Decryption Failure Attacks on IND-CCA Secure Lattice-Based Schemes

Jan-Pieter D'Anvers[1(✉)], Qian Guo[2,3], Thomas Johansson[3], Alexander Nilsson[3], Frederik Vercauteren[1], and Ingrid Verbauwhede[1]

[1] imec-COSIC, KU Leuven, Kasteelpark Arenberg 10, Bus 2452, 3001 Leuven-Heverlee, Belgium
{janpieter.danvers,frederik.vercauteren, ingrid.verbauwhede}@esat.kuleuven.be

[2] Department of Informatics, University of Bergen, Box 7803, 5020 Bergen, Norway
qian.guo@uib.no

[3] Department of Electrical and Information Technology, Lund University, P.O. Box 118, 221 00 Lund, Sweden
{thomas.johansson,alexander.nilsson}@eit.lth.se

Abstract. In this paper we investigate the impact of decryption failures on the chosen-ciphertext security of lattice-based primitives. We discuss a generic framework for secret key recovery based on decryption failures and present an attack on the NIST Post-Quantum Proposal ss-ntru-pke. Our framework is split in three parts: First, we use a technique to increase the failure rate of lattice-based schemes called failure boosting. Based on this technique we investigate the minimal effort for an adversary to obtain a failure in three cases: when he has access to a quantum computer, when he mounts a multi-target attack or when he can only perform a limited number of oracle queries. Secondly, we examine the amount of information that an adversary can derive from failing ciphertexts. Finally, these techniques are combined in an overall analysis of the security of lattice based schemes under a decryption failure attack. We show that an attacker could significantly reduce the security of lattice based schemes that have a relatively high failure rate. However, for most of the NIST Post-Quantum Proposals, the number of required oracle queries is above practical limits. Furthermore, a new generic weak-key (multi-target) model on lattice-based schemes, which can be viewed as a variant of the previous framework, is proposed. This model further takes into consideration the weak-key phenomenon that a small fraction of keys can have much larger decoding error probability for ciphertexts with certain key-related properties. We apply this model and present an attack in detail on the NIST Post-Quantum Proposal – ss-ntru-pke – with complexity below the claimed security level.

Keywords: Lattice-based cryptography · NIST post-quantum standardization · Decryption failure · LWE · NTRU · Reaction attack

This paper is the result of a merge of [14] and [21].

D. Lin and K. Sako (Eds.): PKC 2019, LNCS 11443, pp. 565–598, 2019.
https://doi.org/10.1007/978-3-030-17259-6_19

1 Introduction

The position of integer factorization and the discrete logarithm problem as a cornerstone for asymmetric cryptography is being threatened by quantum computers, as their ability to efficiently solve these mathematical problems compromises the security of current asymmetric primitives. These developments have led to the emergence of post-quantum cryptography and motivated NIST to organize a post-quantum cryptography standardization process, with the goal of standardizing one or more quantum-resistant public-key cryptographic primitives. Submissions originate from various fields within post-quantum cryptography, such as lattice-based, code-based and multivariate cryptography.

Lattice-based cryptography has recently developed into one of the main research areas in post-quantum cryptography. Lattice-based submissions to the NIST Post-Quantum process can be broadly put into one of two categories: NTRU-based schemes (e.g. [39,47]) and schemes based on the learning with errors (LWE) hard problem [36]. A lot of research has been done on their security, such as provable post-quantum secure transformations from IND-CPA to IND-CCA secure schemes [25,29,38,46], security estimates of post-quantum primitives [3,4] and provable reductions for various hard problems underlying the schemes [7,11,32,35,36].

A striking observation is that numerous proposed Key Encapsulation Mechanisms (KEM's) have a small failure probability during decryption, in which the involved parties fail to derive a shared secret key. This is the case for the majority of schemes based on lattices, codes or Mersenne primes. The probability of such failure varies from 2^{-64} in Ramstake [45] to 2^{-216} in New Hope [41], with most of the failure probabilities lying around 2^{-128}. As this failure is dependent on the secret key, it might leak secret information to an adversary. However, reducing this probability has a price, as the parameters should be adjusted accordingly, resulting in a performance loss. An approach used by some schemes is to use error-correcting codes to decrease the failure probability. This leads to a reduction in the communication overhead, but makes the scheme more vulnurable to side-channel attacks.

As suggested by the wide range of failure probabilities in the NIST submissions, the implications of failures are still not well understood. In the absence of a clear evaluation technique for the impact of the failure rate, most NIST submissions have chosen a bound on the decryption failure probability around 2^{-128} based on educated guessing. As far as we know, only NIST-submission Kyber [40] provides an intuitive reasoning for its security against decryption failure attacks, but this approximation is not tight. They introduce a methodology that uses Grover's search algorithm to find ciphertexts that have a relatively high probability of triggering a decryption failure.

1.1 Related Works

The idea of exploiting decryption errors has been around for a long time and applies to all areas of cryptography [9]. For lattice-based encryption systems, the

Ajtai-Dwork scheme and NTRU have been a target for attacks using decryption failures. Hall, Goldberg, and Schneier [23] developed a reaction attack which recovers the Ajtai-Dwork private key by observing decryption failures. Hoffstein and Silverman [24] adapted the attack to NTRU and suggested modifying NTRU to use the Fujisaki-Okamoto transform [18] to protect against such attacks. Further work in this direction is given in [28], [26] and [19]. Fluhrer [17] described an attack against Ring-Learning with Errors (RLWE) schemes. In [15] his work was extended to more protocols and in [8] a chosen-ciphertext attack on the proposal HILA5 [37] was given, using decryption failures.

These attacks are chosen-ciphertext attacks on proposals with only IND-CPA-security and can be thwarted using an appropriate transformation to a chosen-ciphertext secure scheme, such as the Fujisaki-Okamoto transformation [18]. Hofheinz et al. [25] and later Jiang et al. [29] proved a bound on the impact of the failure rate on an IND-CCA secure KEM that is constructed using this transformation, but their bounds are squared in the failure probability in the quantum oracle setting, which seems a very conservative result. Guo, Johansson and Stankovski [22] proposed a key-recovery attack against the IND-CCA-secure version of QC-MDPC, which is a code-based scheme. It uses a distinguishing property that "colliding pairs" in the noise and the secret can change the decryption failure rate.

1.2 Contributions

In this paper we investigate the requirements for KEM's to resist decryption failure cryptanalysis. Having better security estimates can benefit the parameter selection process, resulting in improved security and efficiency. We focus on IND-CCA secure KEM's based on the (Ring/Module-)Learning with Errors and (Ring/Module-)Learning with Rounding paradigms. Nonetheless, the general method can also be applied to investigate the impact of failures on other schemes.

The exploitation of decryption failures of an IND-CCA secure cryptographic scheme proceeds in two main steps: obtaining ciphertexts that result in a decryption failure and estimating the secret based on these ciphertexts. In the first step, an adversary can use failure boosting to find 'weak' input vectors that artificially enlarge the failure rate of the scheme. In Sect. 3, we examine how an adversary can generate these 'weak' ciphertexts that increase the failure probability. We provide a theoretical framework and a Python implementation[1] to calculate an estimate of the minimum effort required for an adversary to obtain one failing ciphertext.

Once ciphertexts that trigger a decryption failure are collected, they can be used to estimate the secret. In Sect. 4, we study how much information is leaked by the collected failures. We develop a statistical model to estimate the secret from the failures and determine the residual entropy of the secret after a

[1] The software is available at https://github.com/danversjp/failureattack.

certain number of failures is collected. The estimate of the secret can be used to construct an easier problem that can be solved faster.

Section 5 combines failure boosting and the secret estimation technique from Sect. 4 to estimate the security of schemes based on (Ring/Module)-Learning with Errors and (Ring/Module)-Learning with Rounding under an attack exploiting decryption failures. We show that an attacker could significantly reduce the security of some schemes if he is able to perform sufficient decryption queries. However, for most NIST submissions, the number of decryption queries required is above practical limits.

In Sect. 6 we propose a new generic weak-key (multi-target) model exploiting the fact that a fraction of keys employed can have much higher error probability if the chosen weak ciphertexts satisfy certain key-related properties. The detailed attack procedure is similar to the attack discussed in the previous sections. It first consists of a precomputation phase where special messages and their corresponding error vectors are generated. Secondly, the messages are submitted for decryption and some decryption errors are observed. Finally, a phase with a statistical analysis of the messages/errors causing the decryption errors reveals the secret key.

In Sect. 7 we apply the model to ss-ntru-pke, a version of NTRUEncrypt targeting the security level of NIST-V. The proposed attack is an adaptive CCA attack with complexity below the claimed security level. We provide a Rust implementation[2] where parts of the attack are simulated.

2 Preliminaries

2.1 Notation

Let $\mathbb{Z}_q$ be the ring of integers modulo q represented in $(-q/2, q/2]$, let R_q denote the ring $\mathbb{Z}_q[X]/(X^n+1)$ and let $R_q^{k_1 \times k_2}$ denote the ring of $k_1 \times k_2$ matrices over R_q. Matrices will be represented with bold uppercase letters, while vectors are represented in bold lowercase. Let $\boldsymbol{A}_{ij}$ denote the element on the i^{th} row and j^{th} column of matrix $\boldsymbol{A}$, and let $\boldsymbol{A}_{ijk}$ denote the k^{th} coefficient of this element. Denote with $\boldsymbol{A}_{:j}$ the j^{th} column of $\boldsymbol{A}$.

The rounding operation $\lfloor x \rceil_{q \to p}$ is defined as $\lfloor p/q \cdot x \rceil \in \mathbb{Z}_p$ for an element $x \in \mathbb{Z}_q$, while $\texttt{abs}(\cdot)$ takes the absolute value of the input. These operations are extended coefficient-wise for elements of R_q and $R_q^{k_1 \times k_2}$. The two-norm of a polynomial $a \in R_q$ is written as $\|a\|_2$ and defined as $\sqrt{\sum_i a_i^2}$, which is extended to vectors as $\|\boldsymbol{a}\|_2 = \sqrt{\sum_i \|\boldsymbol{a}_i\|_2^2}$. The notation $a \leftarrow \chi(R_q)$ will be used to represent the sampling of $a \in R_q$ according to the distribution χ. This can be extended coefficient-wise for $\boldsymbol{A} \in R_q^{k_1 \times k_2}$ and is denoted as $\boldsymbol{A} \leftarrow \chi(R_q^{k_1 \times k_2})$. Let

[2] The software is available at https://github.com/atneit/ss-ntru-pke-attack-simulation.

$\mathcal{U}$ be the uniform distribution. Denote with $\chi_1 * \chi_2$ the convolution of the two distributions χ_1 and χ_2, and denote with $\chi^{*n} = \underbrace{\chi * \chi * \chi * \cdots * \chi * \chi}_{n}$ the n^{th} convolutional power of χ.

2.2 Cryptographic Definitions

A Public Key Encryption (PKE) is defined as a triple of functions PKE $=$ ($\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec}$), where the key generation $\texttt{KeyGen}$ returns a secret key sk and a public key pk, where the encryption $\texttt{Enc}$ produces a ciphertext c from the public key pk and the message $m \in \mathcal{M}$, and where the decryption $\texttt{Dec}$ returns the message m' given the secret key sk and the ciphertext c.

A Key Encapsulation Mechanism (KEM) consists of a triple of functions KEM $=$ ($\texttt{KeyGen}, \texttt{Encaps}, \texttt{Decaps}$), where $\texttt{KeyGen}$ generates the secret and public keys sk and pk respectively, where $\texttt{Encaps}$ generates a key $k \in \mathcal{K}$ and a ciphertext c from a public key pk, and where $\texttt{Decaps}$ requires the secret key sk, the public key pk and the ciphertext c to return a key k or the decryption failure symbol $\perp$. The security of a KEM can be defined under the notion of indistinguishability under chosen ciphertext attacks (IND-CCA),

$$\mathrm{Adv}_{\mathrm{KEM}}^{\text{ind-cca}}(A) = \left| P\left(b' = b : \begin{array}{c} (pk, sk) \leftarrow \texttt{KeyGen}(),\ b \leftarrow \mathcal{U}(\{0,1\}), \\ (c, d, k_0) \leftarrow \texttt{Encaps}(pk), \\ k_1 \leftarrow \mathcal{K},\ b' \leftarrow A^{\texttt{Decaps}}(pk, c, d, k_b), \end{array} \right) - \frac{1}{2} \right| .$$

2.3 LWE/LWR Problems

The decisional Learning with Errors problem (LWE) [36] consists of distinguishing a uniform sample $(\boldsymbol{A}, \boldsymbol{U}) \leftarrow \mathcal{U}(\mathbb{Z}_q^{k_1 \times k_2} \times \mathbb{Z}_q^{k_1 \times m})$ from an LWE-sample $(\boldsymbol{A}, \boldsymbol{B} = \boldsymbol{AS} + \boldsymbol{E})$, were $\boldsymbol{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{k_1 \times k_2})$ and where the secret vectors $\boldsymbol{S}$ and $\boldsymbol{E}$ are generated from the small distributions $\chi_s(\mathbb{Z}_q^{k_2 \times m})$ and $\chi_e(\mathbb{Z}_q^{k_1 \times m})$ respectively. The search LWE problem states that it is hard to recover the secret $\boldsymbol{S}$ from the LWE sample.

This definition can be extended to Ring- or Module-LWE [30,32] by using vectors of polynomials. In this case, the problem is to distinguish the uniform sample $(\boldsymbol{A}, \boldsymbol{U}) \leftarrow \mathcal{U}(R_q^{k_1 \times k_2} \times R_q^{k_1 \times m})$ from a generalized LWE sample $(\boldsymbol{A}, \boldsymbol{B} = \boldsymbol{AS} + \boldsymbol{E})$ in which $\boldsymbol{A} \leftarrow \mathcal{U}(R_q^{k_1 \times k_2})$ and where the secret vectors $\boldsymbol{S}$ and $\boldsymbol{E}$ are generated from the small distribution $\chi_s(R_q^{k_2 \times m})$ and $\chi_e(R_q^{k_1 \times m})$ respectively. Analogous to the LWE case, the search problem is to recover the secret $\boldsymbol{S}$ from a generalized LWE sample.

The decisional generalized Learning with Rounding (LWR) problem [7] is defined as distinguishing the uniform sample $(\boldsymbol{A}, \lfloor \boldsymbol{U} \rceil_{q \to p})$, with $\boldsymbol{A} \leftarrow \mathcal{U}(R_q^{k_1 \times k_2})$ and $\boldsymbol{U} \leftarrow \mathcal{U}(R_q^{k_1 \times m})$ from the generalized LWR sample $(\boldsymbol{A}, \boldsymbol{B} = \lfloor \boldsymbol{AS} \rceil_{q \to p})$ with $\boldsymbol{A} \leftarrow \mathcal{U}(R_q^{k_1 \times k_2})$ and $\boldsymbol{S} \leftarrow \chi_s(R_q^{k_2 \times m})$. In the analogous search problem, one has to find $\boldsymbol{S}$ from a generalized LWR sample.

2.4 (Ring/Module-)LWE Based Encryption

Let `gen` be a pseudorandom generator that expands seed$_{\boldsymbol{A}}$ into a uniformly random distributed matrix $\boldsymbol{A} \in R_q^{k \times k}$. Define `enc` as an encoding function that transforms a message m into a polynomial representation, and `dec` as the inverse decoding function. A general (Ring/Module-)LWE based PKE, consisting of a key generation, an encryption and a decryption phase, can then be constructed as described in Algorithms 1, 2 and 3 respectively. The randomness required for the generation of the secrets $\boldsymbol{S}'_B$, $\boldsymbol{E}'_B$ and $\boldsymbol{E}''_B$ during the encryption, is generated pseudorandomly from the uniformly distributed seed r that is given as an input.

Algorithm 1. PKE.KeyGen

```
Input:
Output: Public key $pk = (\boldsymbol{B}, seed_{\boldsymbol{A}})$, secret key $sk = \boldsymbol{S}_A)$.
1) $seed_{\boldsymbol{A}} \leftarrow \mathcal{U}(\{0,1\}^{256})$
2) $\boldsymbol{A} \leftarrow \texttt{gen}(\text{seed}_{\boldsymbol{A}}) \in R_q^{l \times l}$
4) $\boldsymbol{S}_A \leftarrow \chi_s(R_q^{l \times m}), \boldsymbol{E}_A \leftarrow \chi_e(R_q^{l \times m})$
5) $\boldsymbol{B} = \lfloor \boldsymbol{A}\boldsymbol{S}_A + \boldsymbol{E}_A \rceil_{q \to p}$
```

Algorithm 2. PKE.Enc

```
Input: Public key $pk = (\boldsymbol{B}, seed_{\boldsymbol{A}})$, message $m$, randomness $r$
Output: Ciphertext $c = (\boldsymbol{V}', \boldsymbol{B}')$
1) $\boldsymbol{A} \leftarrow \texttt{gen}(\text{seed}_{\boldsymbol{A}}) \in R_q^{l \times l}$
2) $\boldsymbol{S}'_B \leftarrow \chi_s(R_q^{l \times m}), \boldsymbol{E}'_B \leftarrow \chi_e(R_q^{l \times m})$
3) $\boldsymbol{E}''_B \leftarrow \chi_e(R_q^{m \times m})$
4) $\boldsymbol{B}_r = \lceil \boldsymbol{B} \rfloor_{p \to q}$
5) $\boldsymbol{B}' = \lfloor \boldsymbol{A}^T \boldsymbol{S}'_B + \boldsymbol{E}'_B \rceil_{q \to p}$
6) $\boldsymbol{V}' = \lfloor \boldsymbol{B}_r^T \boldsymbol{S}'_B + \boldsymbol{E}''_B + \texttt{enc}(m) \rceil_{q \to t}$
```

Algorithm 3. PKE.Dec

```
Input: Secret key $sk = \boldsymbol{S}_A$, ciphertext $c = (\boldsymbol{V}', \boldsymbol{B}')$
Output: Message $m'$
1) $\boldsymbol{B}'_r = \lfloor \boldsymbol{B}' \rceil_{p \to q}$
2) $\boldsymbol{V}'_r = \lfloor \boldsymbol{V}' \rceil_{t \to q}$
3) $\boldsymbol{V} = \boldsymbol{B}_r'^T \boldsymbol{S}_A$
4) $m' = \texttt{dec}(\boldsymbol{V}'_r - \boldsymbol{V})$
```

Using this general framework, specific schemes can be described with appropriate parameter choices. When the ring R_q is chosen as $\mathbb{Z}_q$, the encryption is LWE-based as can be seen in FrodoKEM [33] and Emblem [42]. A value of $l = 1$ indicates a Ring-LWE based scheme including New Hope [5], LAC [31], LIMA [43] or R.Emblem [42]. If $l \neq 1$ and $R_q \neq \mathbb{Z}_q$, the scheme is based on the Module-LWE hard problem such as Kyber [10]. When referring to Kyber throughout this paper, we will consider the original version that includes rounding. The special case that $\chi_e = 0$ corresponds to (Module/Ring)-LWR-based schemes such as Round2 [6] and Saber [13]. In Lizard [12], a combination of an LWE and LWR problem is proposed. In most (Ring/Module-)LWE based schemes, $q = p$ and no rounding is performed in the calculation of $\boldsymbol{B}$ and $\boldsymbol{B}'$, while t is in most schemes much smaller than q leading to a drastic rounding of $\boldsymbol{V}'$.

We define $\boldsymbol{U}_A$, $\boldsymbol{U}'_B$ en $\boldsymbol{U}''_B$ as the errors introduced by the rounding operations, which is formalized as follows:

$$\boldsymbol{U}_A = \boldsymbol{A}\boldsymbol{S}_A + \boldsymbol{E}_A - \boldsymbol{B}_r\,, \tag{1}$$

$$\boldsymbol{U}'_B = \boldsymbol{A}^T\boldsymbol{S}'_B + \boldsymbol{E}'_B - \boldsymbol{B}'_r\,, \tag{2}$$

$$\boldsymbol{U}''_B = \boldsymbol{B}_r^T\boldsymbol{S}'_B + \boldsymbol{E}''_B + \texttt{enc}(m) - \boldsymbol{V}'_r\,. \tag{3}$$

Let $\boldsymbol{S}$ be the vector constructed as the concatenation of the vectors $-\boldsymbol{S}_A$ and $\boldsymbol{E}_A + \boldsymbol{U}_A$, let $\boldsymbol{C}$ be the concatenation of $\boldsymbol{E}'_B + \boldsymbol{U}'_B$ and $\boldsymbol{S}'_B$, and let $\boldsymbol{G} = \boldsymbol{E}''_B + \boldsymbol{U}''_B$. An attacker that generates ciphertexts can compute $\boldsymbol{C}$ and $\boldsymbol{G}$ and tries to obtain information about $\boldsymbol{S}$. These variables are summarized below:

$$\boldsymbol{S} = \begin{pmatrix} -\boldsymbol{S}_A \\ \boldsymbol{E}_A + \boldsymbol{U}_A \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} \boldsymbol{E}'_B + \boldsymbol{U}'_B \\ \boldsymbol{S}'_B \end{pmatrix}, \quad \boldsymbol{G} = \boldsymbol{E}''_B + \boldsymbol{U}''_B\,. \tag{4}$$

After the execution of this protocol, the two parties will arrive at the same key if the decoding $\texttt{dec}(\boldsymbol{V}'_r - \boldsymbol{V})$ equals m. The term $\boldsymbol{V}'_r - \boldsymbol{V}$ can be rewritten as $(\boldsymbol{E}_A + \boldsymbol{U}_A)^T\boldsymbol{S}'_B - \boldsymbol{S}_A^T(\boldsymbol{E}'_B + \boldsymbol{U}'_B) + (\boldsymbol{E}'' + \boldsymbol{U}''_B) + \texttt{enc}(m) = \boldsymbol{S}^T\boldsymbol{C} + \boldsymbol{G} + \texttt{enc}(m)$. The message can be recovered if and only if $\texttt{abs}(\boldsymbol{S}^T\boldsymbol{C} + \boldsymbol{G}) < q_t$ for a certain threshold q_t that is scheme dependent.

We will say that a (decryption) failure occurred if the parties do not arrive at a common key due to a coefficient of $\texttt{abs}(\boldsymbol{S}^T\boldsymbol{C} + \boldsymbol{G})$ that is larger than q_t, and will define $F(\boldsymbol{C}, \boldsymbol{G})$ as the probability of a decryption failure given $\boldsymbol{C}$ and $\boldsymbol{G}$ averaged over all $\boldsymbol{S}$, which can be expressed as $\sum_{\boldsymbol{S}} P(\texttt{abs}(\boldsymbol{S}^T\boldsymbol{C} + \boldsymbol{G}) > q_t \mid \boldsymbol{S})P(\boldsymbol{S})$.

2.5 Fujisaki-Okamoto Transformation

Using the Fujisaki-Okamoto transform [18,25], one can transform a chosen plaintext secure PKE to an IND-CCA secure KEM. On top of the encryption from the PKE, the KEM defines an encapsulation and decapsulation function as described in Algorithms 4 and 5, using hash functions $\mathcal{H}$ and $\mathcal{G}$.

Algorithm 4. KEM.ENCAPS

```
Input: Public key $pk$
Output: Ciphertext $c$, key $K$
1) $m \leftarrow \mathcal{U}(\{0,1\}^{256})$
2) $r = \mathcal{G}(m)$
3) $c = \texttt{PKE.Enc}(pk, m, r)$
4) $K = \mathcal{H}(r)$
```

Algorithm 5. KEM.DECAPS

```
Input: Public key $pk$, secret key $sk$, ciphertext $c$
Output: Key $K$ or $\perp$
1) $m' = \texttt{PKE.Dec}(sk, c)$
2) $r' = \mathcal{G}(m')$
3) $c' = \texttt{PKE.Enc}(pk, m', r')$
4) If $c = c'$:
5)     $K = \mathcal{H}(r)$
6) Else:
7)     $K = \perp$
```

3 Weak-Ciphertext Failure Boosting

In this section, we will develop a method to estimate the minimum amount of work to obtain one ciphertext that triggers a decryption failure. In contrast to an honest party that generates ciphertexts randomly, an attacker can search for ciphertexts that have a higher failure probability than average, which will be called 'weak'. As $\boldsymbol{C}$ and $\boldsymbol{G}$ are the only terms with which an attacker can influence decryption failures, the search for weak ciphertexts boils down to the search for weak $(\boldsymbol{C}, \boldsymbol{G})$. However, the pair $(\boldsymbol{C}, \boldsymbol{G})$ is generated through a hash $H()$ with random seed r, and during decryption it is checked whether the generator of the ciphertext knew the preimage r of $(\boldsymbol{C}, \boldsymbol{G})$. Therefore an attacker is forced to resort to a brute force search, which can be sped up at most quadratically using Grover's algorithm [20].

To find a criterion for our search, we sort all possible $(\boldsymbol{C}, \boldsymbol{G})$ according to an increasing failure probability $F(\boldsymbol{C}, \boldsymbol{G})$. This list can then be divided into two sets using a threshold failure probability f_t: weak vectors with a failure probability higher or equal than f_t, and strong vectors with lower failure probability. Let $f()$ be the deterministic function that generates $\boldsymbol{C}$ and $\mathbf{G}$ from the random seed r. For a certain f_t, we can calculate the probability of generating a weak pair: $\alpha = P(F(\boldsymbol{C}, \boldsymbol{G}) > f_t \,|\, r \leftarrow \mathcal{U}, (\boldsymbol{C}, \boldsymbol{G}) = f(H(r)))$, and the probability of a decryption failure when a weak pair is used: $\beta = P(\texttt{abs}(\boldsymbol{S}^T \boldsymbol{C} + \boldsymbol{G}) > q_t \,|\, r \leftarrow \mathcal{U}, (\boldsymbol{C}, \boldsymbol{G}) = f(H(r)), F(\boldsymbol{C}, \boldsymbol{G}) > f_t)$.

The amount of work for an adversary to find a weak pair $(\boldsymbol{C}, \boldsymbol{G})$ is proportional to α^{-1}, but can be sped up quadratically using Grover's algorithm on a quantum computer, resulting in an expected workload of $\sqrt{\alpha^{-1}}$. On the other hand, the probability of a decryption failure given a weak pair cannot be improved using quantum computation assuming that the adversary has no quantum access to the decryption oracle. This assumption is in agreement with the premise in the NIST Post-Quantum Standardization Call for Proposals [2]. The expected work required to find a decryption failure given f_t is therefore the expected number of queries using weak ciphertexts times the expected amount of work to find a weak ciphertext, or $(\alpha \cdot \beta)^{-1}$ with a classical computer and $(\sqrt{\alpha} \cdot \beta)^{-1}$ with a quantum computer. An optimization over f_t gives the minimal effort to find one decryption failure.

3.1 Practical Calculation

For most schemes, the full sorted list $(\boldsymbol{C}, \boldsymbol{G})$ is not practically computable, but using some observations and assumptions, an estimate can be found. The next three steps aim at calculating the distribution of the failure probability $F(\boldsymbol{C}, \boldsymbol{G})$, i.e. what is the probability of finding a $(\boldsymbol{C}, \boldsymbol{G})$ pair with a certain failure probability f. This distribution gives enough information to calculate α and β for a certain f_t.

First, we can remove the hash $H(.)$ in the probability expression by assuming the output of $f(H(.))$ given random input r to behave as the probability distributions (χ_C, χ_G), resulting in: $\alpha = P(F(\boldsymbol{C}, \boldsymbol{G}) > f_t \mid (\boldsymbol{C}, \boldsymbol{G}) \leftarrow (\chi_C, \chi_G))$ and $\beta = P(\texttt{abs}(\boldsymbol{S}^T\boldsymbol{C} + \boldsymbol{G}) > q_t \mid (\boldsymbol{C}, \boldsymbol{G}) \leftarrow (\chi_C, \chi_G), F(\boldsymbol{C}, \boldsymbol{G}) > f_t)$.

Secondly, we assume that the coefficients of $\boldsymbol{S}^T\boldsymbol{C}$ are normally distributed, which is reasonable as the coefficients are a sum of $2(l \cdot n)$ distributions that are somewhat close to a Gaussian. The coefficients of the polynomial $(\boldsymbol{S}^T\boldsymbol{C})_{ij}$ will be distributed with mean $\mu = 0$ because of symmetry around 0, while the variance can be calculated as follows, after defining χ_{e+u} as the distribution of the coefficients of $\boldsymbol{E}_A + \boldsymbol{U}_A$:

$$\texttt{var}((\boldsymbol{S}^T\boldsymbol{C})_{ijk}) = \texttt{var}(\sum_{i=0}^{l-1}\sum_{k=0}^{n-1} \boldsymbol{C}_{ijk} s_{ijk} + \sum_{i=l}^{2l-1}\sum_{k=0}^{n-1} \boldsymbol{C}_{ijk} e_{ijk}) \tag{5}$$

$$\text{where: } s_{ijk} \leftarrow \chi_s \text{ and } e_{ijk} \leftarrow \chi_{e+u} \tag{6}$$

$$= \sum_{i=0}^{l-1}\sum_{k=0}^{n-1} \boldsymbol{C}_{ijk}^2 \texttt{var}(\chi_s) + \sum_{i=l}^{2l-1}\sum_{k=0}^{n-1} \boldsymbol{C}_{ijk}^2 \texttt{var}(\chi_{e+u}) \tag{7}$$

$$= \|(\boldsymbol{E}'_B + \boldsymbol{U}'_B)_{:j}\|_2^2 \texttt{var}(\chi_s) + \|(\boldsymbol{S}'_B)_{:j}\|_2^2 \texttt{var}(\chi_{e+u})\,. \tag{8}$$

Therefore, the variance of the coefficients of $\boldsymbol{S}^T\boldsymbol{C}$ for a given $\boldsymbol{C}$ is the same for all coefficients in the same column. This variance will be denoted as σ_j^2 for coefficients in the j^{th} column of $\boldsymbol{S}^T\boldsymbol{C}$. Furthermore, following the Gaussian assumption, the failure probability given σ_j^2 is the same as the failure probability given the j^{th} column of $\boldsymbol{C}$.

In the third step we gradually calculate the distribution of the failure probability. We start from the distribution of the failure probability of the coefficient at the ijk^{th} position given σ_j, denoted with $\chi_{coef\,|\,\sigma}$. This distribution expresses the probability of finding a $\boldsymbol{G}$ so that the failure probability is equal to f_{ijk} given a certain value of $\boldsymbol{C}$ (or equivalently σ_j^2) and can be expressed as follows:

$$P(f_{ijk}\,|\,\boldsymbol{G} \leftarrow \chi_G, \boldsymbol{C})\,, \tag{9}$$

where:

$$\begin{aligned} f_{ijk} &= P(\texttt{abs}(\boldsymbol{S}^T\boldsymbol{C} + \boldsymbol{G})_{ijk} > q_t\,|\,\boldsymbol{G}, \boldsymbol{C}) && (10)\\ &\approx P(\texttt{abs}(x + \boldsymbol{G}_{ijk}) > q_t\,|\,\boldsymbol{G}, x \leftarrow \mathcal{N}(0, \sigma_j^2), \sigma_j^2)\,. && (11) \end{aligned}$$

The distribution $\chi_{col\,|\,\sigma}$, which models the probability of a failure in the j^{th} column of $\texttt{abs}(\boldsymbol{S}^T\boldsymbol{C} + \boldsymbol{G})$ given σ_j^2, can be calculated using the convolution of the distributions of the mn individual coefficient failures $\chi_{coef\,|\,\sigma}$ as follows:

$$\chi_{col\,|\,\sigma} = \chi_{coef\,|\,\sigma}^{*nm}\,. \tag{12}$$

The conditioning on σ_j^2 is necessary to counter the dependency between the coefficients of the columns of $\texttt{abs}(\boldsymbol{S}^T\boldsymbol{C} + \boldsymbol{G})$, which are dependent as a result of sharing the same variance σ_j^2.

The distribution of failure probabilities in the j^{th} column of $\boldsymbol{S}^T\boldsymbol{C}$, denoted as χ_{col}, can then be calculated using a weighted average over the possible values of σ_j^2 as follows:

$$\chi_{col} = \sum_{l_c} P(f\,|\,f \leftarrow \chi_{col,\sigma}^{*nm}) P(\sigma_j^2 = l_c)\,. \tag{13}$$

Finally we can calculate the full failure distribution χ_{FAIL} as the convolution of the m probability distributions corresponding to the failure distributions of the different columns. This convolution does not have the dependency on σ_j^2 as failures of different columns are independent conditioned on $\boldsymbol{C}$ and $\boldsymbol{G}$, therefore:

$$\chi_{\text{FAIL}} = \chi_{col}^{*m}\,. \tag{14}$$

From this failure distribution, we can calculate α and β for an arbitrary value of f_t:

$$\begin{aligned} \alpha &= P(f > f_t\,|\,f \leftarrow \chi_{\text{FAIL}})\,, && (15)\\ \beta &= \frac{\sum_{f > f_t} f \cdot P(f\,|\,f \leftarrow \chi_{\text{FAIL}})}{\alpha}\,. && (16) \end{aligned}$$

We want to stress that this calculation is not exact, mainly due to the Gaussian assumption in the second step. More accurate estimates could be obtained with a more accurate approximation in step 2, tailored for a specific scheme. In this case, the assumptions and calculations of step 1 and step 3 remain valid. For the estimations of LAC [31] in subsequent paragraphs, we followed their approach for the calculation of the effect of the error correcting code. Note that this is not an exact formula as the inputs of the error correcting code are correlated through their polynomial structure.

In Fig. 1 we compare the values of α and β calculated using the technique described above, with exhaustively tested values on a variant of LAC128 without error correction. For step 2 of the practical calculation, we use both a Gaussian approximation as well as a binomial approximation that is more tailored for LAC. We can observe that our estimation of the effect of failure boosting is relatively close to reality.

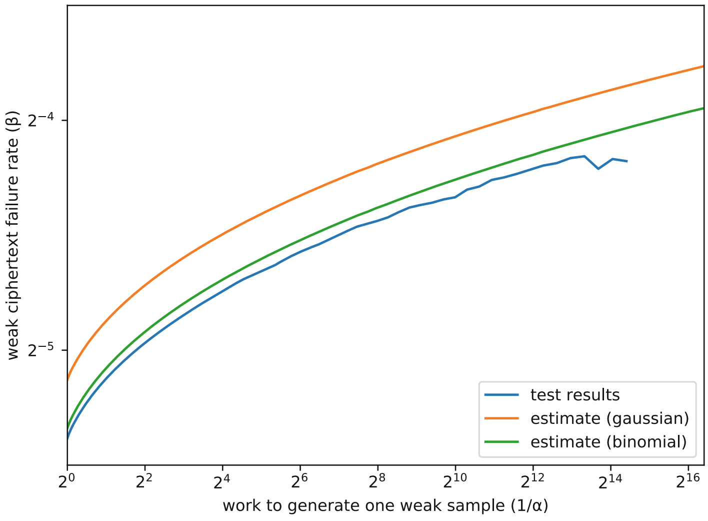

Fig. 1. The failure rate of one weak ciphertext (β) as a function of the work required to generate one weak ciphertext (α) on a classical computer for LAC128 without error correction.

3.2 Applications of Failure Boosting

Failure boosting is a useful technique in at least three scenarios: first, if there is no multi-target protection, second, if the adversary can only perform a limited number of queries to the decryption oracle and third, if the adversary has access to a quantum computer.

In some (Ring/Module-)LWE/LWR schemes, the seed r is the only input to the pseudorandom generator that generates $\boldsymbol{C}$ and $\boldsymbol{G}$. This paves the way to a multi-target attack where precomputed weak values of r can be used against multiple targets: after choosing the parameter f_t, the adversary can generate weak ciphertexts in approximately α^{-1} time ($\sqrt{\alpha^{-1}}$ if he has access to a quantum computer). Each precomputed sample has then a failure probability of β against every target. Figure 2 shows the failure probability of one weak ciphertext versus the amount of work to generate that ciphertext on a classical computer. Multi-target protection, for example by including the public key into the generation of $\boldsymbol{C}$ en $\boldsymbol{G}$ as proposed in Kyber [10] and Saber [13] is a relatively cheap option to resolve this issue.

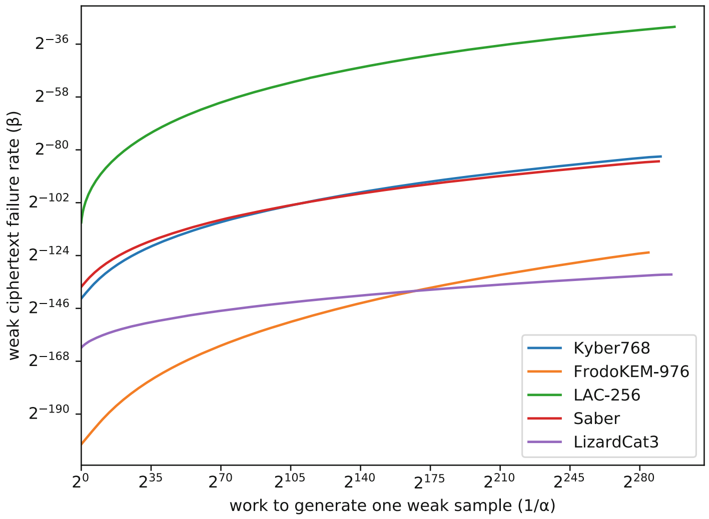

Fig. 2. The failure rate of one weak ciphertext (β) as a function of the work required to generate one weak ciphertext (α) on a classical computer.

If the adversary can only perform a limited number of decryption queries, for example 2^{64} in the NIST Post-Quantum Standardization Call for Proposals [2], the adversary can use failure boosting to reduce the number of required decryption queries. To this end, he chooses the parameter f_t so that the inverse of the failure probability β^{-1} equals the decryption query limit n_d, which results in a probability of finding a decryption failure of approximately $(1 - e^{-1}) \approx 0.63$. To find i failures with similar probability, the failure probability should be brought up so that $\beta^{-1} = n_d/i$. Since the amount of work to generate one input of the decryption query is approximately α^{-1} ($\sqrt{\alpha^{-1}}$ quantumly), the total amount of work expected is $\alpha^{-1}\beta^{-1}$, ($\sqrt{\alpha^{-1}}\beta^{-1}$ quantumly). Figure 3 shows the expected total amount of work to find one decryption failure with a classical computer, versus the failure rate of one weak ciphertext.

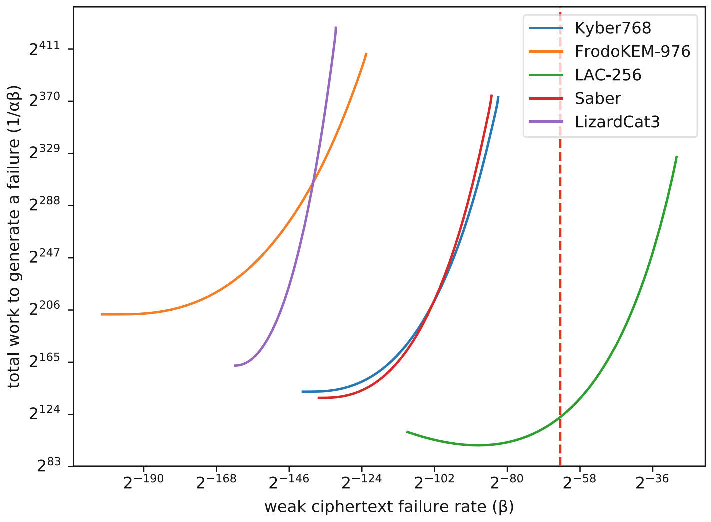

Fig. 3. The expected amount of work ($\alpha^{-1}\beta^{-1}$) on a classical computer, as a function of the failure rate of one weak ciphertext (β). The red dotted line indicates a failure rate of 2^{-64}. (Color figure online)

An adversary with a quantum computer always benefits from failure boosting, as the search for weak ciphertexts can be sped up using Grover's algorithm. However, this speedup is not quadratic if the adversary has no quantum access to the decryption oracle. Figure 4 shows the total amount of expected work to find one decryption failure, versus the amount of work to find one weak ciphertext on a quantum computer $\sqrt{\alpha^{-1}}$.

4 Estimation of the Secret

Finding a decryption failure does not immediately break the security of the KEM, but it does provide extra information to an adversary. In this section we will investigate how much this information leaks about the secret. An adversary that has obtained ciphertexts that produce decryption failures can use them to make an estimation of the secret $\boldsymbol{S}$.

When a failure occurs, we know that at least one coefficient of $\texttt{abs}(\boldsymbol{S}^T\boldsymbol{C}+\boldsymbol{G})$ is larger than the threshold q_t. This leads to a classification of the coefficients in the set of fail coefficients v_f and no-fail coefficients v_s. To each coefficient at position (i, j, k), a vector of integers $\boldsymbol{s}$ can be associated by taking the coefficients of $\boldsymbol{S}_{:i}$. Similarly, the coefficient can be linked to a vector of integers $\boldsymbol{c}$ calculated as a function of $\boldsymbol{C}_{:j}$ and k, so that the multiplication $\boldsymbol{s}\boldsymbol{c}$ equals that coefficient.

No-fail vectors will contain negligible information about the secret $\boldsymbol{s}$, but failure vectors do carry clues, as the threshold exceeding value of the coefficients of $\boldsymbol{S}^T\boldsymbol{C}+\boldsymbol{G}$ implies a correlation between the corresponding $\boldsymbol{c}$ and $\boldsymbol{s}$. This correlation can be positive, in case of a large positive value of the coefficient, or

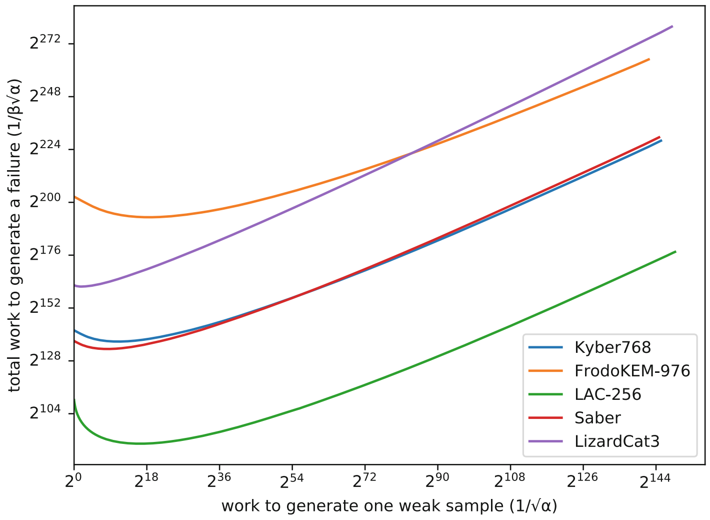

Fig. 4. The expected amount of work ($\sqrt{\alpha^{-1}}\beta^{-1}$) as a function of the work required to generate one weak ciphertext ($\sqrt{\alpha^{-1}}$) on a classical computer.

negative, in case of a large negative value of the coefficient. Consequently, the fail coefficients can be further divided into the sets of positive v_{fp} and negative v_{fn} fail coefficients respectively. Moreover, negative fail vectors can be transformed into positive fail vectors by multiplication with -1. Note that failure coefficients at position (i, j, k) will only contain information about the j^{th} column of $\boldsymbol{S}$, which is why the estimation of the columns of $\boldsymbol{S}$ can be performed independently.

4.1 One Positive Failure Vector

We will first examine the case where we know one positive fail vector $\boldsymbol{c}$ and associated coefficient $\boldsymbol{G}_{i,j,k}$, which we will denote with g. This corresponds to the case where one failing ciphertext and the location and sign of the error is known. The question is how much the knowledge about $\boldsymbol{c}$ and g can improve our estimate of the associated secret $\boldsymbol{s}$. Applying Bayes' theorem and assuming independence between the coefficients of $\boldsymbol{c}$ and $\boldsymbol{s}$ that are on different positions, we can write:

$$\begin{aligned} P(\boldsymbol{s}_i \mid \boldsymbol{c}, g, \boldsymbol{s}\boldsymbol{c} > q_t - g) &\approx P(\boldsymbol{s}_i \mid \boldsymbol{c}_i, g, \boldsymbol{s}\boldsymbol{c} > q_t - g) && (17)\\ &= \frac{P(\boldsymbol{s}\boldsymbol{c} > q_t - g \mid \boldsymbol{s}_i, \boldsymbol{c}_i, g) P(\boldsymbol{s}_i \mid \boldsymbol{c}_i, g)}{P(\boldsymbol{s}\boldsymbol{c} > q_t - g \mid \boldsymbol{c}_i, g)} && (18)\\ &= \frac{P(\sum_j^{j\neq i} \boldsymbol{s}_j \boldsymbol{c}_j > q_t - g - \boldsymbol{s}_i\boldsymbol{c}_i \mid \boldsymbol{s}_i, \boldsymbol{c}_i, g) P(\boldsymbol{s}_i)}{P(\boldsymbol{s}\boldsymbol{c} > q_t - g \mid \boldsymbol{c}_i, g)}. && (19) \end{aligned}$$

The improved estimates for the coefficients of $\boldsymbol{s}$ can in turn be used to get an estimate $\boldsymbol{s}_{est}$ that minimizes its variance $E[(\boldsymbol{s}_{est} - \boldsymbol{s})^2]$ as follows:

$$0 = \frac{d}{d\boldsymbol{s}_{est,i}} E((\boldsymbol{s}_{est,i} - \boldsymbol{s}_i)^2) \tag{20}$$

$$= 2 \sum_{\boldsymbol{s}_i} (\boldsymbol{s}_{est,i} - \boldsymbol{s}_i) P(\boldsymbol{s}_i)\,, \tag{21}$$

$$\text{or:} \quad \boldsymbol{s}_{est,i} = \sum_{\boldsymbol{s}_i} \boldsymbol{s}_i \cdot P(\boldsymbol{s}_i)\,. \tag{22}$$

The estimate of $\boldsymbol{s}$ gives the estimate of the j^{th} column of $\boldsymbol{S}$, which can be divided trivially in an approximation $\boldsymbol{S}_{A,est}$ of $(\boldsymbol{S}_A)_{:j}$ and $\boldsymbol{E}_{A,est}$ of $(\boldsymbol{E}_A + \boldsymbol{U}_A)_{:j}$. These vectors can be used to transform the original (Ring/Module-)LWE/LWR sample $(\boldsymbol{A}, \boldsymbol{A}(\boldsymbol{S}_A)_{:j} + (\boldsymbol{E}_A + \boldsymbol{U}_A)_{:j})$ into a (Ring/Module-)LWE alike problem with a smaller secret variance by subtracting $\boldsymbol{A}\boldsymbol{S}_{A,est} + \boldsymbol{E}_{A,est}$. This results in the sample $(\boldsymbol{A}, \boldsymbol{A}((\boldsymbol{S}_A)_{:j} - \boldsymbol{S}_{A,est}) + (\boldsymbol{E}_A + \boldsymbol{U}_A)_{:j} - \boldsymbol{E}_{A,est})$, which is a problem with smaller secret $(\boldsymbol{S}_A)_{:j} - \boldsymbol{S}_{A,est}$ and noise $(\boldsymbol{E}_A + \boldsymbol{U}_A)_{:j} - \boldsymbol{E}_{A,est}$. We will call this new problem the simplified problem.

4.2 Multiple Fail Vectors

Having access to m positive fail vectors $\boldsymbol{c}^{(1)} \dots \boldsymbol{c}^{(m)}$ from the same column, with corresponding values of G as $g^{(1)} \dots g^{(m)}$, an adversary can improve his estimate of $P(\boldsymbol{s})$ and therefore obtain a better estimate $\boldsymbol{s}_{est}$, assuming that the failure vectors $\boldsymbol{c}_i$ are independent conditioned on $\boldsymbol{s}$. This corresponds to knowing m failing ciphertexts and the location and sign of their errors.

$$P(\boldsymbol{s}_i \,|\, \boldsymbol{c}^{(1)} \dots \boldsymbol{c}^{(m)}, g^{(1)} \dots g^{(m)}) \approx P(\boldsymbol{s}_i \,|\, \boldsymbol{c}_i^{(1)} \dots \boldsymbol{c}_i^{(m)}, g^{(1)} \dots g^{(m)}) \tag{23}$$

$$= \frac{P(\boldsymbol{c}_i^{(1)} \dots \boldsymbol{c}_i^{(m)} \,|\, \boldsymbol{s}_i, g^{(1)} \dots g^{(m)}) P(\boldsymbol{s}_i \,|\, g^{(1)} \dots g^{(m)})}{P(\boldsymbol{c}_i^{(1)} \dots \boldsymbol{c}_i^{(m)} \,|\, g^{(1)} \dots g^{(m)})} \tag{24}$$

$$= \frac{P(\boldsymbol{s}_i) \prod_{k=1}^{m} P(\boldsymbol{c}_i^{(k)} \,|\, \boldsymbol{s}_i, g^{(k)})}{\prod_{k=1}^{m} P(\boldsymbol{c}_i^{(k)} \,|\, g^{(k)})}\,. \tag{25}$$

Similar to Eq. 19, $P(\boldsymbol{c}_i \,|\, \boldsymbol{s}_i, g^{(k)})$ can be calculated as:

$$P(\boldsymbol{c}_i \,|\, \boldsymbol{s}_i, g, \boldsymbol{s}\boldsymbol{c} > q_t - g) = \frac{P(\boldsymbol{s}\boldsymbol{c} > q_t - g \,|\, \boldsymbol{s}_i, \boldsymbol{c}_i, g) P(\boldsymbol{c}_i \,|\, \boldsymbol{s}_i, g)}{P(\boldsymbol{s}\boldsymbol{c} > q_t - g \,|\, \boldsymbol{s}_i, g)} \tag{26}$$

$$= \frac{P(\sum_j^{j \neq i} \boldsymbol{s}_j \boldsymbol{c}_j > q_t - g - \boldsymbol{s}_i \boldsymbol{c}_i \,|\, \boldsymbol{s}_i, \boldsymbol{c}_i, g) P(\boldsymbol{c}_i)}{P(\boldsymbol{s}\boldsymbol{c} > q_t - g \,|\, \boldsymbol{s}_i, g)}\,. \tag{27}$$

In subsequent calculations, each value of the coefficient of g is taken as the maximum possible value.

4.3 Classification of Vectors

The above approach assumes a prior knowledge of the exact position and sign of the errors. This information is needed to link coefficients of $\boldsymbol{C}$ with their corresponding coefficient of $\boldsymbol{S}$. However, this is not always a trivial problem. For most schemes there are three sources of extra information that will allow to perform this classification with a high probability using only a few decryption failures.

Firstly, a large coefficient of $\boldsymbol{G}$ would induce a higher failure probability for the corresponding coefficient of the error term $\boldsymbol{S}^T\boldsymbol{C}+\boldsymbol{G}$. Thus, failures are more likely to happen at positions linked to that coefficient of $\boldsymbol{G}$. Moreover, a positive value of the coefficient suggests a positive error so that $\boldsymbol{c} \in v_{fp}$, while a negative value hints at a negative error, or $\boldsymbol{c} \in v_{fn}$.

Secondly, as vectors $\boldsymbol{c} \in v_f$ are correlated with the secret $\boldsymbol{s}$, they are also correlated with each other. Therefore, vectors $\boldsymbol{c} \in v_f$ are more correlated between each other than a vector $\boldsymbol{c} \in v_f$ with a vector $\boldsymbol{c} \in v_s$. Moreover, a high positive correlation suggests that the vectors share the same class v_{fp} or v_{fn}, while a high negative correlation indicates that the vectors have a different classes. This allows for a clustering of the fail vectors using the higher than average correlation, under the condition that the correlation difference is high enough. This correlation difference is related to the failure rate: a low failure rate implies a higher correlation because only ciphertexts that are highly correlated with the secret lead to a failure rate in this case. For example, Fig. 5 shows an estimate of the correlations between vectors of the classes v_{fp} (pos), v_{fn} (neg) and v_s (nofail) in Kyber768. This approach does not work for schemes with strong error correcting codes (ECC) such as LAC, as the bit error rate before correction is relatively high for these types of algorithms, leading to a relatively low correlation between failure vectors.

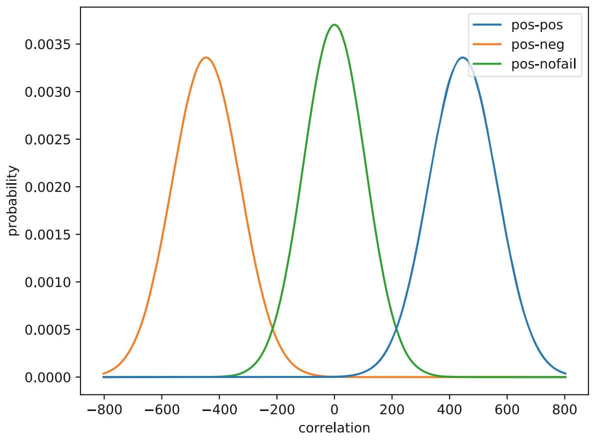

Fig. 5. The probability of a certain value of the correlation between different classes of vectors in Kyber768.

In case of a ring/module structure of the coefficients of $\boldsymbol{S}$, an additional structure arises leading to an artifact in which some pairs of no-fail coefficients within the same polynomial also have high correlation of their corresponding vectors. Imagine a pair of failure coefficients at positions (i, j, k_1) and (i, j, k_2) from different decryption failures a, b, with corresponding matrices $\boldsymbol{C}^{(a)}$ and $\boldsymbol{C}^{(b)}$. The correlation of the vectors $\boldsymbol{c}^{(a)}$ and $\boldsymbol{c}^{(b)}$ can be written as $X^{k_1} \boldsymbol{C}^{(a)T}_{:,j} X^{k_2} \boldsymbol{C}^{(b)}_{:,j} = X^{k_1+k_2} \boldsymbol{C}^{(a)T}_{:,j} \boldsymbol{C}^{(b)}_{:,j}$, from which is clear that the vectors from $\boldsymbol{C}^{(a)}$ and $\boldsymbol{C}^{(b)}$, with respective positions $(i, j, k_1 - t)$ and $(i, j, k_2 + t)$ have the same correlation. The clustering will thus result in n classes, with one class containing the failure vectors. Combining this information with the information of the first method gives an adversary the failure vectors with high probability. Otherwise, an adversary can estimate the secret n times and check the validity of the result using the (Ring/Module-)LWE/LWR problem.

Finally, for schemes that use error correcting codes to reduce their failure probability, side channel leakage during the error correction might reveal information on the presence or position of failure coefficients. Note that if this is the case, it might not even be necessary to obtain a decryption failure since failing coefficients could also be collected on successful decryptions where there is at least one failing coefficient.

4.4 Implications

Figure 6 depicts the relative variance reduction of the secret as a function of the number of positive failure vectors for various schemes. For schemes that have a very low failure probability for individual coefficients of $\boldsymbol{S}^T \boldsymbol{C} + \boldsymbol{G}$, such as Kyber, Saber and FrodoKEM, the variance of the secret drastically reduces upon knowing only a few failing ciphertexts. Assuming that the simplified problem, that takes into account the estimate of the secret, has the same complexity as a regular (Ring/Module-)LWE problem with similar secret variance, we can calculate the remaining hardness of the simplified problem as a function of the number of positive failure vectors as shown in Fig. 7 using the toolbox provided by Albrecht et al. [4] using the Q-core sieve estimate.

The effectiveness of the attack declines as the failure probability of the individual coefficients increases, since the correlation between the secret and the failing ciphertext is lower in this case. This can be seen in the case of LAC, where the individual coefficients have relatively high failure rates due to a strong error correcting code. On the other hand, a failing ciphertext will contain multiple errors, making it easier to collect multiple failure vectors.

Note that once one or more failures are found, they can be used to estimate the secret. This estimate in turn can be used to improve the search for weak ciphertexts by considering $F(\boldsymbol{C}, \boldsymbol{G})$ as $\sum_{\boldsymbol{S}} P(\text{FAIL}(\boldsymbol{C}, \boldsymbol{G}), \boldsymbol{S})$, where $\boldsymbol{S}$ is not sampled from $\chi_{\boldsymbol{S}}$, but from the new probability distribution $\chi_{\boldsymbol{S}_{est}}$. Therefore, the search for weak keys could become easier the more failures have been found. However, we do not take this effect into account in this paper.

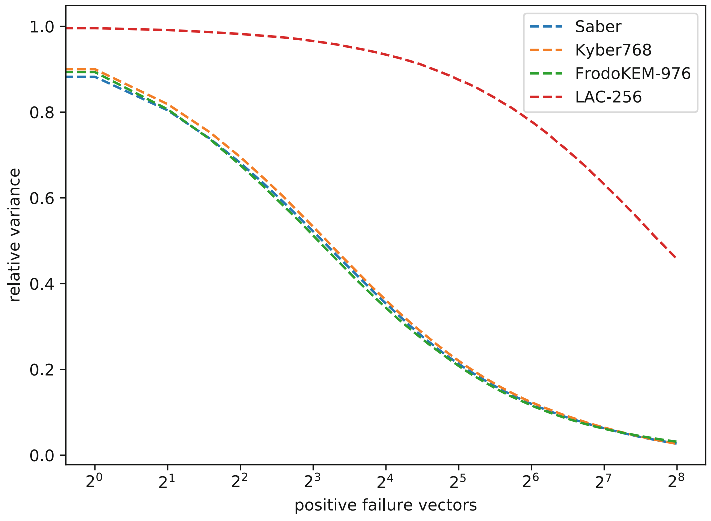

Fig. 6. The relative reduction in entropy as a function of the number of positive failure vectors

5 Weak-Ciphertext Attack

Using the failure boosting technique from Sect. 3 and the secret estimation method from Sect. 4, we can lower the security of a (Ring/Module-)LWE/LWR scheme on the condition that its failure rate is high enough. To this end, we first

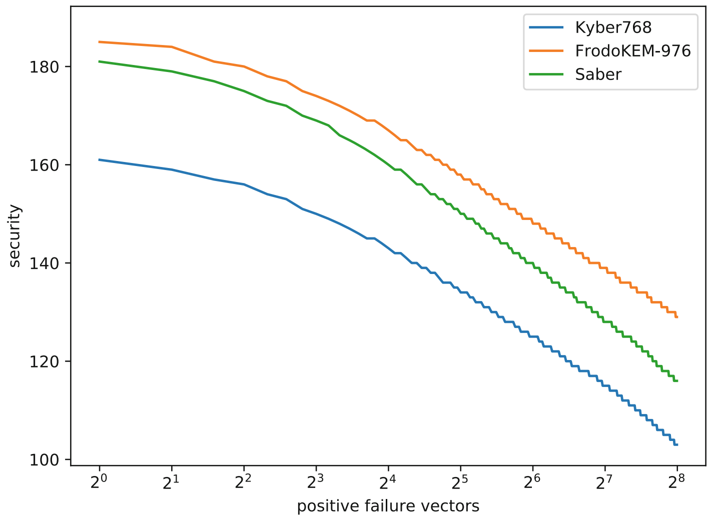

Fig. 7. The hardness of the simplified problem as a function of the number of positive failure vectors

collect i decryption failures using the failure boosting technique, which would cost approximately $i\sqrt{\alpha^{-1}}\beta^{-1}$ work. Then, the exact error position and failure type should be determined for all of the failure vectors using the techniques of Subsect. 4.3. Based on this information, the secret can be estimated, which in turn can be used to simplify the (Ring/Module-)LWE/LWR problem. These last two operations require a negligible amount of work compared to finding decryption failures. Finally, we need to solve the simplified problem, with has a complexity $S_{\text{simplified}}(i)$ as estimated in Sect. 4. The total amount of work is therefore $\mathcal{O}(S_{\text{simplified}}(i) + i\sqrt{\alpha^{-1}}\beta^{-1})$, which is depicted in Fig. 8 as a function of the number of failures i. Note that the practical security of Kyber relies on an error term $\boldsymbol{E}_A$ as well as a rounding term $\boldsymbol{U}_A$. Both are taken into account in the security calculation.

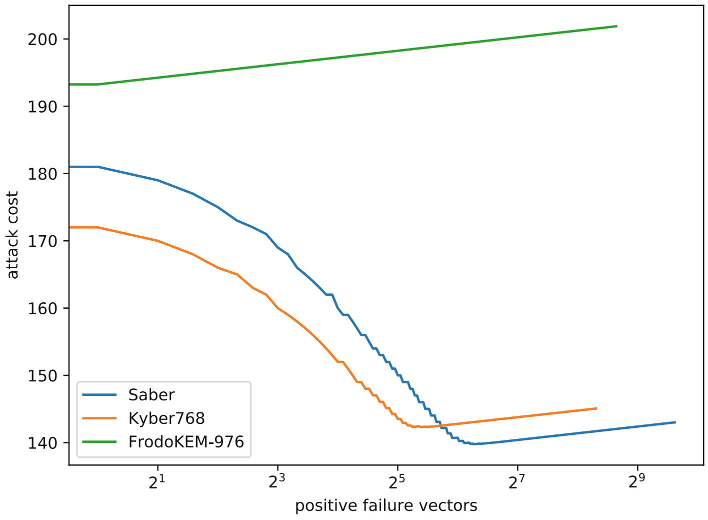

Fig. 8. The full amount of work to break the scheme as a function of the number of collected decryption failures

Table 1 gives an overview of the original hardness of the scheme before decryption failure usage S, and the attack cost $S_{\text{simplified}}(i) + i\sqrt{\alpha^{-1}}\beta^{-1}$ using decryption failures for ideal values of i and f_t, which are calculated through a brute force sweep. The number of collected decryption failures i and the expected number of decryption queries $i\beta^{-1}$ is also included. These values are calculated assuming that the adversary can perform an unlimited number of decryption queries. From this table we can see that the security of Kyber and Saber is considerably reduced. This is due to the fact that finding a failure is easier than breaking the security of the scheme S. For the case of FrodoKEM976, the security is not affected as the work to obtain a failure is considerably larger than breaking the security S.

In other situations such as a multi-target attack or having only a limited number of decryption queries, other values of f_t and i will obtain optimal results. For example in a multi-target attack scenario one would select a higher threshold f_t to be able to efficiently re-use the precomputation work α^{-1} for weak ciphertexts and therefore reduce the overall work. A limit on the number of decryptions n_d could make it necessary to increase the amount of precomputational work α^{-1} in order to reduce the failure rate $\beta^{-1} < n_d/i$. This would make the attack more expensive or might even invalidate it. For example, the NIST Post-Quantum Standardization Process decryption limit is set to 2^{64}, which rules out a decryption failure attack on schemes with a low enough failure rate such as Saber and Kyber, which can be deduced from Fig. 3. As such, the security of this schemes is not affected within the NIST framework.

6 A Weak-Key Attack Model

In this section we elaborate a weak-key (multi-target) attack model when the adversary can only have a limited number of decryption queries to one user but multiple users can be queried. We observe that for certain keys, the error probability can be much higher when applying the failure boosting technique, i.e., choosing 'weak' ciphertexts as discussed in Sect. 3, if the chosen ciphertexts satisfy certain key-related properties. The major targets are the same as before – lattice-based NIST post-quantum proposals with CCA security using some CCA transformations.

We set the maximum number of ciphertexts that can be submitted to a node with a public key to be 2^K and we set the maximum number of public keys in the system to be 2^L. Referring again to the NIST Post-Quantum Standardization Process, they have indicated in their call that at least $K = 64$ can be considered. In the discussion forum [1] for the same project, we have also seen researchers mentioning that $L = 64$ can be considered. We will adopt $K = L = 64$ in the further sections since it seems these values are not questioned, although

Table 1. The security of different schemes with and without decryption failures

	original security	attack cost	reduction factor	decryption failures	decryption queries
Saber	2^{184}	2^{139}	2^{45}	77	2^{131}
FireSaber	2^{257}	2^{170}	2^{87}	233	2^{161}
Kyber768	2^{175}	2^{142}	2^{33}	42	2^{131}
Kyber1024	2^{239}	2^{169}	2^{70}	159	2^{158}
LAC256	2^{293}	$2^{97\dagger}$	2^{196}	$106 \cdot 56$	2^{80}
FrodoKEM976	2^{188}	2^{188}	2^{0}	0	0

† Note that it seems not straightforward for LAC256 to obtain the exact position and type of the errors, which is required to obtain this result larger values of K and L can give more powerful attacks and could definitely be relevant. For example, comparing with attacks on symmetric schemes, such attacks may require a number of plaintext-ciphertext pairs that are close to the number of possible keys (like 2^{200}), and still they are considered valid attacks.

The proposed attack procedure is split in three steps.

1. Do a precomputation step to establish pairs of messages and corresponding ciphertexts and let informally the set $\mathcal{F}$ denote error vectors corresponding to the different messages, which are equivalent to the $(\boldsymbol{C}, \boldsymbol{G})$ pairs chosen before. These selected error vectors should be with particular properties, e.g, with large norm and/or with several large entries in certain positions, etc.
2. Send the ciphertexts contained in $\mathcal{F}$ and assume that we learn the decrypted messages. Assume further that a subset have been erroneously decrypted (wrong decoding due to too large error) and let $\mathcal{F}'$ be the error vectors causing decryption failure. The cardinality of this set could be larger than average if certain properties (related to $\mathcal{F}$) of the secret vector hold. So we submit the set of ciphertexts to each node holding a public key. The node giving the largest decryption failure rate is selected as the target public key for the attack.
3. Do statistical testing on the set $\mathcal{F}'$ (and possibly the set $\mathcal{F}$) to establish relationships between the secret key and given the noise vectors leading to a decryption failure. Analyzing their correlation, we may be able to recover partial secrets, which can considerably reduce the solving complexity of the underlying hard problem. We are then able to perform a full key-recovery attack via classic approaches such as lattice reduction algorithms.

Note that the above procedure is very close to the weak-ciphertext attack described in the previous sections. One major difference is that here we choose the set $\mathcal{F}$ of 'weak' ciphertexts to be related to the 'weak' keys targeted, while in the prior, the 'weak' ciphertexts are chosen to have a larger decryption failure rate averaged over all keys.

We discuss the three steps briefly. In the precomputation step, we can observe a first difference between different schemes. Most schemes include the public key in the generation of the noisy vectors (as input in the hash function generating the noise). This means that a constructed set $\mathcal{F}$ can only be used for a single public key and a new such set must be constructed for the next public key. For simplicity, we assume $|\mathcal{F}| = 2^K$ and note that the number of nodes with a public key is 2^L. If we set the computational complexity of precomputing a set $\mathcal{F}$ to be 2^λ, the overall complexity of this first step is $2^{\lambda+L}$. On the other hand, there are also schemes where error vectors are generated independent of the public key (e.g. LAC). In such a case the same set $\mathcal{F}$ can be used on all public keys and the complexity is only 2^λ. We could also use Grover's search algorithm to accelerate the pre-computation step, as discussed in Sect. 3. However, since the pre-quantum and post-quantum security goals in the NIST Post-Quantum Standardization Process are different for a certain security level, this quantum acceleration may not help us to break the claimed security level of a submission.

For the second step, the idea is that among many public keys, there will be one where the corresponding secret values have a property that causes more decryption errors than on average. So to increase the decryption error probability to a reasonable and detectable level, we consider that a special property in the secret value is held with probability at least p', where $0 < p' < 1$. We then assume that $p' = 2^{-L}$, so we can expect that this special property in the secret value holds for one public key. As mentioned, with respect to the CCA security, NIST restricts to have at most 2^{64} decryption calls to each user (public key). So in order to distinguish a special property in the secret value corresponding to a public key, one needs to get the failure rate for this case to be larger than 2^{-64}.

Finally, in the statistical testing part, we have a set of error vectors that have caused decryption errors. There seems to be a plethora of methods that can be used to recover secret values. For instance, the strong maximum-likelihood approach has been discussed in Sect. 4 and heuristic approaches can also be applied. A general approach that we can adopt is to consider a smaller part of the secret vector under reconstruction, and select the most probable values for this part, based on the observed error vectors in $\mathcal{F}$. Then one combines such guesses for different parts and builds an approximation of a secret vector. A good approximation will mostly be sufficient as it can be used in lattice-basis reduction algorithms.

We note that in many applications, the challenge is to detect the first decryption failure, since we can usually have adaptive approaches to find more failures afterwards with a lower complexity. This idea is further demonstrated in the next section where an adaptive CCA attack on ss-ntru-pke will be presented, and also in a code-based application [34].

7 A Weak-Key Attack on ss-ntru-pke

We have applied the described weak-key approach and provide the details of attacking ss-ntru-pke, a version in the submission to the NIST Post-Quantum Standardization Process – NTRUEncrypt [47]. Connected is also the provably secure NTRU [44] whose security is based purely on the hardness of Ring-LWE. NTRUEncrypt with different parameter choices has been around for a long time and is one of the most competitive lattice-based schemes when it comes to performance.

Note that our attack in this section is in the pre-quantum (classic) security framework due to the different security goal for NIST-V when Grover's algorithm is considered. We adopt the notations from the NTRUEncrypt submission [47] throughout this section.

7.1 The ss-ntru-pke Scheme

ss-ntru-pke is the version of NTRUEncrypt targeting the highest security level, being 256 bits. This scheme achieves CCA2 security via the NAEP transform [27], a transform similar to the Fujisaki-Okamoto transformation with an additional mask. We give a very brief explanation of the scheme. For most of the description and details, we refer to [47]. In the key generation (see Algorithm 6), two secret polynomials $\mathbf{f}, \mathbf{g} \in \mathcal{R}$ are selected, where the coordinates are chosen from a discrete Gaussian $\mathcal{X}_\sigma$ distribution with standard deviation σ. A public key is formed by computing $\mathbf{h} = \mathbf{g}/(p\mathbf{f} + 1)$.

Algorithm 6. ss-ntru-pke.KeyGen

```
Input: Parameter sets PARAM = $\{N, p, q, \sigma\}$ and a seed.
Output: Public key $\mathbf{h}$ and secret key $(\mathbf{f}, \mathbf{g})$.
1) Instantiate Sampler with $\mathcal{X}_\sigma^N$ and seed;
2) $\mathbf{f} \leftarrow$ Sampler, $\mathbf{g} \leftarrow$ Sampler;
3) $\mathbf{h} = \mathbf{g}/(p\mathbf{f} + 1) \mod q$;
```

We show in Algorithm 7 the encryption algorithm of ss-ntru-pke and in Algorithm 8 the decryption algorithm, both from the original proposal [47]. In these descriptions, Hash() represents a hash function, and $\mathcal{B}$ represents a set including all binary polynomials with degree at most $N-1$. The Pad() operation is a function to ensure the message has sufficient entropy, and the Extract() operation is the inverse of Pad().

In each encryption of a message $\mathbf{m}$, two polynomials $\mathbf{r}, \mathbf{e} \in \mathcal{R}$ are generated, where the coordinates are again chosen from a discrete Gaussian distribution $\mathcal{X}_\sigma$ with standard deviation σ. This randomness source uses a seed generated as Hash$(\mathbf{m}, \mathbf{h})$. This means that each choice of a message $\mathbf{m}$ will generate also the polynomials $\mathbf{r}, \mathbf{e} \in \mathcal{R}$. Let us denote this by

$$(\mathbf{r}, \mathbf{e}) = \mathbf{G}(\mathbf{m}, \mathbf{h}).$$

Algorithm 7. ss-ntru-pke.Encrypt

```
Input: Public key $\mathbf{h}$, message msg of length mlen, PARAM and a seed.
Output: Ciphertext $\mathbf{c}$.
1) $\mathbf{m} = \mathrm{Pad}(msg, seed)$;
2) $rseed = \mathrm{HASH}(\mathbf{m}|\mathbf{h})$;
3) Instantiate Sampler with $\mathcal{X}_\sigma^N$ and rseed;
4) $\mathbf{r} \leftarrow$ Sampler, $\mathbf{e} \leftarrow$ Sampler;
5) $\mathbf{t} = p \cdot \mathbf{r} * \mathbf{h}$;
6) $tseed = \mathrm{HASH}(\mathbf{t})$;
7) Instantiate Sampler with $\mathcal{B}$ and tseed;
8) $\mathbf{m}_{mask} \leftarrow$ Sampler;
9) $\mathbf{m}' = \mathbf{m} - \mathbf{m}_{mask} \pmod p$;
10) $\mathbf{c} = \mathbf{t} + p \cdot \mathbf{e} + \mathbf{m}'$;
```

In decryption, with ciphertext $\mathbf{c}$, one computes the message by computing

$$\mathbf{f} * \mathbf{c} = p \cdot \mathbf{r} * \mathbf{g} + p \cdot \mathbf{e} * \mathbf{f} + \mathbf{m}' * \mathbf{f}.$$

A decryption error occurs if $||p \cdot \mathbf{r} * \mathbf{g} + p \cdot \mathbf{e} * \mathbf{f} + \mathbf{m}' * \mathbf{f}||_\infty > q/2$. This basically translates to $||\mathbf{r} * \mathbf{g} + \mathbf{e} * \mathbf{f}||_\infty > q/4$ as $p = 2$ and the last term is much smaller than the first two.

The proposed parameters for ss-ntru-pke for the security level of NIST-V are shown in Table 2. The decoding error probability is estimated to be less than 2^{-80} in [47].

Table 2. Proposed ss-ntru-pke parameters.

N	q	p	$\mathcal{R}$	σ	ϵ	Security
1024	$2^{30} + 2^{13} + 1$	2	$\frac{\mathbb{Z}_q[x]}{x^N+1}$	724	$< 2^{-80}$	V

7.2 The Attack

We now follow the approach of the previous section and describe an attack. The detailed attack is shown in Algorithm 9, where a more efficient CCA2 version is adopted. We define an equivalence relation for two polynomials $u(x)$, $v(x) \in \mathcal{R}$ if $u(x) = x^i \cdot v(x) \pmod{x^N + 1}$, or if $u(x) = -x^i \cdot v(x) \pmod{x^N + 1}$, for $i \in \mathbb{Z}$.

Algorithm 8. ss-ntru-pke.DECRYPT

```
Input: Secret key $\mathbf{f}$, public key $\mathbf{h}$, ciphertext $\mathbf{c}$, and PARAM.
Output: result.
1) $\mathbf{m}' = \mathbf{f} * \mathbf{c} \pmod{p}$;
2) $\mathbf{t} = \mathbf{c} - \mathbf{m}'$;
3) $tseed = \text{HASH}(\mathbf{t})$;
4) Instantiate Sampler with $\mathcal{B}$ and $tseed$;
5) $\mathbf{m}_{mask} \leftarrow$ Sampler;
6) $\mathbf{m} = \mathbf{m}' + \mathbf{m}_{mask} \pmod{p}$;
7) $rseed = \text{HASH}(\mathbf{m}|\mathbf{h})$;
8) Instantiate Sampler with $\mathcal{X}_\sigma^N$ and $rseed$;
9) $\mathbf{r} \leftarrow$ Sampler;
10) $\mathbf{e} = p^{-1}(\mathbf{t} - \mathbf{r} * \mathbf{h})$;
11) if $||\mathbf{e}||_\infty$ is big then
    result = $\perp$;
else
    result = Extract($\mathbf{m}$);
```

Algorithm 9. The CCA2 attack against ss-ntru-pke

```
Input: A number (say $2^{64}$) of public keys.
Output: The secret polynomials $(\mathbf{f}, \mathbf{g})$ of one public key.
1) Collect messages/ciphertexts with special form for all public keys;
2) Submit them for decryption and determine a weak public key $\mathbf{h}$;
1') Prepare messages/ciphertexts with special form for this weak key $\mathbf{h}$;
2') Submit them for decryption and collect the decryption results;
3) Use statistical analysis to have a guess $(\hat{\mathbf{f}}, \hat{\mathbf{g}})$ close to the corresponding secret key $(\mathbf{f}, \mathbf{g})$;
4) Use lattice reduction algorithms to recover the secret key $(\mathbf{f}, \mathbf{g})$;
```

Attack step 1 – pre-computation.

We pick random messages $\mathbf{m}$ and generate corresponding $(\mathbf{r}, \mathbf{e}) = \mathbf{G}(\mathbf{m}, \mathbf{h})$ for a given public key $\mathbf{h}$. We keep only vectors $\mathbf{e}$ equivalent to a polynomial that has the first l (e.g., $l = 2$) positions with the same sign and each with size larger than $c \cdot \sigma$, where c is a constant determining the computational effort of finding such error vectors. These vectors form our chosen set $\mathcal{F}$.

We set $l = 2$ to illustrate the idea in a concrete attack. For one position, the probability that the entry is larger than $c\sigma$ is $1 - \text{erf}(c/\sqrt{2})$. As we can start from any position, the probability to have two consecutive positions with the same sign and entries larger than $c\sigma$ is $p_{\mathbf{e}} = N * (1 - \text{erf}(c/\sqrt{2}))^2/2$. If we set $p_{\mathbf{e}}$ to be 2^{-120}, then c can be as large as 9.193.

Attack step 2 – submit ciphertexts for decryption.

We then send the ciphertexts corresponding to the noise vectors in $\mathcal{F}$ to the decryption algorithm. If the targeted secret key $\mathbf{f}$ is also equivalent to a polynomial that has the first l (e.g., $l = 2$) positions with the same sign and each with size larger than $c_s \cdot \sigma$, where c_s is another constant, then the decoding errors can be detectable. We expect to collect several errors and store their corresponding error vectors $(\mathbf{r}, \mathbf{e})$. The probability to have two consecutive positions with the same sign and entries larger than $c_s\sigma$ is $p_{\mathbf{s}} = N * (1 - \text{erf}(c_{\mathbf{s}}/\sqrt{2}))^2/2$. If we set $p_{\mathbf{s}}$ to be 2^{-64}, then $c_{\mathbf{s}}$ can be as large as 6.802.

If we run 2^{120} precomputation steps for each stored vector with the desired properties, then the overall complexity is 2^{248} since $p_{\mathbf{s}} = 2^{-64}$. Let C_1 denote $2 \cdot c_{\mathbf{s}} c \sigma^2$. We can then have a coefficient in $\mathbf{r} * \mathbf{g} + \mathbf{e} * \mathbf{f}$ whose absolute contribution from these two big entries is at least $C_1 = 2^{25.97}$. We consider the probabilistic behavior of the remaining $(2N - 2)$ positions. As the coefficients of $\mathbf{r}, \mathbf{g}, \mathbf{e}, \mathbf{f}$ are all sampled from a Gaussian distribution with mean 0 and stand deviation $\sigma = 724$, the expected norm of the rest vector in $\mathbf{f}, \mathbf{g}$ with $2N - 2$ entries is about $\sqrt{2N - 2} \cdot \sigma$. Given a public key, $\mathbf{f}, \mathbf{g}$ is fixed. Thus, this coefficient of $\mathbf{r} * \mathbf{g} + \mathbf{e} * \mathbf{f}$ can be approximated as $C_1 + \Phi_0$, where Φ_0 is Gaussian distribution with mean 0 and standard deviation $\sqrt{2N-2} \cdot \sigma^2$. As the error appears when this coefficient is larger than $q/4$, the error probability[3] can be approximated as

$$P_e = \left(1 - \text{erf}(\frac{q/4 - C_1}{\sqrt{2(2N-2)\sigma^2}})\right) \cdot \frac{1}{2}.$$

We obtain a decoding error probability of $2^{-57.3}$ for this example.

Thus we can obtain about $2^{6.7}$ errors from the 2^{64} decryption trails.

An adaptive CCA attack. If we keep the previous setting, i.e., a CCA1 attack, the cost is larger than 2^{248}. However, we can adopt a much more powerful attack model, namely an adaptive CCA (CCA2) attack, consisting of two phases. In the first phase, the attacker spends half of his computational power to determine a weak key; in the later phase, he would put all his remaining resources into attacking this weak key.

To be more specific, we first prepare 2^{63} messages/ciphertexts for each of the 2^{64} public keys. Then we expect two errors corresponding to one key, which can be claimed as a weak key.

We can also reduce the precomputation work for each key to 2^{89}, if there are 2^{64} public keys. We have $c = 7.956$ and the error probability is $2^{-62.0}$, so we expect to have two errors in the testing stage. We then spend 2^{216} work on another precomputation to have 2^{63} messages with c to be 10.351, done only for this weak key. The error probability in the second phase is estimated as $2^{-53.0}$, so we can have 2^{10} errors. The overall complexity is 2^{217}.

Attack step 3 – statistical analysis.

In this step we will try to recover the secret $\mathbf{f}$. Let us first assume that $\mathbf{f}$ has its two big entries in the first two positions of the vector. Then the position in $\mathbf{e} * \mathbf{f}$ where the error occurs, denoted i_0, is the position where the two significant coefficients in $\mathbf{e}$ and those in $\mathbf{f}$ coincide. We now transform each $\mathbf{e}$ in such a way that its two big entries are also to be found in the first two positions. This is done by replacing $\mathbf{e}$ with the corresponding equivalent vector where the two big entries are in the first two positions. Assuming M decryption errors, this now gives us the following knowledge from the received decryption errors:

$$\sum_{i=2}^{N-1} e_i^{(j)} f_i + N_i^{(j)} > q/4 - 2 \cdot c_{\mathbf{s}} c \sigma^2,$$

for $j = 1 \ldots M$ and where $N^{(j)}$ denotes the remaining contribution to the noise. Finally we note that assuming that $\mathbf{f}$ has its two big entries in the first two positions is not a restriction, as such an $\mathbf{f}$ vector will just be an equivalent

[3] The error can occur in both directions. We omit the term $\left(1 - \text{erf}(\frac{q/4+C_1}{\sqrt{2(2N-2)\sigma^2}})\right) \cdot \frac{1}{2}$ as it is negligible compared with $\left(1 - \text{erf}(\frac{q/4-C_1}{\sqrt{2(2N-2)\sigma^2}})\right) \cdot \frac{1}{2}$ for C_1 a very big positive integer.

vector of the true $\mathbf{f}$. So we need only to recover $\mathbf{f}$ and then check all equivalent vectors.

We next show how to derive more knowledge of $\mathbf{f}, \mathbf{g}$ with statistical tools.

A heuristic approach. As we have assumed that the two big entries in $(\mathbf{f}, \mathbf{g})$ (or $(\mathbf{e}, \mathbf{r})$) are the first two entries, we use $\mathbf{K}$ (or $\mathbf{V}_i$ for $1 \leq i \leq M$) to denote a vector consisting of the remaining $2N-2$ entries. Thus, the size of $\mathbf{K}$ (or $\mathbf{V_i}$) can be estimated as $\sqrt{(2N-2)}\sigma$.

We adopt the heuristic assumptions from [19] that all the errors are very close to the folding bound $q/4$, meaning that all the messages leading to an error belong to a hyperplane

$$\mathbf{V}_i \cdot \mathbf{K} = \frac{q}{4} - C_1,$$

where C_1 is the contribution from the two significant entries.

Thus, the mean vector $\hat{\mathbf{V}}$ of $\mathbf{V_i}$ should be close to a scaled vector of $\mathbf{K}$, i.e.,

$$\hat{\mathbf{V}} = \frac{\sum_{i=1}^{M} \mathbf{V}_i}{M} \approx \frac{q/4 - C_1}{\|\mathbf{K}\|^2}\mathbf{K}.$$

We can have an estimation $\hat{\mathbf{K}} = \frac{(2N-2)\sigma^2}{q/4-C_1}\hat{\mathbf{V}}$. If we round the entries of $\mathbf{K}$ to the nearest integer in $\mathbb{Z}_q$, we obtain an estimation $(\hat{\mathbf{f}}, \hat{\mathbf{g}})$ of the secret vector $(\mathbf{f}, \mathbf{g})$.

The remaining question is how good this estimation can be? We heuristically answer this question using the central limit theorem.

Each observation $\mathbf{V_i}$ with approximated norm $\sqrt{2N-2}\sigma$ can be viewed as the summation of the signal point

$$\frac{q/4 - C_1}{\|\mathbf{K}\|^2}\mathbf{K},$$

and a noise vector with squared norm

$$(2N-2)\sigma^2 - \frac{(q/4 - C_1)^2}{(2N-2)\sigma^2}.$$

By the central limit theorem, if we have M observations, then the squared norm (variance) of the noise can be reduced by a factor of M. Hence, the error norm should be

$$\sqrt{\frac{1}{M} \cdot \left((2N-2)\sigma^2 - \frac{(q/4 - C_1)^2}{(2N-2)\sigma^2}\right)}.$$

As we consider $\hat{\mathbf{K}}$ instead of $\hat{\mathbf{V}}$, the true error norm should be resized as

$$\frac{(2N-2)\sigma^2}{q/4 - C_1} \cdot \sqrt{\frac{1}{M} \cdot \left((2N-2)\sigma^2 - \frac{(q/4 - C_1)^2}{(2N-2)\sigma^2}\right)}. \tag{28}$$

Using this formula, we can have a candidate with error norm $0.169\sqrt{2N-2}\sigma$, assuming that 1024 errors have been collected.

Table 3. The simulated error rates v.s. the estimated error rates.

q	error rate	
	-estimated-	-simulated-
$q = 2^{29}$	$2^{-9.05}$	$2^{-9.19}$
$q = 2^{29} + 2^{26}$	$2^{-12.64}$	$2^{-12.96}$
$q = 2^{29} + 2^{27}$	$2^{-16.91}$	$2^{-17.09}$
$q = 2^{29} + 2^{27} + 2^{26} + 2^{25}$	$2^{-24.62}$	$2^{-24.57}$

Attack step 4 – lattice reduction.

If $(\Delta\mathbf{f}, \Delta\mathbf{g}) = (\mathbf{f}, \mathbf{g}) - (\hat{\mathbf{f}}, \hat{\mathbf{g}})$ is small, we can recover it using lattice reduction algorithms efficiently. Thus, we obtain the correct value of $(\mathbf{f}, \mathbf{g})$.

If we have the error size to be only $0.169\sqrt{2N-2}\sigma$, as assumed in the previous step, using the LWE estimator from Albrecht et al. [4], it takes about 2^{181} time and 2^{128} memory if one uses sieving to implement the SVP oracle in BKZ. Though the authors of [47] discussed about memory constraint for applying sieving in lattice-based cryptanalysis, we believe it is reasonable to assume for 2^{128} memory when considering a scheme targeting the classic 256-bit security level. Another possibility is to implement the SVP oracle using tuple sieving, further reducing the memory complexity to 2^{117}. The time complexity then increases to 2^{223}, but still far from achieving the claimed 256-security level.

7.3 Experimental Results

We have implemented some of the parts of the attack to check performance against theory. We have chosen exactly the same parameters in ss-ntru-pke as well as in the attack, except for the q value, which in the experiment was set to the values shown in Table 3. The reason being that is we wanted to lower the decryption error rate so that simulation was possible.

We put two consecutive entries in $\mathbf{f}$ each of size $6.2 \cdot \sigma$ and we generated error vectors with two large positive entries each of size $9.2 \cdot \sigma$. For such choice, we first verified the decryption error probabilities, as seen in Table 3. These match the theoretical results well.

Table 4. The simulated error norm v.s. the estimated error norm. $(M = 1024)$

q	error norm $/(\sqrt{2N-2}\sigma)$	
	-estimated-	-simulated-
$q = 2^{29}$	0.487	0.472
$q = 2^{29} + 2^{26}$	0.391	0.360
$q = 2^{29} + 2^{27}$	0.326	0.302
$q = 2^{29} + 2^{27} + 2^{26} + 2^{25}$	0.261	0.250

Table 5. The simulated error norm v.s. the estimated error norm. ($q = 2^{29} + 2^{27} + 2^{26} + 2^{25}$)

M	error norm $/(\sqrt{2N-2}\sigma)$ -estimated-	-simulated-
$M = 256$	0.522	0.490
$M = 512$	0.369	0.348
$M = 1024$	0.261	0.250
$M = 1536$	0.213	0.212

For each choice of q we then collected up to $M = 2^{10}+2^9 = 1536$ error vectors and processed them in a statistical analysis step, to get a good approximation of $(\mathbf{f}, \mathbf{g})$. As the heuristic approach described, we first created an approximation of $(\mathbf{f}, \mathbf{g})$, say denoted by $(\hat{\mathbf{f}}, \hat{\mathbf{g}})$, by simply computing $\hat{f}_i = E \cdot \frac{\sum_{j=0}^{M-1} e_i^{(j)}}{M}$ as the value in the ith position. Here E is a constant that makes the norm of the vector to be as the expected norm of $\mathbf{f}$. Clearly, this is a very simple way of exploring the dependence between f_i and e_i, but still it seems to be sufficient.

We have plotted the simulated error norms for various q and M in Figure 9. Furthermore, we show in Tables 4 and 5 the comparison between the simulated error norms and the estimated error norms according to Eq. 28.

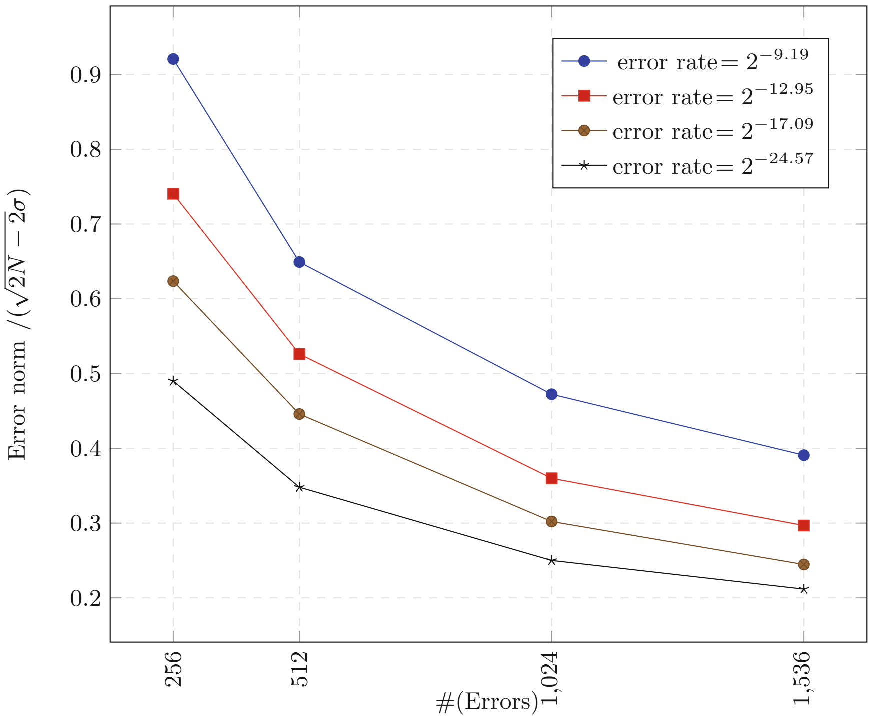

Fig. 9. Error norm as a function of the number of collected error vectors.

In the prior table, M is fixed to 1024 and q varies, while in the latter table, q is fixed to $2^{29}+2^{27}+2^{26}+2^{25}$ and M varies. We see that in all the cases, the simulated data match the estimated data well, though the simulation seems always better than the estimation, i.e., with smaller error norms. Another observation from Table 5 is that the estimation using the central limit theorem becomes more accurate when M becomes larger, which is also very reasonable.

7.4 Summarizing the Attack

The best attack is a CCA2 type attack where we in precomputation use $2^{89+63} = 2^{152}$ operations to derive 2^{63} special ciphertexts that are submitted for decryption. With probability 2^{-64} the secret $\mathbf{f}$ has the desired property of two consecutive big entries. If so, we will most likely see several decoding errors and such a weak key has been detected. When the weak key has been detected, we perform yet another precomputation that uses 2^{216} operations to derive 2^{63} additional special ciphertexts again submitted for decryption. We receive in expectation 1024 decryption errors and the knowledge from the error vectors will allow us to reconstruct $\mathbf{f}$ without too much trouble using lattice reduction algorithms, as experimental results strongly indicated. The overall complexity is thus approximately 2^{217} if the SVP oracle in BKZ is implemented via lattice sieving. Actually, the cost of the lattice reduction algorithms in the final stage is not the bottleneck, since we can employ other powerful statistical tools in Step 3 (e.g., the Maximum Likelihood Test approach) to make this cost negligible.

8 Conclusion

In this paper we introduced a method to increase the decryption failure rate of a scheme, based on the search for 'weak' ciphertexts. This method benefits an adversary in at least three scenarios: if he has access to a quantum computer, if he can only perform a limited number of decryption queries or if he wants to stage a multi-target attack on schemes that do not have the appropriate protection. We explicitly calculated the effect of failure boosting in these scenarios for various (Ring/Module-)LWE/LWR schemes. We also proposed a method to estimate the secret key given ciphertexts that lead to decryption failures. The remaining security after a certain number of decryption failures was calculated, given the exact location of the error. We suggested three methods to obtain the exact location of errors in failing ciphertexts. Finally, we estimated the security of several schemes under an attack that optimally uses these decryption failures and show that for some schemes the security is drastically reduced if an attacker can perform sufficient decryption queries. However, for most NIST post-quantum standardization candidates, the expected number of required decryption queries is too high for a practical attack. We also identify the changes to this attack under a multi-target scenario or when an attacker has only access to a limited number of decryption queries.

We further proposed a generic weak-key attack model against lattice-based schemes, which is slightly different from the previous attack, based on the observation that the error probability can be much higher for certain 'weak' keys. We applied this model to attacking ss-ntru-pke, a version in the NTRUEncrypt submission to the NIST Post-Quantum Standardization Process. Specifically, we have presented an adaptive CCA attack on the claimed 256-bit classic security level (NIST-V) of ss-ntru-pke. This attacking idea can be treated as extension of reaction attacks [16,22] that already jeopardize the CCA security of MDPC and LDPC based crypto-systems.

Acknowledgements. The authors would like to thank Tancrède Lepoint and the anonymous reviewers for their helpful comments. They would also like to thank Andreas Hülsing for interesting discussions. This work was supported in part by the Research Council KU Leuven: C16/15/058, by the European Commission through the Horizon 2020 research and innovation programme Cathedral ERC Advanced Grant 695305, by the Research Council KU Leuven grants C14/18/067 and STG/17/019, by the Norwegian Research Council (Grant No. 247742/070), by the Swedish Research Council (Grant No. 2015-04528), by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and by the Swedish Foundation for Strategic Research (SSF) project RIT17-0005.

References

1. NIST Post-Quantum Cryptography Forum. https://groups.google.com/a/list.nist.gov/forum/#!forum/pqc-forum. Accessed 11 Jan 2019
2. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2016). https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf
3. Albrecht, M., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. **9**, 169–203 (2015)
4. Albrecht, M.R., et al.: Estimate all the LWE, NTRU schemes! Cryptology ePrint Archive, Report 2018/331 (2018). https://eprint.iacr.org/2018/331
5. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange – a new hope. In: USENIX Security 2016 (2016)
6. Baan, H., et al.: Round2: KEM and PKE based on GLWR. Cryptology ePrint Archive, Report 2017/1183 (2017). https://eprint.iacr.org/2017/1183
7. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_42
8. Bernstein, D.J., Bruinderink, L.G., Lange, T., Panny, L.: HILA5 pindakaas: on the CCA security of lattice-based encryption with error correction. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 2018. LNCS, vol. 10831, pp. 203–216. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89339-6_12
9. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: On symmetric encryption with distinguishable decryption failures. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 367–390. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43933-3_19

10. Bos, J., et al.: CRYSTALS − Kyber: a CCA-secure module-lattice-based KEM. Cryptology ePrint Archive, Report 2017/634 (2017). http://eprint.iacr.org/2017/634
11. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th Annual ACM Symposium on Theory of Computing, 1–4 June 2013, pp. 575–584. ACM Press, Palo Alto (2013)
12. Cheon, J.H., Kim, D., Lee, J., Song, Y.: Lizard: cut off the tail! Practical post-quantum public-key encryption from LWE and LWR. Cryptology ePrint Archive, Report 2016/1126 (2016). http://eprint.iacr.org/2016/1126
13. D'Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 2018. LNCS, vol. 10831, pp. 282–305. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89339-6_16
14. D'Anvers, J.P., Vercauteren, F., Verbauwhede, I.: On the impact of decryption failures on the security of LWE/LWR based schemes. Cryptology ePrint Archive, Report 2018/1089 (2018). https://eprint.iacr.org/2018/1089
15. Ding, J., Alsayigh, S., Saraswathy, R.V., Fluhrer, S., Lin, X.: Leakage of signal function with reused keys in RLWE key exchange. Cryptology ePrint Archive, Report 2016/1176 (2016). http://eprint.iacr.org/2016/1176
16. Fabsic, T., Hromada, V., Stankovski, P., Zajac, P., Guo, Q., Johansson, T.: A reaction attack on the QC-LDPC McEliece cryptosystem. Cryptology ePrint Archive, Report 2017/494 (2017). http://eprint.iacr.org/2017/494
17. Fluhrer, S.: Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085 (2016). https://eprint.iacr.org/2016/085
18. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_34
19. Gama, N., Nguyen, P.Q.: New chosen-ciphertext attacks on NTRU. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 89–106. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_7
20. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, pp. 212–219. STOC 1996. ACM, New York (1996). https://doi.org/10.1145/237814.237866
21. Guo, Q., Johansson, T., Nilsson, A.: A generic attack on lattice-based schemes using decryption errors with application to ss-ntru-pke. Cryptology ePrint Archive, Report 2019/043 (2019). https://eprint.iacr.org/2019/043
22. 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. Part I. LNCS, vol. 10031, pp. 789–815. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_29
23. 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
24. Hoffstein, J., Silverman, J.H.: NTRU Cryptosystems Technical Report Report# 016, Version 1 Title: Protecting NTRU Against Chosen Ciphertext and Reaction Attacks

25. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. Part I. LNCS, vol. 10677, pp. 341–371. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_12
26. 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
27. 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
28. Jaulmes, É., Joux, A.: A chosen-ciphertext attack against NTRU. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 20–35. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_2
29. Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: Post-quantum IND-CCA-secure KEM without additional hash. Cryptology ePrint Archive, Report 2017/1096 (2017). https://eprint.iacr.org/2017/1096
30. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Des. Codes Crypt. **75**(3), 565–599 (2015). https://doi.org/10.1007/s10623-014-9938-4
31. Lu, X., Liu, Y., Jia, D., Xue, H., He, J., Zhang, Z.: LAC. Technical report, National Institute of Standards and Technology (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
32. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_1
33. Naehrig, M., et al.: Frodokem. Technical report, National Institute of Standards and Technology (2017). https://frodokem.org/files/FrodoKEM-specification-20171130.pdf
34. Nilsson, A., Johansson, T., Stankovski, P.: Error amplification in code-based cryptography. IACR Trans. Crypt. Hardw. Embed. Syst. **2019**(1), 238–258 (2019). https://doi.org/10.13154/tches.v2019.i1.238-258
35. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, pp. 333–342. STOC 2009. ACM, New York (2009). https://doi.org/10.1145/1536414.1536461
36. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th Annual ACM Symposium on Theory of Computing, 22–24 May 2005, pp. 84–93. ACM Press, Baltimore (2005)
37. Saarinen, M.J.O.: HILA5. Technical report, National Institute of Standards and Technology (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
38. Saito, T., Xagawa, K., Yamakawa, T.: Tightly-secure key-encapsulation mechanism in the quantum random oracle model. Cryptology ePrint Archive, Report 2017/1005 (2017). https://eprint.iacr.org/2017/1005
39. Schanck, J.M., Hulsing, A., Rijneveld, J., Schwabe, P.: NTRU-HRSS-KEM. Technical report, National Institute of Standards and Technology (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
40. Schwabe, P., et al.: CRYSTALS-Kyber. Technical report, National Institute of Standards and Technology (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

41. Schwabe, P., et al.: Newhope. Technical report, National Institute of Standards and Technology (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
42. Seo, M., Park, J.H., Lee, D.H., Kim, S., Lee., S.J.: Emblem and R.Emblem. Technical report, National Institute of Standards and Technology (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
43. Smart, N.P., et al.: LIMA. Technical report, National Institute of Standards and Technology (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
44. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_4
45. Szepieniec, A.: Ramstake. Technical report, National Institute of Standards and Technology (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
46. Targhi, E.E., Unruh, D.: Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 192–216. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_8
47. Zhang, Z., Chen, C., Hoffstein, J., Whyte, W.: NTRUEncrypt. Technical report, National Institute of Standards and Technology (2017). https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions