{n}\cdot(\hat{p}\cdot(1-\tilde{p}) + (1-\hat{p})\cdot\tilde{p}), \tag{7}$$

$$\Pr[e'_k - \Upsilon_{\mathsf{part}} = 1, e_k = 1] = \frac{\omega_{\mathbf{e}}}{n}\cdot(\hat{p}\cdot\tilde{p} + (1-\hat{p})\cdot(1-\tilde{p})), \tag{8}$$

$$\Pr[e'_k - \Upsilon_{\mathsf{part}} = 1, e_k = 0] = \left(1 - \frac{\omega_{\mathbf{e}}}{n}\right)\cdot(\hat{p}\cdot(1-\tilde{p}) + (1-\hat{p})\cdot\tilde{p}), \tag{9}$$

and

$$\Pr[e'_k - \Upsilon_{\mathsf{part}} = 0, e_k = 0] = \left(1 - \frac{\omega_{\mathbf{e}}}{n}\right)\cdot(\hat{p}\cdot\tilde{p} + (1-\hat{p})\cdot(1-\tilde{p})). \tag{10}$$

Putting all the formulas together, we can compute the probability $\Pr[e_j = 1|\mathcal{D}]$ in the different cases. The distinguishing property can then be depicted as in Fig. 7. We plotted two repetition decoding intervals, a CASE II interval at the right with a probability of $\Pr[e_j = 1|\mathcal{D}]$ denoted by p_{random} and a CASE I interval at the left. For the latter, for the position outside a window of length l_1 corresponding to the controlled section, we know that the contribution of Υ_{part} is always 0, thus showing a higher probability of $\Pr[e_j = 1|\mathcal{D}]$ than p_{random} denoted by p_{high}; otherwise, the contribution Υ_{part} can be either 0 or 1, showing a higher or lower probability of $\Pr[e_j = 1|\mathcal{D}]$, respectively. We denote the lower probability of $\Pr[e_j = 1|\mathcal{D}]$ by p_{low}.

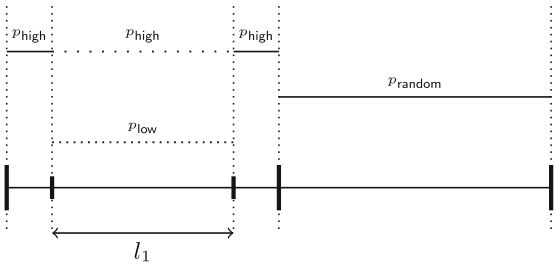

Fig. 7. Graphic illustration of the distinguishing property.

Example 1: We apply the theoretical analysis on hqc-256-1 with results as shown in Table 4. When the starting position is 0, $l_1 = 55$ and $l_0 = 38$, for a CASE II repetition decoding window, we estimate that $\Pr[e_j = 1|\mathcal{D}]$ is 0.002406 (p_{random}); for a CASE I repetition decoding window, we estimate that $\Pr[e_j = 1|\mathcal{D}]$ is 0.002401 (p_{low}) for the case $\Upsilon_{\mathsf{part}} = 1$, and $\Pr[e_j = 1|\mathcal{D}]$ is 0.002412 (p_{high}) for the case $\Upsilon_{\mathsf{part}} = 0$. The difference (or called bias) between p_{high} and p_{low} becomes larger for parameters with lower estimated decryption error rates (say $l_0 = 29$).

The remaining problem is to correctly guess the contribution Υ_{part} from the marked strip, which is a hypothesis testing problem revealing partial information on the secret key $\mathbf{x}$.

Maximum Likelihood Ratio Test. We describe the maximum likelihood ratio test to correctly guess the contribution Υ_{part}. We guess a small chunk of $\mathbf{x}$ and then obtain the corresponding Υ_{part}. We then group the positions according to its guessed contribution Υ_{part} and to decide if the resulted distribution is more close to the theoretically derived one, i.e., with a high $\Pr[e_j = 1|\mathcal{D}]$ for $\Upsilon_{\mathsf{part}} = 0$ and withe a low $\Pr[e_j = 1|\mathcal{D}]$ for $\Upsilon_{\mathsf{part}} = 1$, or more close to the random case (i.e. CASE II).

In the implementation, we test bit-by-bit, i.e., running through all the positions of $\mathbf{x}$ and for each position, using maximum likelihood test to decide if that position should be zero, under the assumption that the rest positions are all zero. This approach is definitely the simplest strategy, and is demonstrated to be efficient in simulation. There exist many other possible approaches for this statistical test step.

Double Distinguishing. In the previous discussion, we prepare the weak ciphertext set $\mathcal{A}$ whose $\mathbf{r_2}$ part has a consecutive l_1 positions with l_0 ones, to recover partial information of $\mathbf{x}$. Also, we can prepare another set (denoted $\mathcal{A}$') of weak ciphertexts whose $\mathbf{r_1}$ part has a consecutive l_1 positions with l_0 ones, to recover partial information of $\mathbf{y}$. Due to the 2^{64} constraint on the number of the submitted ciphertexts, we let $\mathcal{A}$' and $\mathcal{A}$ be both of size 2^{63}. One call to the encryption function will return both the $\mathbf{r_1}$ and $\mathbf{r_2}$ parts. Thus, if 2^{γ} encryptions

are required to detect a ciphertext in $\mathcal{A}$ (or $\mathcal{A}'$) with the desired pattern, we only need $2^{\gamma+63}$ encryption calls in total to prepare the two sets.

4.4 Information Set Decoding

This step will be applied if no full key-recovery is achieved after the statistical analysis. The basic idea is that, with the partial information obtained from the statistical analysis step, recovering the secret key $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$ from the public key $\mathsf{pk} = (\mathbf{h}, \mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y})$ is a simple task.

To be more specific, if one knows from the previous step the l positions in $\mathbf{x}$ and also l positions in $\mathbf{y}$ that are zero with the highest probability, then a new decoding problem with code length $2l$ and dimension n can be derived. This problem can be much easier to solve as the chosen positions are rarely non-zero.

The plain ISD [40] algorithm works very well if the guessed positions are reliable. This process could possibly be accelerated by applying advanced information set decoding algorithms like Stern's algorithm [43] and BJMM [11]. In addition, the previous hypothesis testing step provides soft information of the secret key, so the ISD variant with bit reliability [39] and the soft-Stern algorithm [23,24] could be employed to accelerate this post-processing step.

4.5 Simulation Results on hqc-256-1

We in this part present implementation results on hqc-256-1. In the later analysis we will stick with the empirical distributions from the largest simulation we conducted. This analysis is conservative from an attacker's viewpoint, since the distinguishing property is stronger when the DER drops.

The Tested Bias. We have tested the different probabilities of p_{high}, p_{low}, and p_{random}, computed in the previous parts. The comparison between the theoretical results and the simulated results is shown in Table 4. We could see that for a fixed l_1 (here $l_1 = 55$), when the value of l_0 drops (meaning that the precomputation cost drops and also the decryption error probability), the differences (also named bias) between two of the probabilities increase. Most importantly, the simulated bias is much larger than then estimated bias. For instance, we observed a gap of 0.000055 between p_{high} and p_{random}, while only of 0.000006 in the theoretical estimation, when setting $l_1 = 55$ and $l_0 = 38$.

This table (Table 4) verifies our theoretical analysis, i.e., explaining the reason of observing the different probabilities in CASE I and CASE II, though the real bias is much larger than the one computed. Due to the crypt-analytic nature of this work, we want to upper-bound the complexity of the newly proposed attack. In the later analysis, thus, we make a conservative choice by employing the empirical distributions obtained from the largest simulation we conducted. Note that these chosen probabilities ($p_{\mathsf{random}}, p_{\mathsf{high}}, p_{\mathsf{low}}$) correspond to the case that the decryption error probability is about 2^{-22}. Though the true bias when launching the attack for parameters (of $l_1 = 53$ and $l_0 = 29$) with decryption

Table 5. The simulated DER v.s. the estimated DER (with starting position 0).

l_1	l_0	The DER (in $\log_2(\cdot)$) in estimation	The DER (in $\log_2(\cdot)$) in simulation
55	38	-14.72	-14.3
55	37	-17.65	-17.2
55	36	-20.84	-20.3

error probability of about 2^{-48} is beyond our capability to simulate, it will be larger than the one in our simulation.

The Accuracy of the Convolution Method. We have tested the accuracy of the estimation using the convolution of probability distributions as shown in Table 5. The estimation is always slightly lower than the simulated decryption error rate, caused by the weak dependence between different positions. We see that the ratio between the simulated DER and the estimated DER becomes larger when the error rate is smaller, and it is already $2^{0.54}$ when the DER is about 2^{-20}. We expect to have the ratio to be 2 when considering the case that the DER is smaller than 2^{-48}.

Estimation from Different Starting Positions. We have observed that the estimated decryption error rates fluctuate slightly if the starting position differs. The mean over all starting positions is $2^{-48.98}$ for $l_1 = 53$ and $l_0 = 29$. Since the estimation is slightly conservative, we could expect to collect $2^{64-49+1} = 2^{16}$ decryption errors in practice.

A Full Test. We run simulation to demonstrate that 2^{16} decryption errors, i.e., 2^{15} decryption errors for each test when employing the double distinguishing procedure, are enough for a full-key recovery with complexity bounded by the 2^{64} online decryption oracle queries. As discussed before, if one launches the attack with decryption error probability of about 2^{-48}, the bias will be larger and lead to a reduced attack complexity.

In the simulation, we pick $l_1 = 40$ and $l_0 = 34$ leading to a simulated decryption error probability of $2^{-21.8}$. As it is slightly smaller than $2^{-20.3}$, the simulated decryption error probability when setting $l_1 = 55$ and $l_0 = 36$, it is reasonable to have the bias of the simulated probabilities $(p_{\mathsf{random}}, p_{\mathsf{high}}, p_{\mathsf{low}}) = (0.2404, 0.2467, 0.2333)$ also slightly larger. We then roughly compute the divergence, which is $2^{-13.17}$, between the probability distributions over an interval of length 83 in CASE I and CASE II. Since we only need to detect positions that are highly probable to be 0 and the noise vector is very sparse, the sample number 2^{15} is theoretically large enough.

We test the bits of the secret vector $\mathbf{x}$ and assume that it works for $\mathbf{y}$ as well since the double distinguishing procedure just repeats the same test. To

be specific, we run $2^{36.8}$ encryptions using randomly generated ciphertexts with the $\mathbf{r}_2$ part having the desired pattern. We then obtain the likelihood that one position of $\mathbf{x}$ is zero by the bit-by-bit test. It is interesting to see that in the 46875 most reliable positions that are decided to be zero, only 22 positions are actually one, and these positions all have a non-zero neighbor which falsifies the assumption of our test, i.e., only one bit is one and the rest are all zero. A test can be more efficient if it can handle in a smarter manner the close-by non-zero pairs in the secret.

Since only one position error occurs in the 16056 most reliable positions, we could have 14500 error-free positions effortlessly. Assuming that we have the same test result for the $\mathbf{y}$ vector, we then need to solve a new decoding problem with code-length 64750, dimension 35587, and the error weight 44. This new problem can be solved with complexity 2^{50} Gaussian Eliminations, by using the plain ISD algorithm. The overall complexity of the post-processing can be bounded by 2^{57} Gaussian Eliminations, which is negligible compared with the cost of the online decryption oracle calls.

We also test that the post-processing complexity increases to 2^{165} Gaussian Eliminations if the number of decryption errors is reduced by a factor of 6.

4.6 Summarizing the Complexity of Attacking hqc-256-1

We summarize the attack on hqc-256-1. Firstly, if the attacker chooses the attaching parameters $l_1 = 53$ and $l_0 = 29$, then he will obtain 2^{16} decryption errors after sending out 2^{64} decryption oracles calls, i.e., 2^{15} errors for each test when the double distinguishing procedure are applied. We then in simulation test that this amount of decryption errors are sufficient for a full key recovery.

Note that in the above analysis, we employ the bias tested empirically from simulations with decryption error rate of only 2^{-22}, which is stronger in the attacking scenario with decryption error rate of 2^{-48}. Unlike the precomputation phase and the online decryption phase, the statistical analysis phase and the post-processing phase using the ISD algorithms require computational costs that are sensitive to the bias. Though the current complexity number claimed is already below the cost of the online phase, the true cost should be even smaller since it drops drastically if the bias increases. This trend has been well verified in the simulation.

We conclude that 2^{16} decryption errors are enough to correct a number of entries in $\mathbf{x}$ and $\mathbf{y}$ so that the full key can be recovered in a later ISD step with complexity negligible to the 2^{64} online decryption requests. Thus, the complexity of this CCA attack on hqc-256-1 can be estimated as 2^{64} online decryption requests, after one large precomputation of 2^{254}.

If we remove the constraint that an attack can only submit 2^{64} ciphertexts for decryption, the precomputation cost can be lower. The precomputation cost for one ciphertext and its corresponding decryption error rate are shown in Table 2. Since the bias will be even stronger when the DER becomes smaller, 2^{16} decryption errors are more than sufficient in these parameter settings. If an attacker is allowed to submit 2^{80} decryption requests, then the precomputation

Table 6. The trade-off between the precomputation cost and the number of online decryption oracle queries for hqc-256-1. We assume that 2^{16} decryption failures are sufficient for a full-key recovery.

l_1	l_0	Online DOQs (in $\log_2(\cdot)$)	Precomputation cost (in $\log_2(\cdot)$)
53	29	64	254
55	26	80	242
45	16	128	213

cost can drop to 2^{242} when setting $l_1 = 55$ and $l_0 = 26$. The cost even drops to 2^{213} if 2^{128} ciphertexts are allowed to be decrypted online. We summarize the trade-off between the precomputation cost and the number of online decryption oracle queries (DOQs) in Table 6, when assuming that 2^{16} decryption failures are sufficient for a full-key recovery.

When the Overall Complexity Includes the Precomputation Cost. In the previous analysis, we ensure the complexity of the online querying and the post-processing to be bounded by the 2^{64} decryption oracle calls. This cost also bounds the whole attacking complexity since the large precomputation only needs to be done once and therefore should not be included in the attacking complexity of one particular attack. We now consider a different optimization goal to optimize the complexity including the precomputation cost. Thus, the attacker could employ a heavier post-processing to reduce the precomputation cost.

As stated in Sect. 4.5, we tested that one could reduce the required number of decryption errors by a factor of 6 and still kept the post-processing complexity far below 2^{220}. We then select $(l_1, l_0) = (61, 29)$ and estimate the DER to be $2^{-51.5}$ by the convolution method. We obtain enough decryption errors if allowing 2^{64} decryption oracle calls, and the corresponding precomputation cost is 2^{248} due to the double distinguishing procedure.

In Table 4, we see that the simulated bias is larger than the estimated bias. We can extrapolate the distribution when the DER is close to 2^{-50} if a simple model where p_{random} is almost unchanged and $p_{\mathsf{T}}^{\mathsf{sim}} \approx p_{\mathsf{T}}^{\mathsf{esti}} + \delta$ are adopted. Here $\mathsf{T} \in \{\mathsf{high}, \mathsf{low}\}$, $p_{\mathsf{T}}^{\mathsf{sim}}$ means the simulated p_{T} and $p_{\mathsf{T}}^{\mathsf{esti}}$ means the estimated p_{T}. By this simple model, we could select $(l_1, l_0) = (63, 29)$ and give a sharper estimation that the precomputation cost of 2^{246} could be enough for a full key-recovery.

Comparing with AES256 in the TMTO Model. We emphasize the practical relevance of ensuring the complexity in the real attacking phase to be bounded by the 2^{64} online decryption oracle queries. In this case, a large precomputation is done and the real-time complexity T and the memory constraint M can be set to be 2^{64}. Therefore, by splitting the cost into the precomputation and the online parts, a natural model to compare with is the famous Hellman's

Table 7. The estimated decryption failure probability.

	l_1	l_0	$\log_2(p^{-1})$	The DER (in $\log_2(\cdot)$) in estimation
hqc-128-1	20	16	110	-46.25
hqc-192-1	41	24	157	-47.56
hqc-192-2	50	28	187	-48.36

time-memory trade-off (TMTO) attack on block ciphers [30]. It is well-known that the trade-off relationship for Hellman's TMTO attack is

$$TM^2 = 2^{2\lambda},$$

where T is the time complexity, M is the memory complexity, and λ is the claimed security level. It can be easily checked that, for our attack on hqc-256-1, the value of TM^2 is much smaller than 2^{512}.

NIST classified the range of the security strengths from symmetric cryptographic primitives like AES (see [1]). We conclude, also from the perspective of time-memory trade-off attacks, that hqc-256-1 cannot provide sufficient security compared with block ciphers such as AES256. Similar discussions on several other HQC parameter sets will be presented in the next section.

5 On Other HQC Parameters

Comparing with AES in the TMTO Model. We adopt the assumption that 2^{16} decryption errors – for the case that the decryption error probability is close to 2^{-48} – are sufficient for recovering the key with complexity negligible to the 2^{64} online decryption oracle calls, as we discussed in the previous section. This assumption for hqc-192-1 was verified in simulation. Table 7 shows the estimated DER from the convolution of probability distributions for three HQC parameter sets, hqc-128-1, hqc-192-1, and hqc-192-2. For all the three parameter sets, it is possible to attack with complexity bounded by 2^{64} online decryption oracle calls. We can set the time and memory constraints to be 2^{64} and the new attacks are much better than Hellman's TMTO relationship for a block cipher.

Thus, we similarly claim that all the three parameter sets cannot provide sufficient security regarding the TMTO attacks.

When the Precomputation Can Be Amortized. The precomputation is excluded from the time complexity constraint T in the TMTO model. We could consider a different model of attacking more keys and amortizing the precomputation complexity. In other words, the precomputation is still done once and never again; the attacker then use the precomputed weak ciphertexts to attack all the public keys once the HQC cryptosystem is deployed. If HQC is used for a very long period, we could assume that the attacker can have 2^{64} public keys

to target. His goal is to recover all the keys with complexity below $2^{64+\lambda}$, where λ is the claimed security level. If so, the amortized complexity is below 2^{λ} for recovering one key. Moreover, the memory cost for storing the precomputed noise seeds is also amortized.

The idea of attacking many keys for a lower RAM and precomputation cost than attacking each key separately was originally discussed by Hellman, also in his seminal work [30]. Note that this model is different from the usual multi-target model where only one key among the 2^{64} keys needs to be recovered, whose counterpart for block ciphers (e.g., AES) is discussed as time/memory/data trade-off attacks in [15] for recovering one out of many possible keys.

The previous section shows that hqc-256-1 can be solved with 2^{254} precomputation, which can be amortized to 2^{190} for recovering 2^{64} keys in the new model. Due to the double distinguishing procedure, we see from Table 7 that the precomputation costs for hqc-128-1, hqc-192-1 and hqc-192-2 are $2^{110+63=173}$, 2^{220}, and 2^{250}, respectively. If 2^{64} keys are attacked in this model, the amortized precomputation complexity can be estimated as 2^{109}, 2^{156}, and 2^{186}, respectively, each of which is below its claimed security level. If one makes an extreme assumption that there is an infinite number of HQC keys to attack, then the complexity per key is only 2^{64}, dominated by that of the online attacking phase.

Experiments. We tested our assumption for hqc-192-1 in simulation that 2^{16} decryption errors are sufficient in our attack scenario. For the ease of simulation, we also select parameters leading to a DER close to 2^{-20}. Firstly, we have detected a stronger bias for hqc-192-1, e.g., the absolution value of $(p_{\mathsf{high}} - p_{\mathsf{random}})$ is almost twice as large as the simulated value for hqc-256-1. We then collected 2^{15} decryption errors and performed a full test similar to the version described in Sect. 4.5. As expected, the experimental results are even better, and the post-processing complexity is estimated to be less than 2^{49} Gaussian Eliminations using the plain ISD.

We also tested the accuracy of the convolution method by running $2^{37.3}$ encryptions. The simulated DER is $2^{-19.29}$, larger than the estimated DER of $2^{-20.55}$ by a factor of about $2^{1.3}$. In this sense, our attack complexity estimation is conservative and the actual complexity could be slightly lower.

6 Discussion and Countermeasures

In [4], NIST commented on HQC that "HQC presents a strong argument that its decryption failure rate is low enough to obtain chosen ciphertext security. This is the strongest argument, at present, of CCA security among the second-round candidate code-based cryptosystems, where information set decoding is the limiting attack for both private key recovery and message recovery (BIKE, HQC, and LEDAcrypt)." The newly proposed attack partly proves and disproves these comments at least for certain parameter sets.

From one perspective, we do not falsify the designers' claim on the DER analysis (for instance, we do not break their claim that the averaged decryption error

probability (DER) is below 2^{-128} for hqc-256-1)[4]. Our experiments to test the accuracy of the convolution method can be treated as a partial verification that the error-correcting implementation in HQC matches its theoretical estimation, i.e., the correlation among positions is much weaker (than that in LAC [19,28]).

On the other hand, our attack shows that the (CCA) security claim of HQC is problematic. In the NIST setting – with only 2^{64} online chosen ciphertext submission and only one public key is assumed – the proposed parameters of hqc-256-1 with DER below 2^{-128} is insufficient to ensure the claimed CCA security level. One should consider parameter settings with even lower DER. The problem becomes even more severe in the model where more keys are assumed and the precomputation cost can be amortized.

Secure Parameters. The current attack version does not affect the security claim of hqc-256-2 and hqc-256-3.

Protection. The attacks on hqc-128-1, hqc-192-1 and hqc-192-2 with amortized precomputation cost can be thwarted by employing the multi-target protection technique, i.e., using $\mathcal{G}(pk, m)$ rather than only $\mathcal{G}(m)$ where $\mathcal{G}$ is a hash function, in the random noise generation process. We suggest the designers to include this technique in a later version. For the highest security level, we suggest to use the other parameter settings, hqc-256-2 and hqc-256-3, that are invulnerable to the new attacks.

Not a Weak-Key Attack. Note that the current attack version includes no weak-key analysis, which has been proven a big threat [28] to the LAC [34] scheme. We leave this problem for further research.

More Discussions on the Attack Models. In this paper, we have discussed the attack with three different models, i.e., the NIST setting, the model that the precomputation cost can be amortized, and the TMTO model that the precomputation cost is excluded from the trade-off formula. The last two models are of practical relevance.

To be specific, the precomputation cost is done only once and is done *before* the adversary gets a target to attack. It can be used for all targets in the future. Later the adversary may get one or many targets to attack and the online complexity to find the key is only 2^{64}. Also, the targeted keys can be in different HQC systems. We see that the second model and the TMTO model are not the same as a multitarget attack model in general.

[4] Actually, by computing the convolution of distributions, we estimate the decryption error rates for the selected keys in simulation to be 2^{-163} for hqc-192-1 and 2^{-151} for hqc-256-1, which supports their official claim of DER smaller than 2^{-128}. Since the gap between the two estimation method (2^{-151} v.s. 2^{-128}) is large, it may be too conservative to only multiply a factor of 2 to adjust DER estimation value in Sect. 4.5.

The TMTO attacks are not considered in the current NIST PQC framework but could be considered relevant because if precomputation is for free, a TMTO attack is the best known attack on all versions of HQC (and other NIST candidates), by attacking the random seed to public key setup. In such a case the NIST security definition would relate to a generic TMTO attack instead of a simple exhaustive key search.

It is stated in the NIST submission requirements that "Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 128-bit key (e.g. AES128)", and NIST goes on "any attack must require computational resources comparable to or greater than the stated threshold, with respect to ALL metrics that NIST deems to be potentially relevant to practical security". Similar requirements are explicitly stated for other security levels. Thus, if NIST considers the TMTO attack model to be a relevant metric, then hqc-128-1, hqc-192-1, and hqc-192-2 are all affected.

7 Concluding Remarks

We have presented a new CCA attack on the HQC proposal that has advanced to the second round in the NIST post-quantum cryptography standardization project. For hqc-256-1, the secret key can be recovered with complexity 2^{248} estimated by using simulation data (or 2^{246} using an extrapolation model), if only 2^{64} online decryption oracle calls are submitted. This analysis questions the security claim of hqc-256-1 in the NIST setting. Moreover, we could bound the online and post-processing complexity to 2^{64} online decryption oracle calls, if a large precomputation of 2^{254} is included. The precomputation cost needs to be done only once and can be smaller if more decryption oracle calls are allowed. We also presented attacks on hqc-128-1, hqc-192-1, and hqc-192-2 with amortized precomputation complexity below the claimed security levels, respectively, if multiple keys are assumed. Compared with AES in the TMTO attack model, all the four parameter sets, hqc-128-1, hqc-192-1, hqc-192-2, and hqc-256-1 are insufficient for providing security w.r.t. the claimed security levels. There are safe parameter choices like hqc-256-2 and hqc-256-3.

We present this attack version to demonstrate the vulnerability of the broken HQC parameter settings; the attack, however, can be further optimized. Firstly, employing better ISD algorithms with bit reliability like the Soft-Stern [23,24] will allow reduced post-processing complexity. Secondly, performing larger implementation will lead to a more accurate complexity estimation with a smaller complexity number, as the bias will be stronger if the error rate becomes even lower. Thirdly, a more sophisticated distinguisher better dealing with the close-by ones would reduce the required number of errors, thereby reducing the attack complexity further. Last, one could design a more powerful weak-key version similar to the attack [28] on LAC, which only works for a small fraction of keys.

It is interesting to apply similar ideas to other proposals in the NIST PQC project, e.g., the rank-based proposal Rollo [6]. This attacking approach can be

generally applied to code-based primitives using the tensor product codes built from BCH codes and the repetition codes. This error-correcting scheme is also employed in Lepton [44], a round-1 candidate in the NIST PQC project. The attack may need to be adjusted when considering other coding implementations, which would be interesting for future research.

Acknowledgments. The authors would like to thank the anonymous reviewers from Asiacrypt 2020 for their helpful comments. This work was supported in part by the Swedish Research Council (Grant No. 2019-04166), by the Swedish Foundation for Strategic Research (Grant No. RIT17-0005), and by the Norwegian Research Council (Grant No. 247742/070). The computations/simulations were performed on resources provided by UNINETT Sigma2 - the National Infrastructure for High Performance Computing and Data Storage in Norway, and by Swedish National Infrastructure for Computing.

References

1. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf. Accessed 05 Feb 2020
2. Aguilar Melchor, C., et al.: HQC. Technical report. National Institute of Standards and Technology (2019). https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
3. Aguilar Melchor, C., et al.: RQC. Technical report. National Institute of Standards and Technology (2019). https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
4. Alagic, G., et al.: Status report on the first round of the NIST post-quantum cryptography standardization process (2019). https://doi.org/10.6028/NIST.IR.8240
5. Aragon, N., et al.: BIKE. Technical report. National Institute of Standards and Technology (2019). https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
6. Aragon, N., et al.: ROLLO. Technical report. National Institute of Standards and Technology (2019). https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
7. Aragon, N., Gaborit, P.: A key recovery attack against LRPC using decryption failures. In: Coding and Cryptography, International Workshop, WCC, vol. 2019 (2019)
8. Băetu, C., Durak, F.B., Huguenin-Dumittan, L., Talayhan, A., Vaudenay, S.: Misuse attacks on post-quantum cryptosystems. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. Misuse attacks on post-quantum cryptosystems, vol. 11477, pp. 747–776. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_26
9. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: LEDAcrypt. Technical report. National Institute of Standards and Technology (2019). https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
10. Bauer, A., Gilbert, H., Renault, G., Rossi, M.: Assessment of the key-reuse resilience of NewHope. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 272–292. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12612-4_14

11. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: how $1 + 1 = 0$ improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 520–536. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_31
12. Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.A.: On the inherent intractability of certain coding problems (corresp.). IEEE Trans. Inf. Theory **24**(3), 384–386 (1978). https://doi.org/10.1109/TIT.1978.1055873
13. Bernstein, D.J., Bruinderink, L.G., Lange, T., Panny, L.: HILA5 pindakaas: on the CCA security of lattice-based encryption with error correction. Cryptology ePrint Archive, Report 2017/1214 (2017). https://eprint.iacr.org/2017/1214
14. Berson, T.A.: Failure of the McEliece public-key cryptosystem under message-resend and related-message attack. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 213–220. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052237
15. Biryukov, A., Mukhopadhyay, S., Sarkar, P.: Improved time-memory trade-offs with multiple data. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 110–127. Springer, Heidelberg (2006). https://doi.org/10.1007/11693383_8
16. Canteaut, A., Sendrier, N.: Cryptanalysis of the original McEliece cryptosystem. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 187–199. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-49649-1_16
17. D'Anvers, J.-P., Guo, Q., Johansson, T., Nilsson, A., Vercauteren, F., Verbauwhede, I.: Decryption failure attacks on IND-CCA secure lattice-based schemes. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 565–598. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17259-6_19
18. D'Anvers, J.-P., Rossi, M., Virdia, F.: *(One) Failure Is Not an Option*: bootstrapping the search for failures in lattice-based encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12107, pp. 3–33. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45727-3_1
19. D'Anvers, J.-P., Vercauteren, F., Verbauwhede, I.: The impact of error dependencies on Ring/Mod-LWE/LWR based schemes. In: Ding, J., Steinwandt, R. (eds.) PQCrypto 2019. LNCS, vol. 11505, pp. 103–115. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25510-7_6
20. 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
21. Fluhrer, S.: Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085 (2016). http://eprint.iacr.org/2016/085
22. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_34
23. Guo, Q., Johansson, T., Mårtensson, E., Stankovski, P.: Information set decoding with soft information and some cryptographic applications. In: 2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, 25–30 June 2017, pp. 1793–1797 (2017). https://doi.org/10.1109/ISIT.2017.8006838
24. Guo, Q., Johansson, T., Mårtensson, E., Wagner, P.S.: Some cryptanalytic and coding-theoretic applications of a soft stern algorithm. Adv. Math. Commun. **13**(4), 559–578 (2019). https://doi.org/10.3934/amc.2019035

25. Guo, Q., Johansson, T., Nilsson, A.: A generic attack on lattice-based schemes using decryption errors with application to ss-ntru-pke. IACR Cryptology ePrint Archive 2019, 43 (2019). https://eprint.iacr.org/2019/043
26. 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
27. Guo, Q., Johansson, T., Wagner, P.S.: A key recovery reaction attack on QC-MDPC. IEEE Trans. Inf. Theory **65**(3), 1845–1861 (2019). https://doi.org/10.1109/TIT.2018.2877458
28. Guo, Q., Johansson, T., Yang, J.: A novel CCA attack using decryption errors against LAC. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11921, pp. 82–111. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_4
29. Hall, C., Goldberg, I., Schneier, B.: Reaction attacks against several public-key cryptosystem. In: Varadharajan, V., Mu, Y. (eds.) ICICS 1999. LNCS, vol. 1726, pp. 2–12. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-540-47942-0_2
30. Hellman, M.E.: A cryptanalytic time-memory trade-off. IEEE Trans. Inf. Theory **26**(4), 401–406 (1980). https://doi.org/10.1109/TIT.1980.1056220
31. 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
32. Howgrave-Graham, N., et al.: The impact of decryption failures on the security of NTRU encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 226–246. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_14
33. Howgrave-Graham, N., Silverman, J.H., Singer, A., Whyte, W.: NAEP: provable security in the presence of decryption failures. Cryptology ePrint Archive, Report 2003/172 (2003). http://eprint.iacr.org/2003/172
34. Lu, X., et al.: LAC. Technical report. National Institute of Standards and Technology (2019). https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
35. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. JPL DSN Prog. Rep. **44**, 114–116 (1978)
36. Melchor, C.A., Blazy, O., Deneuville, J., Gaborit, P., Zémor, G.: Efficient encryption from random quasi-cyclic codes. IEEE Trans. Inf. Theory **64**(5), 3927–3943 (2018). https://doi.org/10.1109/TIT.2018.2804444
37. Misoczki, R., Tillich, J., Sendrier, N., Barreto, P.S.L.M.: MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In: Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, 7–12 July 2013, pp. 2069–2073. IEEE (2013). https://doi.org/10.1109/ISIT.2013.6620590
38. Nilsson, A., Johansson, T., Wagner, P.S.: Error amplification in code-based cryptography. IACR Tran. Cryptogr. Hardw. Embed. Syst. **2019**, 238–258 (2019)
39. Pessl, P., Mangard, S.: Enhancing side-channel analysis of binary-field multiplication with bit reliability. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 255–270. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_15
40. Prange, E.: The use of information sets in decoding cyclic codes. IRE Trans. Inf. Theory **8**(5), 5–9 (1962). https://doi.org/10.1109/TIT.1962.1057777

41. Samardjiska, S., Santini, P., Persichetti, E., Banegas, G.: A reaction attack against cryptosystems based on LRPC codes. In: Schwabe, P., Thériault, N. (eds.) LATINCRYPT 2019. LNCS, vol. 11774, pp. 197–216. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30530-7_10
42. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science, pp. 124–134, 20–22 November 1994. IEEE Computer Society Press, Santa Fe, NM, USA (1994)
43. 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
44. Yu, Y., Zhang, J.: Lepton. Technical report. National Institute of Standards and Technology (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions