Misuse Attacks on Post-Quantum Cryptosystems

Ciprian Băetu, F. Betül Durak, Loïs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay

EPFL
CH-1015 Lausanne, Switzerland
http://lasec.epfl.ch

Abstract. Many post-quantum cryptosystems which have been proposed in the National Institute of Standards and Technology (NIST) standardization process follow the same meta-algorithm, but in different algebras or different encoding methods. They usually propose two constructions, one being weaker and the other requiring a random oracle. We focus on the weak version of nine submissions to NIST. Submitters claim no security when the secret key is used several times. In this paper, we analyze how easy it is to run a key recovery under multiple key reuse. We mount a classical key recovery under plaintext checking attacks (i.e., with a plaintext checking oracle saying if a given ciphertext decrypts well to a given plaintext) and a quantum key recovery under chosen ciphertext attacks. In the latter case, we assume quantum access to the decryption oracle.

1 Introduction

By anticipating that quantum computers will eventually compute discrete logarithms and integer factorization [22], all public key cryptosystems which are being used today will break down. There is an urgent need to replace them. For this purpose, the US National Institute of Standards and Technology (NIST) initiated a standardization process for post-quantum algorithms. The call for proposals expired in 2017 resulting with many submissions. Among these, only a very few types of algorithms are proposed such as lattice-based, code-based, hash-based, or isogeny-based. One of the most promising algorithms is lattice-based.

Many of the lattice-based cryptosystems are inspired by the Regev cryptosystem [20] such as the Lyubashevsky-Peikert-Regev cryptosystem [18]. We can easily extract a common pattern in proposals which followed them. All proposed constructions require never to reuse the secret key because of possible attacks. In other words, the proposed cryptosystems are ephemeral in the sense that the secret key is meant to be used only once. The approach of designers is to start from this ephemeral construction and then to transform it into a strongly secure (i.e. with reusable keys) key encapsulation mechanism (KEM) by using the Fujisaki-Okamoto transformation or one of its variants [12,13,15,25]. What

transformations share in common is that they imply a computation overhead. Concretely, after normal decryption, a re-encryption is made to check if the ciphertext was correctly formed. This re-encryption does not seem so useful to non-experts. Additionally, these transformations need a random oracle which has no practical existence.

When there is a cryptosystem which is practical but comes with a warning and an extension which looks only motivated by academic people, we believe that users will eventually try to use the weakly secure cryptosystems and pay little attention to the warning, or even misunderstand the strengthened version, just because the threat is not clear. For this reason, we should understand what are the risks under misuse of keys.

Another observation was made by Lepoint [17]. He checked that several implementations of the strongly secure KEM have side channels leaking the result of the decryption under the weak cryptosystem. It comes from a misimplementation of the Fujisaki-Okamoto (FO) transform. In the FO transformation, the decryption is done, then the verification checks that the ciphertext is well-formed. The result is released only if the test passes. However, Lepoint has shown that side channels in implementations were leaking the result in any case, no matter whether the ciphertext was well-formed or not.

In 2015, the NSA [16] reported some concerns about recurring problems with key leakage in key agreement protocols. They suggested to explicitly check that ciphertexts are well-formed by using the FO transform. This recommendation was followed by designers, as mentioned above.

In 2016, Fluhrer [11] published an attack based on the key reuse. In his attack, an adversary encrypts a message by deviating a bit from the protocol. Then, he sends the ciphertext for decryption and checks if the decryption matches what he expected. After a few trials, the adversary recovers the secret key. The attack applies to all protocols using a special signaling (a.k.a. error-reconciliation) function. In 2017, Ding et al. [10] expanded this attack to a class of key agreement protocols based on ring-LWE with signaling. Our goal is to apply the Fluhrer attack model to more protocols and to minimize the number of key reuse to recover the key and to be able to assess how weak those protocols are under key reuse.

At CT-RSA'2019, Bauer et al. [5] presented an attack on the weak version of NewHope-CPA-PKE [1]. For $n = 1\,024$, they recover the secret with high probability using 2^{14} queries to a *key mismatch oracle*, what we herein call a *plaintext checking oracle* (PCA).

Our contribution. In this paper, we first define the meta-structure of constructions with ephemeral keys. Then, we formalize the noise learning problem which is required to break these constructions. We identify optimal bounds in terms of the number of oracle calls to solve this problem. Then, we mount a classical key recovery under plaintext checking attack (KR-PCA), which is also the model of Fluhrer attacks [11]. This model makes sense when an adversary can play with a server with a modified ciphertext and check if it still decrypts to the same plaintext as before. Compared to Fluhrer [11], we apply it to different classes

of protocols, we optimize the number of oracle calls, and we identify the link with the noise learning problem. We give optimal lower bounds for the number of oracle calls to solve this problem. We finally propose a quantum variant of the attack in an "imaginary" model where the adversary has quantum access to a decryption oracle. This is a key recovery under chosen ciphertext attack (KR-CCA). Our quantum attack is based on the GKZ algorithm [14] to solve LWE from a superposition of inputs. We also adapt the AJOP attack [2] based on the Bernstein-Vazirani algorithm [7] to solve the LPN problem from a superposition of inputs.[1] The AJOP-based attack has better performances but is more restrictive. It only works for cryptosystems of a special form and it also assumes a quantum decryption oracle working with addition in a special group instead of the XOR. Our result shows that a single use of the key leads to a full or partial key recovery with a probability of success proving the attacks are a big threat.

Table 1. Attacks on post-quantum cryptosystems. For two types of attacks (classical KR-PCA and quantum KR-CCA), we report the number of oracle calls as O, the probability of success as P, the number of collected linear equations in $\mathbf{Z}_q$ as E, and the number of unknowns in $\mathbf{Z}_q$ as U. We also indicate for information the expected total number $T = \frac{OU}{PE}$ of oracle calls obtained to recover the full key with probability 1 by iterating the attacks.

		classical KR-PCA attack				GKZ-based quantum KR-CCA attack				AJOP-based quantum KR-CCA attack			
	U	O	P	E	(T)	O	P	E	(T)	O	P	E	(T)
EMBLEM128	2^{10}	2^9	1	2^5	(2^{14})	2	2^{-16}	2^{10}	(2^{17})	1	1	2^{10}	(1)
R.EMBLEM128	2^9	2^{13}	1	2^9	(2^{13})	2	2^{-24}	2^9	(2^{25})	1	2^{-1}	2^9	(2)
Frodo-640	2^{12}	2^{10}	1	2^6	(2^{16})	2	2^{-13}	2^9	(2^{17})	1	2^{-2}	2^{12}	(2^2)
KINDI256	2^{10}	2^{12}	1	2^8	(2^{14})	2	2^{-14}	2^{10}	(2^{15})	1	2^{-1}	2^{10}	(2)
Lepton Light I	2^{13}	2^{13}	1	2^{12}	(2^{14})	-	-	-	-	-	-	-	-
LIMA227-2p	2^{10}	2^{14}	1	2^{10}	(2^{14})	2	2^{-17}	2^{10}	(2^{18})	1	2^{-1}	2^{10}	(2)
LIMA152-sp	2^{10}	2^{15}	1	2^{10}	(2^{15})	2	2^{-24}	2^{10}	(2^{25})	1	2^{-1}	2^{10}	(2)
Lizard536	2^{17}	-	-	-	-	2	2^{-9}	2^9	(2^{18})	1	2^{-1}	2^9	(2^9)
RLizard536	2^{10}	-	-	-	-	2	2^{-8}	2^{10}	(2^9)	1	2^{-1}	2^{10}	(2)
LOTUS128	2^{16}	2^{11}	1	2^7	(2^{20})	2	2^{-13}	2^9	(2^{21})	1	2^{-1}	2^9	(2^8)
NewHope512	2^9	-	-	-	-	2	2^{-28}	2^9	(2^{29})	1	2^{-1}	2^9	(2)
TitaniumStd128	2^{11}	2^{12}	1	2^8	(2^{15})	2	2^{-16}	2^{10}	(2^{18})	1	2^{-1}	2^{10}	(2^2)

We report our results for both types of attacks in Table 1. *For information only*, the table indicates the total number of oracle calls we need, by iterating, to recover the full key with probability 1. We should, however, take this measurement with care. This is because running a single instance of the attack may

[1] The AJOP attack was released after we submitted this paper. For completeness, we include its adaptation here.

be enough to decrease the security of the cryptosystem and to recover the secret by other means.

2 A Meta-PKC Construction

We define a cryptosystem as follows.

Definition 1 (Public-key cryptosystem). *A* public-key cryptosystem (PKC) *with security parameter* λ *consists of four algorithms:*

- $\mathsf{PKC.setup}(1^\lambda; \mathsf{coinS}) \xrightarrow{\$} \mathsf{pp}$
- $\mathsf{PKC.gen}(\mathsf{pp}; \mathsf{coinA}) \xrightarrow{\$} (\mathsf{sk}, \mathsf{pk})$
- $\mathsf{PKC.enc}(\mathsf{pp}, \mathsf{pk}, \mathsf{pt}; \mathsf{coinB}) \xrightarrow{\$} \mathsf{ct}$
- $\mathsf{PKC.dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct}) \to \mathsf{pt}'$

Correctness *implies that for any* pt*, by running all these algorithms, we obtain* $\mathsf{pt} = \mathsf{pt}'$ *with probability* $1 - \mathsf{negl}(\lambda)$*, over the random selection of the coins.*

In this paper, we consider two types of security notions.

Definition 2 (KR-PCA and KR-CCA). *We use the key recovery game with oracle* $\mathcal{O}$ *of Fig. 1. We consider two types of oracles:* PCO *and* DecO *which are defined on the figure. The KR-PCA game uses* $\mathcal{O} = \mathsf{PCO}$*. The KR-CCA game uses* $\mathcal{O} = \mathsf{DecO}$*.*

$\mathsf{PCO}(\mathsf{ct}, \mathsf{pt})$ is a plaintext checking oracle which receives ct and pt, runs the decryption and only returns one bit saying if it decrypts to pt. KR-PCA is an adaptive key recovery attack. Security against KR-PCA is implied by IND-CCA security. KR-PCA attacks are *not* in the IND-CPA security framework. Hence, a PKC could be IND-CPA secure but still vulnerable to a KR-PCA attack.

Game $\mathsf{KR}^{\mathcal{O}}_{\mathcal{A}}(\lambda)$:
1: pick coinS, coinA
2: $\mathsf{setup}(1^\lambda; \mathsf{coinS}) \to \mathsf{pp}$
3: $\mathsf{gen}(\mathsf{pp}; \mathsf{coinA}) \to (\mathsf{sk}, \mathsf{pk})$
4: $\mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pp}, \mathsf{pk}) \to \mathsf{sk}'$
5: **return** $1_{\mathsf{sk}=\mathsf{sk}'}$

Oracle $\mathsf{PCO}(\mathsf{ct}, \mathsf{pt})$:
1: $\mathsf{dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct}) \to \mathsf{pt}'$
2: **return** $1_{\mathsf{pt}'=\mathsf{pt}}$

Oracle $\mathsf{DecO}(\mathsf{ct})$:
3: **return** $\mathsf{dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$

Fig. 1. KR-PCA and KR-CCA games.

The PCA model makes sense in several cases. For instance, in the client-server protocol where the encryption is used to transport a symmetric key to start secure messaging, an adversary can try to encrypt a symmetric key by deviating

from the protocol. He generates malformed ciphertexts which may decrypt to the chosen symmetric key or not. By sending the malformed ciphertext to the server, the adversary can easily see if secure messaging with the server is possible, hence simulate a PCO oracle. Clearly, it is devastating that such an attack would lead to a key recovery.

We define the following algebra. We consider six *additive Abelian groups* S_{sk}, S_A, S_B, S_t, S_U, and S_V and four *bilinear mappings* which are all denoted with $\times$. The four bilinear mappings have domains $S_A \times S_{\mathsf{sk}} \to S_B$, $S_U \times S_{\mathsf{sk}} \to S_V$, $S_t \times S_A \to S_U$, and $S_t \times S_B \to S_V$. We assume associativity in the sense that

$$(t \times A) \times \mathsf{sk} = t \times (A \times \mathsf{sk})$$

for all $t \in S_t$, $A \in S_A$, and $\mathsf{sk} \in S_{\mathsf{sk}}$. Hence, multiplication works as in the diagram on Fig. 2.

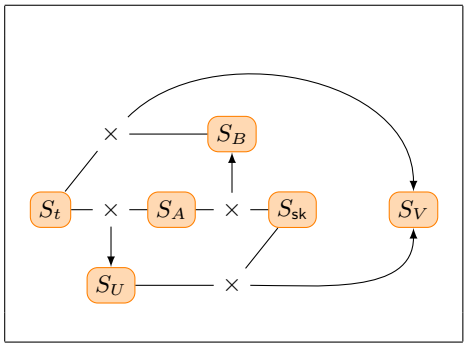

Fig. 2. Algebra: the four bilinear functions on the six spaces. For instance, one element of S_t multiplied by one element of S_A gives one element of S_U.

We also assume that there is a norm $\|\cdot\|$ on S_{sk}, S_B, S_t, S_U, and S_V (i.e. all spaces except S_A for which we need no norm). By definition, the norm is positive and satisfies the triangular inequality. We assume that we can upper bound $\|x \times y\|$ in terms of $\|x\|$ and $\|y\|$ for the four bilinear functions.

Finally, we assume two functions $\mathsf{encode} : \mathcal{M} \to S_V$ and $\mathsf{decode} : S_V \to \mathcal{M}$ such that encode is injective. The image set $C = \mathsf{encode}(\mathcal{M})$ is called a *code*. Elements of the code are *codewords*. The packing radius of C is denoted as ρ_- and the covering radius of C is denoted as ρ_+. Around every $W \in S_V$, balls of radius ρ_- contain no more than one codeword and balls of radius ρ_+ contain at least one codeword:

$$\forall V, V' \in C \quad \forall W \in S_V \quad (\|V - W\| \le \rho_- \wedge \|V' - W\| \le \rho_-) \Longrightarrow V = V'$$
$$\forall W \in S_V \quad \exists V \in C \quad \|V - W\| \le \rho_+$$

Additionally,

$$\forall W \in S_V \quad \mathsf{decode}(W) = \arg\min_{\mathsf{pt}} \|W - \mathsf{encode}(\mathsf{pt})\|$$

in the sense that $\mathsf{encode}(\mathsf{decode}(W))$ is *one* closest codeword to W (which may be ambiguous if there is no unique closest codeword). This implies

$$\forall W \in S_V \quad \forall \mathsf{pt} \quad \|W - \mathsf{encode}(\mathsf{pt})\| \leq \rho_- \Longrightarrow \mathsf{decode}(W) = \mathsf{pt}$$
$$\forall W \in S_V \quad \|W - \mathsf{encode}(\mathsf{decode}(W))\| \leq \rho_+$$

An element δ such that $\|\delta\| \leq \rho_-$ will be called *sparse*. In what follows, we use small letters to designate sparse elements.

Algorithm $\mathsf{setup}(1^\lambda)$:
1: set up the algebra and define pp
2: **return** pp

Algorithm $\mathsf{gen}(\mathsf{pp}; \mathsf{coinA})$:
3: pick a random $A \in S_A$ and random *sparse* $\mathsf{sk} \in S_{\mathsf{sk}}$ and $d \in S_B$ by using coinA
4: $B \leftarrow A \times \mathsf{sk} + d$
5: $\mathsf{pk} \leftarrow (A, B)$
6: **return** $(\mathsf{sk}, \mathsf{pk})$

Algorithm $\mathsf{enc}(\mathsf{pp}, \mathsf{pk}, \mathsf{pt}; \mathsf{coinB})$:
1: parse $\mathsf{pk} = (A, B)$
2: pick random *sparse* $t \in S_t$, $e \in S_U$, and $f \in S_V$ by using coinB
3: $U \leftarrow t \times A + e$
4: $V \leftarrow t \times B + f + \mathsf{encode}(\mathsf{pt})$
5: **return** $\mathsf{ct} = (U, V)$

Algorithm $\mathsf{dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$:
6: parse $\mathsf{ct} = (U, V)$
7: $W \leftarrow V - U \times \mathsf{sk}$
8: $\mathsf{pt}' \leftarrow \mathsf{decode}(W)$.
9: **return** pt'

Fig. 3. The meta-cryptosystem defined on the algebra.

We define a PKC as on Fig. 3 in which the choice of the algebra, norm, encoding/decoding, and the probability distributions are left free. Thanks to bilinearity and associativity, we have $W = \delta + \mathsf{encode}(\mathsf{pt})$ with

$$\delta = t \times d + f - e \times \mathsf{sk} \tag{1}$$

This value δ will be called the *noise*. By controlling the size of t, d, f, e, sk (with their respective probability distribution), we can make sure that the noise δ is sparse. Hence, $\mathsf{decode}(W) = \mathsf{pt}$.

In what follows, for an element $X \in \mathbf{Z}_q$, $|X|$ is the absolute value of $X \in \mathbf{Z}_q$ when represented modulo q such that $-\frac{q}{2} \leq X \leq \frac{q}{2}$. As an example of norm over $\mathbf{Z}_q^n$, we can consider the L_∞ norm $\|X\| = \max(|X_1|, \ldots, |X_n|)$, or the L_1 norm $\|X\| = |X_1| + \cdots + |X_n|$, or a combination of both $\|X\| = \max_i \sum_j |X_{i,j}|$.

As an example of $\mathsf{encode}/\mathsf{decode}$ with the L_∞ norm, we could consider $\mathcal{M} = \{0, 1, \ldots, z-1\}^n$ (z is the alphabet size) and $\mathsf{encode}(\mathsf{pt}) = L \cdot \mathsf{pt}$ in $S_V = \mathbf{Z}_q^n$, for a positive integer L such that $L > 2\rho_-$ and $zL \leq q$. We define $\mathsf{decode}(X) = \lfloor \frac{X}{L} \rceil$ component-wise when the coordinates of X are taken modulo q.

We only give three examples below. More are provided in Appendix A.

Example 3 (Frodo). FrodoPKE [8] works with $S_{\mathsf{sk}} = S_B = \mathbf{Z}_q^{n\bar{n}}$, $S_A = \mathbf{Z}_q^{n^2}$, $S_t = S_U = \mathbf{Z}_q^{\bar{m}n}$, and $S_V = \mathbf{Z}_q^{\bar{m}\bar{n}}$ with the L_∞ norm and $\mathcal{M} = \{0,1\}^{\ell\bar{m}\bar{n}}$. It uses $q = 2^{15}$, $\ell = 2$, $\bar{m} = \bar{n} = 8$, and $n = 640$ (for the Frodo-640 parameters). The bilinear mappings are matrix multiplications. I.e., elements of $S_{\mathsf{sk}} = S_B$ are $n \times \bar{n}$ matrices, elements of S_A are $n \times n$ matrices, elements of $S_t = S_U$ are $\bar{m} \times n$ matrices, and elements of S_V are $\bar{m} \times \bar{n}$ matrices. Frodo uses the L_∞ norm (when considering elements as vectors). Encoding ℓ bits per $\mathbf{Z}_q$ elements is done by taking them as an integer and multiplying them by $L = q2^{-\ell}$. Decoding takes the ℓ most significant bits. So, each $\mathbf{Z}_q$ element is at a distance up to $\rho_+ = \rho_- = q2^{-\ell-1}$ to a codeword. Elements of t, d, f, e, sk, are sampled in $\{-11, \ldots, +11\}$ with Gaussian-looking distribution.

Example 4 (NewHope). NewHope-CPA-PKE [1] defines $S_{\mathsf{sk}} = S_A = S_B = S_t = S_U = S_V = \mathbf{Z}_q^n$. Elements are considered as polynomials in variable X modulo $X^n + 1$. Bilinear mappings are simply multiplications of polynomials in this structure. Message bits are encoded by multiplication to $L = q/2$ and represented twice in NewHope512-CPA-PKE. Namely, if $Y = \mathsf{encode}(\mathsf{pt})$, the bit pt_i of the message appears at position i and $i + 256$ of Y by $Y_i = Y_{i+256} = \mathsf{pt}_i \frac{q}{2}$. How decoding works is also important. Namely, if $Y \in S_V$, the algorithm decodes $b = (\mathsf{decode}(Y))_i$ to the value b minimizing $|Y_i - b\frac{q}{2}| + |Y_{i+256} - b\frac{q}{2}|$, i.e. the L_1 distance. For this reason, it uses an L_1 norm on pairs of components at position i and $i + 256$ and the L_∞ norm over all i. Namely,

$$\|Y\| = \max_i(|Y_i| + |Y_{i+256}|)$$

Hence, $\rho_- = \frac{q}{2}$ and $\rho_+ = \frac{q}{2}$. For t, d, f, e, sk, sparse elements of $\mathbf{Z}_q$ are sampled by taking the difference of the Hamming weight of two uniformly distributed random bytes. Hence, they are in $\{-8, \ldots, +8\}$. NewHope deviates a bit from our meta-construction in the sense that encryption replaces V by its compression $\bar{V} = \lceil \frac{p}{q} V \rfloor$ and decryption replaces $\bar{V}$ by $V' = \lceil \frac{q}{p} \bar{V} \rfloor$. This adds an error bounded by $\frac{q}{p}$ in δ. In NewHope512-CPA-PKE, the parameters are $n = 512$, $q = 12\,289$, and $p = 8$.

Example 5 (Lepton). Lepton.CPA [26] defines $S_{\mathsf{sk}} = S_A = S_B = S_t = S_U = \mathbf{Z}_2^n$ and $S_V = \mathbf{Z}_2^\ell$ with the Hamming weight norm. All bilinear functions are the $\mathsf{GF}(2^n)$ multiplications represented with the trinomial $X^n + X^m + 1$, except for $S_t \times S_B \to S_V$ which does the multiplication then truncate to ℓ bits. For t, d, f, e, and sk, of weight bounded by k, it is proven that δ has a weight bounded by $4k^2 + k + 2m - 2$. Encoding uses a BCH code and a repetition code to correct at least this amount of error. The Light I version of Lepton.CPA uses $n = 8\,100$, $k = 16$, $m = 9$, and $\ell = 4\,572$. The BCH code is a $[508, 256]$ binary code which can decode up to 30 errors. After BCH-encoding, it uses a repetition code with 9 repetitions. By design, $\|\delta\| \leq 1\,056$.

3 The Noise Learning Problem

In this section, we will introduce a noise learning problem with examples for different metrics. Solving it is the heart of the KR-PCA attack that we describe in the next section. We prove lower bounds for the number of oracle calls to solve this problem.

3.1 Definition

We consider a sampling algorithm $\mathcal{S}$ generating a sparse δ (which we call *noise*), a threshold ρ, and an algorithm $\mathsf{learn}^{\mathsf{BOO}(\cdot)}$ which makes r queries to a *bounded offset oracle* BOO. We will later give a few instances of the learning problem. We define $\mathsf{Adv}(\mathsf{LEARN}^{\rho}_{\mathcal{S}}(\lambda))$ as the probability that the game on Fig. 4 gives 1.

Game $\mathsf{LEARN}^{\rho}_{\mathcal{S}}(\lambda)$:
1: pick coinS
2: $\mathsf{setup}(1^{\lambda}; \mathsf{coinS}) \to \mathsf{pp}$
3: pick the noise δ using the sampling algorithm $\mathcal{S}$
4: $\mathsf{learn}^{\mathsf{BOO}(\cdot)}(\mathsf{pp}) \to \delta'$
5: **return** $1_{\delta=\delta'}$

Oracle $\mathsf{BOO}(x)$:
1: **return** $1_{\|\delta+x\|\leq\rho}$

Fig. 4. The noise learning game with threshold ρ.

In all cases, the idea of the learning algorithm is to start with $x = 0$ and to gradually increase $\|x\|$ (i.e. reduce the sparsity of $\delta + x$) until the decoding is no longer the same. This is a hill-climbing method. Then, we can explore the top of the hill with small modifications of this critical x and analyze the decoding algorithm to deduce the value of δ.

3.2 Lower Bounds for the Noise Learning Problem

Theorem 6. *Given a probability distribution for δ, we assume that an algorithm* learn *has an advantage of 1 in the* LEARN *game, using r queries. We have*

$$E(r) \geq H(\delta)$$

where H is the Shannon entropy.

Proof. Since we do not consider the running time, we assume without loss of generality that learn uses the random coins minimizing $E(r)$. Hence, we consider it as deterministic. We let $C(\delta)$ be the sequence of answers from the oracle BOO when δ is sampled. When running learn alone with oracle answers simulated by

the sequence $b_1, \ldots, b_r$, we let $D(b_1 \cdots b_r) = \delta'$ be the output from learn. We let r_δ be the length of $C(\delta)$. Due to the hypothesis that the advantage is 1, we have $D(C(\delta)) = \delta$ for all δ. Thanks to this property, the Kraft Inequality says that $\sum_\delta 2^{-r_\delta} \leq 1$. We have

$$\begin{aligned} E(r_\delta) - H(\delta) &= \sum_\delta \Pr[\delta] \log_2(2^{r_\delta} \Pr[\delta]) \\ &\geq \frac{1}{\ln 2} \sum_\delta \Pr[\delta] \left(1 - \frac{1}{2^{r_\delta} \Pr[\delta]}\right) \\ &= \frac{1}{\ln 2} \left(1 - \sum_\delta 2^{-r_\delta}\right) \\ &\geq 0 \end{aligned}$$

by using $\ln x \geq 1 - \frac{1}{x}$. □

Theorem 7. *Given a probability distribution for δ, we assume that an algorithm* learn *has an advantage of p in the* LEARN *game, using r queries. We have*

$$E(r|\mathsf{success}) \geq H_\infty(\delta) + \log_2 p$$

where H_∞ is the min-entropy and success *is the event that* LEARN *returns* 1.

Proof. As in Th. 6, we assume without loss of generality that learn is deterministic. Let S be the distribution of δ and we define $\bar{S} = [S|\delta' = \delta]$, i.e. the distribution S conditioned to $\delta' = \delta$. The advantage with δ of distribution $\bar{S}$ is 1. Due to the previous result, we have that $E_{\bar{S}}(r) \geq H(\bar{S})$. We have

$$H(\bar{S}) = -\sum_{\delta=\delta'} \frac{\Pr[\delta]}{p} \log_2 \frac{\Pr[\delta]}{p} \geq \sum_{\delta=\delta'} \frac{\Pr[\delta]}{p} (H_\infty(\delta) + \log_2 p) = H_\infty(\delta) + \log_2 p$$

and

$$E_{\bar{S}}(r) = E_S(r|\delta = \delta') = E_S(r|\mathsf{success})$$

therefore, $E(r|\mathsf{success}) \geq H_\infty(\delta) + \log_2 p$. □

3.3 Example: Learning a Small Integer

We consider the learning problem over $\mathbf{Z}_q$. We assume that ρ and p are defined parameters. If $\rho = \lfloor \frac{q}{2} \rfloor$, the BOO oracle always answers 1 and the problem is unsolvable. Hence, we assume that $\rho < \lfloor \frac{q}{2} \rfloor$. We design a learning algorithm with parameter p by a cut-and-choose algorithm as shown on Fig. 5. We first assume that ρ is known.

Theorem 8. *Assume that $\rho < \lfloor \frac{q}{2} \rfloor$. The algorithm on Fig. 5 succeeds with probability at least p. The number r of oracle calls satisfies*

$$E(r) \leq 1.41 + 2.41 \times (H(\delta) + \log_2 p) + r_1^{\max}$$

Algorithm $\mathsf{learn}(\mathsf{pp})$:

1: define q from pp
2: set $a = -\rho$ ▷ we have $\mathsf{BOO}(-a-\rho) = 1$
3: set $b = \rho$
4: **if** $\rho > \frac{q}{6} - \frac{1}{3}$ **then**
5: $\quad \ell = \lfloor \frac{q}{2} \rfloor - \rho$ ▷ $-\rho \le a \le \delta \le b$
6: $\quad$ **while** $\max_{\delta \in [a,b]} \Pr[\delta] < p \cdot \Pr[\delta \in [a,b]]$ and $b - a \ge \ell$ **do**
7: $\quad\quad b' = a + \ell - 1$
8: $\quad\quad$ **if** $\mathsf{BOO}(-b' - 1 - \rho) = 1$ **then** $a \leftarrow b' + 1$ **else** $b \leftarrow b'$
9: $\quad$ **end while**
10: **end if** ▷ we have $\mathsf{BOO}(-b-1-\rho) = 0$ and $b - a \le \ell - 1$
11: **while** $\max_{\delta \in [a,b]} \Pr[\delta] < p \cdot \Pr[\delta \in [a,b]]$ **do** ▷ we have $a \le \delta \le b$
12: $\quad$ cut $[a,b] = [a,c] \cup [c+1,b]$ which minimizes

$$\max\left(\Pr[\delta \in [a,c]], \Pr[\delta \in [c+1,b]]\right)$$

▷ the median of $(\delta | [a \le \delta \le b])$ is either c or $c+1$
13: $\quad$ **if** $\mathsf{BOO}(-c-1-\rho) = 1$ **then** $a \leftarrow c+1$ **else** $b \leftarrow c$
14: **end while**
15: pick $\delta = \arg\max_{\delta \in [a,b]} \Pr[\delta]$
16: **return** δ

Fig. 5. Learning an integer with probability p.

with $r_1^{\max} = \frac{2\rho}{\lfloor \frac{q}{2} \rfloor - \rho}$ *if* $\rho > \frac{q}{6} - \frac{1}{3}$ *and* $r_1^{\max} = 0$ *otherwise. When* δ *is uniform in* $[-\rho, +\rho]$*, we have*

$$E(r) \le H(\delta) + \log_2 p + 1 + r_1^{\max}$$

For $\rho \approx \frac{q}{4}$, $r_1^{\max}$ is typically 2. If we neglect this overhead, we deduce that the learning algorithm is optimal up to a factor of 2.41 in general. It is further optimal for a uniform distribution.

Proof. Let $\ell = \lfloor \frac{q}{2} \rfloor - \rho$ and $b' = a + \ell - 1$. If $-\rho \le a \le \delta \le b$, we have $\delta - b' - 1 - \rho = \delta - a - \ell - \rho \ge -\ell - \rho = -\lfloor \frac{q}{2} \rfloor$ and $\delta - b' - 1 - \rho = \delta - a - \ell - \rho \le \delta - \ell \le \rho$ thus $\|\delta - b' - 1 - \rho\| \le \rho$ is equivalent to $\delta - b' - 1 - \rho \ge -\rho$. This means that $\mathsf{BOO}(-b' - 1 - \rho) = 1$ is equivalent to $\delta \ge b' + 1$. This shows that the condition $-\rho \le a \le \delta \le b$ is preserved in the loop in Step 6–9.

The loop in Step 6–9 terminates as soon as soon as $\mathsf{BOO}(-b' - 1 - \rho) = 0$. The previous argument shows that the loop terminates when $\delta \le b'$. Hence, the number r_1 of iterations of this loop is bounded by $r_1 \le \frac{2\rho}{\ell}$ which is $r_1^{\max}$.

The purpose of both loops is to find an interval $[a,b]$ containing δ such that $\max_{\delta \in [a,b]} \Pr[\delta] < p \cdot \Pr[\delta \in [a,b]]$. The loop in Step 6–9 either directly finds this interval or finds one containing δ such that $b - a \le \ell - 1$. If $\rho \le \frac{q}{6} - \frac{1}{3}$, this loop is skipped but we already have $b - a = 2\rho \le \ell - 1$.

We can repeat the previous argument: if δ and c belong to $[a,b]$, $b - a \le \ell - 1$, and $b - a \le 2\rho$, then $\delta - c - 1 - \rho \ge \delta - a - \ell - \rho \ge -\ell - \rho = -\lfloor \frac{q}{2} \rfloor$ and

$\delta - c - 1 - \rho \leq b - a - \rho \leq \rho$ thus $\|\delta - c - 1 - \rho\| \leq \rho$ is equivalent to $\delta - c - 1 - \rho \geq -\rho$. This means that $\mathsf{BOO}(-c-1-\rho) = 1$ is equivalent to $\delta \geq c+1$. Hence, we prove by induction that at every step of the loop in Step 11–14, δ belongs to $[a,b]$, $b-a \leq \ell - 1$, and $b - a \leq 2\rho$.

When the algorithm terminates, it returns δ in the interval with maximal likelihood. The loop enforces that $\Pr[\delta | a \leq \delta \leq b] \geq p$ for all δ in the interval. Hence, the probability that the best δ in the interval is correct is at least p.

Let r_2 be the number of iterations. Let a_1, b_i, c_i be the values of a, b, c in the ith iteration. In the ith iteration, we let $p_i = \Pr[a_i \leq \delta \leq b_i]$, $x_i \in \{c_i, c_i + 1\}$ be the median point in the $[a_i, b_i]$ interval, $m_i = \Pr[\delta = x_i]$, and we set $d_i = 1_{x_i \in [a_{i+1}, b_{i+1}]}$. We have $a_0 = -\rho$, $b_0 = \rho$, $p_0 = 1$.

When cutting $[a_i, b_i]$ in two intervals $[a_i, c_i]$ and $[c_i + 1, b_i]$, we let x_i in the most probable of these two intervals. Hence, $p_{i+1} \leq \frac{p_i + d_i m_i}{2}$.

Another property of the x_i is that when $d_i = 1$, it becomes one border of the next interval. Hence, if it is taken in a further iteration as a median point, it can only be the final iteration reducing the interval to the point x_i, which is the r_2th iteration. We deduce that the sequence of x_i is non-repeating until the r_2th iteration. One consequence is that $m_i + m_{i+1} + \cdots + m_{r_2 - 1} \leq p_i$ for any $i < r_2$.

Let $S_i = d_i m_i + d_{i+1} m_{i+1} + \cdots + d_{r_2 - 1} m_{r_2 - 1}$. We have $S_i \leq p_i$. We also have

$$p_{i+1} \leq \frac{p_i + S_i - S_{i+1}}{2}$$

for all $i < r_2$. By induction, we prove

$$p_i \leq 2^{-i} + 2^{-i-1} S_0 + \sum_{j=0}^{i-1} 2^{-i+j-1} S_j - \frac{1}{2} S_i$$

for all $i < r_2$. We deduce

$$2^i p_i \leq \frac{3}{2} + \frac{1}{2} \sum_{j=0}^{i-1} 2^j p_j$$

and hence $2^i p_i \leq \left(\frac{3}{2}\right)^{i+1}$ by induction. Therefore, $p_i \leq \frac{3}{2} \left(\frac{3}{4}\right)^i$ for $i < r_2$.

For $i \geq \frac{\log\left(\frac{2 \Pr[\delta]}{3p}\right)}{\log \frac{3}{4}}$, we have $\Pr[\delta] \geq \frac{3}{2} \left(\frac{3}{4}\right)^i p \geq p_i p$. So, any δ with probability of occurrence $\Pr[\delta]$ makes $r_2 \leq \frac{\log\left(\frac{2 \Pr[\delta]}{3p}\right)}{\log \frac{3}{4}}$ iterations. Hence,

$$\begin{aligned} E(r_2) &\leq \frac{\log \frac{2}{3}}{\log \frac{3}{4}} + \frac{1}{\log_2 \frac{3}{4}} E\left(\log_2 \frac{\Pr[\delta]}{p}\right) \\ &= \frac{\log \frac{2}{3}}{\log \frac{3}{4}} - \frac{1}{\log_2 \frac{3}{4}} (H(\delta) + \log_2 p) \\ &\leq 1.41 + 2.41 (H(\delta) + \log_2 p) \end{aligned}$$

We treat the uniform case similarly. We let $\ell_i = b_i - a_i + 1$. We have $\ell_0 = 2\rho+1$. We have $\ell_{i+1} \leq \frac{\ell_i+1}{2}$. Hence, $\ell_i \leq 2^{1-i}\rho + 1$. For $i \geq \log_2(2\rho) + \log_2 p$, we have $\ell_i p \leq 1 + p$. The ℓ_i sequence is strictly decreasing until $i = r_2$. Hence, if $i < r_2$, then $\ell_{i+1} \leq \ell_i - 1$ so $\ell_{i+1} p \leq 1$ which implies $i + 1 = r_2$. Therefore $r_2 \leq i + 1$. Hence,

$$E(r_2) \leq \log_2(2\rho) + \log_2 p + 1 \leq H(\delta) + \log_2 p + 1$$

The number of oracle calls is $r = r_1 + r_2$. □

In $\mathbf{Z}_q^n$ with the L_∞ norm, we can run this algorithm for each coordinate and use $r \leq 1.41 + 2.41 \times n \log_2(2\rho + 1)$ queries to learn δ (using that the entropy of a single component is bounded by $\log_2(2\rho + 1)$).

Learning an integer with unknown threshold. The previous algorithm does not work if the threshold ρ is unknown. However, we could learn the offset x such that $\delta + x$ reaches this unknown threshold, either $+\rho$ or $-\rho$, by using an upper bound ρ' on this threshold. We write the algorithm on Fig. 6 for the uniform distribution and $\rho' \leq \frac{q}{6} - \frac{1}{3}$, for simplicity. (For $\rho' > \frac{q}{6} - \frac{1}{3}$, we apply the same strategy as previously to first find a small interval and run the algorithm on this interval.) The algorithm first learns $\rho - \delta$ on Step 8. Similarly, it learns $-\rho - \delta$ on Step 15. Finally, it deduces both δ and ρ.

Algorithm learn(pp):

1: define q from pp
2: set $a = 0$ ▷ we have $\mathsf{BOO}(a) = 1$
3: set $b = 2\rho' + 1$ ▷ we have $\mathsf{BOO}(b) = 0$
4: **while** $b > a + 1$ **do** ▷ we have $a \leq \rho - \delta < b$
5: set $c = \left\lfloor \frac{a+b}{2} \right\rceil$
6: **if** $\mathsf{BOO}(c) = 1$ **then** $a \leftarrow c$ **else** $b \leftarrow c$
7: **end while**
8: $x \leftarrow a$ ▷ $x = \rho - \delta$
9: set $a = -2\rho' - 1$ ▷ we have $\mathsf{BOO}(a) = 0$
10: set $b = 0$ ▷ we have $\mathsf{BOO}(b) = 1$
11: **while** $b > a + 1$ **do** ▷ we have $a < -\rho - \delta \leq b$
12: set $c = \left\lfloor \frac{a+b}{2} \right\rceil$
13: **if** $\mathsf{BOO}(c) = 0$ **then** $a \leftarrow c$ **else** $b \leftarrow c$
14: **end while**
15: $y \leftarrow b$ ▷ $y = -\rho - \delta$
16: **return** $-\frac{x+y}{2}$ ▷ we deduce $\rho = \frac{x-y}{2}$ as well

Fig. 6. Learning an integer with a bound $\rho' \leq \frac{q}{6} - \frac{1}{3}$ on the unknown threshold ρ.

3.4 Example: Learning a Vector which is L_1-Small

When δ is a vector with components in $\mathbf{Z}_q$ and the norm is the L_1 norm, we can learn δ easily when $\rho \leq \frac{q}{6} - \frac{1}{3}$. The idea is first to learn δ_1 and $\rho - |\delta_2| - \cdots - |\delta_n|$. Then we can iterate and learn every δ_i. We can easily see that the number of oracle calls is roughly $(2n-1)\log_2 \rho$.

This is more complicated when $\rho > \frac{q}{6} - \frac{1}{3}$. Below, we study the $\rho = \frac{q}{2}$ case in dimension $n = 2$ which is significant for NewHope [1].

Lemma 9. *We consider the* LEARN *game over* $\mathbf{Z}_q^2$ *with the* L_1 *norm and* $\rho = \frac{q}{2}$. *Given* $c \in [-\rho, \rho]$, *we have* $\mathsf{BOO}(c-\rho, 0) = 1 \iff |\delta_1 + c| \geq |\delta_2|$.

Similarly, $\mathsf{BOO}(0, c-\rho) = 1 \iff |\delta_2 + c| \geq |\delta_1|$.

Proof. If $\delta_1 + c - \rho \geq -\frac{q}{2}$, we observe that $\delta_1 + c - \rho \leq \frac{q}{2}$ since $c \leq \rho$, so $\|\delta + (c-\rho, 0)\| = |\delta_1 + c - \rho| + |\delta_2|$. Hence, $\mathsf{BOO}(c-\rho, 0) = 1$ is equivalent to $|\delta_2| - \rho \leq \delta_1 + c - \rho \leq -|\delta_2| + \rho$. The second inequality is always true because $\|\delta\| \leq \rho$ and $c \leq \rho$. $\mathsf{BOO}(c-\rho, 0) = 1$ is therefore equivalent to $|\delta_2| - \rho \leq \delta_1 + c - \rho$, thus to $|\delta_2| \leq \delta_1 + c$. Furthermore, $\delta_1 + c - \rho \geq -\frac{q}{2}$ implies that $\delta_1 + c \geq 0$ (since $\rho = \frac{q}{2}$). Hence, $\mathsf{BOO}(c-\rho, 0) = 1$ is equivalent to $|\delta_2| \leq |\delta_1 + c|$.

If $\delta_1 + c - \rho \leq -\frac{q}{2}$, we observe that $\delta_1 + c - \rho \geq -\frac{3q}{2}$, so $\|\delta + (c-\rho, 0)\| = |\delta_1 + c - \rho + q| + |\delta_2|$. Hence, $\mathsf{BOO}(c-\rho, 0) = 1$ is equivalent to $|\delta_2| - \rho \leq \delta_1 + c - \rho + q \leq -|\delta_2| + \rho$. The first inequality is always granted, hence $\mathsf{BOO}(c-\rho, 0) = 1$ is equivalent to $\delta_1 + c \leq -|\delta_2|$. Furthermore, $\delta_1 + c - \rho \leq -\frac{q}{2}$ implies that $\delta_1 + c \leq 0$, therefore $\mathsf{BOO}(c-\rho, 0) = 1$ is equivalent to $-|\delta_1 + c| \leq -|\delta_2|$ as well.

The equivalence between $\mathsf{BOO}(0, c-\rho) = 1$ and $|\delta_2 + c| \geq |\delta_1|$ is obtained by swapping the two coordinates of the BOO oracle and of δ. □

Theorem 10. *We consider the* LEARN *game over* $\mathbf{Z}_q^2$ *with the* L_1 *norm and* $\rho = \frac{q}{2}$. *There exists a learning algorithm making up to* $3\log_2 \rho + 2$ *oracle calls.*

Proof. Using Lemma 9 with $c = 0$, we can first learn if $|\delta_1| \leq |\delta_2|$ by making an oracle call $\mathsf{BOO}(0, -\rho)$. We assume without loss of generality that $|\delta_1| < |\delta_2|$. (If not, we flip the pair of entries in the BOO oracle and learn a flipped δ; the $|\delta_1| = |\delta_2|$ case will solve by itself.) Hence, the inequality $|\delta_1 + c| \geq |\delta_2|$ is not satisfied for $c = 0$ but it may be for $c = \rho$ or $c = -\rho$, or even both. By two cut-and-choose algorithms, we find a threshold c making it an equality in both intervals $[-\rho, 0]$ and $[0, \rho]$, if there exists one. This requires up to $2\log_2 \rho$ oracle calls. If there are two, the smallest one has the same sign as δ_1. If there is one, it has the same sign as δ_1. The $|\delta_1| = |\delta_2|$ case gives $c = 0$ and any oracle query with $c \neq 0$ gives the sign of δ_1. In all cases, we learn the sign of δ_1 and the smallest value c such that $|\delta_1 + c| = |\delta_2|$. We can also learn if δ_1 is null and isolate this case as we already learned $\delta_1 = 0$ and $|\delta_2| = |c|$. For $\delta_1 \neq 0$, we continue with Lemma 9 again. We have $\mathsf{BOO}(0, -|c| - 1 - \rho) = 1 \iff |\delta_2 - |c| - 1| \geq |\delta_1|$, which is equivalent to $\delta_2 < 0$. Hence, we learn the sign of δ_2 too with one oracle call. So far, with $2\log_2 \rho + 2$ oracle calls, we found the sign of δ_1 and δ_2, and a smallest c such that $|\delta_1 + c| = |\delta_2| = \frac{1}{2}\|\delta + (c, 0)\|$.

Let $\varepsilon_i \in \{-1, +1\}$ such that $\varepsilon_i \delta_i \geq 0$ for $i = 1, 2$. Let $d \in [0, \rho]$. We consider oracle calls of form $\mathsf{BOO}(c + \varepsilon_1 \frac{d}{2}, \varepsilon_2 \frac{d}{2})$. We have $|\delta_1 + c| = \frac{\|\delta + (c,0)\|}{2} \leq \frac{\rho}{2}$ so $|\delta_1 + c + \varepsilon_1 \frac{d}{2}| \leq \rho$ and similarly, $|\delta_2 + \varepsilon_2 \frac{d}{2}| \leq \rho$. Hence,

$$\left\| \delta + \left(c + \varepsilon_1 \frac{d}{2}, \varepsilon_2 \frac{d}{2} \right) \right\| = \left| \delta_1 + c + \varepsilon_1 \frac{d}{2} \right| + \left| \delta_2 + \varepsilon_2 \frac{d}{2} \right| = \|\delta\| + d$$

By cut-and-choose, we learn $d \in [0, \rho]$ such that $\|\delta\| + d = \rho$. We deduce $\|\delta\|$ then δ completely. □

3.5 Example: Learning a String of Small Hamming Weight

As another example, consider that δ lives in $\{0, 1\}^n$ and that $\|x\|$ is the Hamming weight of x. We consider $\oplus$, the bitwise XOR, as an addition. We assume $\rho \leq \frac{n}{2}$.

Algorithm $\mathsf{learn}(\mathsf{pp})$:
1: define n from pp
2: set $x = (0, 0, \ldots, 0)$ ▷ we have $\mathsf{BOO}(x) = 1$
3: set $y = (1, 1, \ldots, 1)$ ▷ we have $\mathsf{BOO}(x \oplus y) = 0$
4: **while** $\|y\| > 1$ **do** ▷ during the loop, $\mathsf{BOO}(x) = 1$ and $\mathsf{BOO}(x \oplus y) = 0$
5: split at random $y = u \oplus v$ with $u \wedge v = 0$, $\|u\| = \left\lfloor \frac{\|y\|}{2} \right\rfloor$, and $\|v\| = \left\lceil \frac{\|y\|}{2} \right\rceil$,
6: **if** $\mathsf{BOO}(x \oplus u) = 1$ **then**
7: $x \leftarrow x \oplus u$
8: $y \leftarrow v$
9: **else**
10: $y \leftarrow u$
11: **end if**
12: **end while**
13: $z \leftarrow x$ ▷ we know that $\|\delta \oplus x\| = \rho$
14: set i_{done} such that $y_{i_{\mathsf{done}}} = 1$
15: **for** $i = 1$ to n except $i \neq i_{\mathsf{done}}$ **do**
16: set y to x with the i-th bit flipped
17: **if** $\mathsf{BOO}(y) = 1$ **then** in z, flip the i-th bit
18: **end for**
19: **return** z

Fig. 7. Learning a bitstring.

We can have a learning algorithm by a cut-and-choose algorithm.

Theorem 11. *The algorithm on Fig. 7 succeeds with probability* 1. *The number of iterations r satisfies*

$$r \leq n + \log_2 n$$

The analysis is similar as before.

When δ is uniformly distributed among the strings of Hamming weight at most ρ, the entropy is

$$H(\delta) = \log_2 \left(\sum_{w=0}^{\rho} \binom{n}{w} \right) \leq nH\left(\frac{\rho}{n}\right)$$

where H is the binary entropy function.

For $\rho = \frac{n}{2}$, we obtain $H(\delta) \leq n$. The number of queries is asymptotically equivalent to n and it is nearly optimal.

For $\rho = \frac{n}{4}$, we obtain $H(\delta) \leq 0.82n$, meaning that the algorithm is not optimal.

4 Key-Recovery with a Plaintext Checking Oracle

With the meta-PKC construction, there is a fixed sk and the adversary can play with an oracle $\mathsf{PCO}(U, V, \mathsf{pt})$ checking if the noise is sparse, i.e. if $\mathsf{encode}(\mathsf{pt})$ is the closest codeword to $V - U \times \mathsf{sk}$. One strategy consists of learning the noise $\delta = t \times d + f - e \times \mathsf{sk}$ by playing with PCO. Actually, by defining $\mathcal{O}(x) = \mathsf{PCO}(U, V + x, \mathsf{pt})$, we simulate a bounded offset oracle to run the LEARN game.

Learning δ gives a linear equation with unknown d and sk. We can eliminate d using the known relation $B = A \times \mathsf{sk} + d$. With a few equations (depending on the selected algebra), we deduce sk.

More precisely, if we can learn δ sampled by $\mathcal{S}$ which generates δ like $\delta = t \times d + f - e \times \mathsf{sk}$, we assume that for some k (typically, $k = 1$ if all sets are equal), there exists an algorithm such that given k equations of form $(t_i \times A + e_i) \times \mathsf{sk} = t_i \times B + f_i - \delta_i$ with sk unknown, it solves sk. We use the algorithm on Fig. 8.

We obtain the following result.

Theorem 12. *Let t, d, f, e, sk follow the random distributions of the cryptosystem. We define*

$$\delta = t \times d + f - e \times \mathsf{sk}$$

*With probability 1, the algorithm of Fig. 8 gives at every iteration of the **for** loop, one linear equation over S_V with unknown $\mathsf{sk} \in S_{\mathsf{sk}}$, using the same number of oracle calls as learn with the distribution of δ.*

In the case that $S_{\mathsf{sk}} = \mathbf{Z}_q^{n_{\mathsf{sk}}}$, $S_A = \mathbf{Z}_q^{n_A}$, $S_B = \mathbf{Z}_q^{n_B}$, $S_t = \mathbf{Z}_q^{n_t}$, $S_U = \mathbf{Z}_q^{n_U}$, $S_V = \mathbf{Z}_q^{n_V}$, when the considered norm is the L_∞ norm, this means that we obtain n_V linear equations over $\mathbf{Z}_q$ with n_{sk} unknowns. We use $H(\delta)$ oracle calls and we can approximate $H(\delta) \approx n_V \log_2(2\rho_+)$. (This can only over-estimate the entropy.)

One difficulty is to know what happens when the distance of $V - U \times \mathsf{sk}$ to the code is between ρ_- and ρ_+. If $\rho_- = \rho_+$, there is no problem. If decryption aborts when the number of errors exceeds ρ_-, then ρ_- becomes de facto the threshold to be used in the LEARN game. Otherwise, the learn algorithm must be refined.

Algorithm $\mathcal{A}^{\mathsf{PCO}(\cdot)}(\mathsf{pp}, \mathsf{pk})$:

1: parse $\mathsf{pk} = (A, B)$
2: **for** $i = 1$ to k **do**
3: pick pt_i at random
4: run $\mathsf{enc}(\mathsf{pp}, \mathsf{pk}, \mathsf{pt}_i) \to \mathsf{ct}_i$ ▷ this defines t_i, e_i, f_i
5: parse $\mathsf{ct}_i = (U_i, V_i)$
6: run $\mathsf{learn}^{\mathcal{O}(\cdot)}(\mathsf{pp}) \to \delta_i$ ▷ deduce $(t_i \times A + e_i) \times \mathsf{sk} = t_i \times B + f_i - \delta_i$
7: **end for**
8: solve $\begin{pmatrix} t_1 \times A + e_1 \\ \vdots \\ t_k \times A + e_k \end{pmatrix} (\mathsf{sk}) = \begin{pmatrix} t_1 \times B + f_1 - \delta_1 \\ \vdots \\ t_k \times B + f_k - \delta_k \end{pmatrix}$
9: **return** sk

Oracle $\mathcal{O}(x)$:

10: **return** $\mathsf{PCO}(U_i, V_i + x, \mathsf{pt}_i)$

Fig. 8. KR-PCA attack based on learning.

Example 13 (Frodo continued). The Frodo-640 parameters are $n_{\mathsf{sk}} = n_B = n\bar{n} = 640 \times 8$ and $n_V = \bar{m}\bar{n} = 8 \times 8$. Since $\rho_+ = \rho_- = q2^{-\ell-1} = 2^{12}$, we need about 2^{10} oracle calls to recover 2^6 equations in 2^{12} unknowns. (By iterating, we have a full key recovery using 2^{16} oracle calls.)

Example 14 (NewHope continued). NewHope512 defines $n_{\mathsf{sk}} = n_B = n_V = n = 512$, $\rho_- = \frac{q}{4}$, $\rho_+ = \frac{q}{2}$, and $p = 8$. Decoding in NewHope is based on the cumulative distance. We should rather apply the algorithm in Th. 10 for the L_1 norm in dimension two to learn each pair of components of δ. However, the use of compression of V into $\bar{V}$ in NewHope makes the attack learning an approximation of δ instead of δ completely. Essentially, we learn $\lceil \frac{p}{q}\delta \rfloor$ (i.e., the $\log_2 p$ most significant bits of each of the n_V components of δ). However, we already know that δ is sparse, so we learn zero bits only. The attack does not work but we can adopt another strategy of well selecting $(U, \bar{V})$ to progressively learn the bits of sk. This was done by Bauer et al. [5].

Example 15 (Lepton continued). With Lepton.CPA Light I, δ is a string of length $\ell = 4\,572$ of Hamming weight bounded by $1\,056$. Encoding consists of $d = 9$ repetitions of a BCH code which can correct up to 30 errors. One problem is that decoding fails if decoding the repetition code results in more than $t = 30$ errors in BCH decoding. Nevertheless, we can adapt the attack based on the learn algorithm of Fig. 7. In learn, instead of considering a string of ℓ bits, we consider a string of $\frac{\ell}{d}$ packets of d bits in which the packet 1 represents the packet of d bits set to 1, and flipping a packet means xoring it to the packet 1. Thus, we learn which packets of δ have an error using up to $\frac{\ell}{d} + \log_2 \frac{\ell}{d}$ queries. Then, for each packet, we modify δ to have exactly $t - 1$ incorrect other packets

and we apply the learn algorithm of Fig. 7 at the bit level. For each packet, we need up to $d + \log_2 d$ oracle calls. In total, the number of oracle calls is bounded by $\ell + \frac{\ell}{d}\log_2 d + \frac{\ell}{d} + \log_2 \frac{\ell}{d} \approx 2^{13}$. This gives 4 572 equations in 8 100 unknowns. So we use $k = 2$ and recover the entire sk using 2^{14} oracle calls.

5 Quantum Key-Recovery with a Decryption Oracle

5.1 GKZ-Based Attack

We build a KR-CCA attack which is inspired by the quantum LWE solving algorithm from Grilo, Kerenidis, and Zijlstra (GKZ) [14]. This algorithm works with a quantum superposition of LWE entries. In this attack, we consider an adversary with quantum access to a decryption oracle. More precisely, we assume that the oracle makes the following mapping:

$$|\mathsf{ct}\ \ x\ \ Z\rangle \mapsto |\mathsf{ct}\ \ (x \oplus \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}))\ \ Z\rangle$$

Instead of calling this oracle on a chosen ct, we will call it on a superposition of ct states.

We denote $\omega_q = e^{\frac{2i\pi}{q}}$, a qth primitive root of unity.

We consider our meta PKC construction in which all groups are powers of $\mathbf{Z}_q$ for simplicity: $S_{\mathsf{sk}} = \mathbf{Z}_q^{n_{\mathsf{sk}}}$, $S_A = \mathbf{Z}_q^{n_A}$, $S_B = \mathbf{Z}_q^{n_B}$, $S_t = \mathbf{Z}_q^{n_t}$, $S_U = \mathbf{Z}_q^{n_U}$, $S_V = \mathbf{Z}_q^{n_V}$.

We let $I \subseteq \{1, \ldots, n_V\}$ be a set of indices i.

We split the quantum state into several registers:

- one register $U \in S_U$;
- one register $V \in S_V$;
- a plaintext in $\mathcal{M}$;
- one register $Z \in \mathbf{Z}_q^I$.

We assume that we have an operator L mapping

$$|U\ \ V\ \ \mathsf{pt}\ \ Z\rangle \mapsto |U\ \ V\ \ \mathsf{pt}\ \ (Z \oplus (V - \mathsf{encode}(\mathsf{pt}))_I)\rangle$$

where V_I denotes the restriction of the vector V on indices in I. This means that we compute the ith coordinates of $V - \mathsf{encode}(\mathsf{pt})$ and XOR it to the working register Z.

We run the algorithm on Fig. 9. Steps 6–7 of our attack are equivalent to the GKZ algorithm [14]. As usual quantum algorithms, the algorithm is deterministic until we perform a measurement (in Step 7). What follows the measurement in Step 7 is done by a classical computer.

We define $W_U = V - U \times \mathsf{sk}$, $\mathsf{pt}_U = \mathsf{decode}(W_U)$, $Z_U = (V - \mathsf{encode}(\mathsf{pt}_U))_I$, $\delta_U = W_U - \mathsf{encode}(\mathsf{pt}_U)$. Due to the property of the encoding/decoding algorithms,[2] we have $\|\delta_U\| \leq \rho_+$. We define

$$\psi_U = (\delta_U)_I = (W_U - \mathsf{encode}(\mathsf{decode}(W_U)))_I \tag{2}$$

[2] We recall that we assume that decoding is defined over the entire S_V space.

Input: $I \subseteq \{1, \ldots, n_V\}$ and $V \in S_V$
Decryption oracle: $|U\ \ V\ \ x\ \ Z\rangle \mapsto |U\ \ V\ \ x \oplus \mathsf{Dec}(\mathsf{sk}, (U, V))\ \ Z\rangle$

1: set the quantum state to $|0\ \ V\ \ 0\ \ 0\rangle$
2: make a quantum Fourier transform on the first register and obtain $q^{-\frac{n_U}{2}} \sum_U |U\ \ V\ \ 0\ \ 0\rangle$
3: make a decryption oracle call and obtain $q^{-\frac{n_U}{2}} \sum_U |U\ \ V\ \ \mathsf{pt}_U\ \ 0\rangle$
4: apply the L operator and obtain $q^{-\frac{n_U}{2}} \sum_U |U\ \ V\ \ \mathsf{pt}_U\ \ Z_U\rangle$
5: make a decryption oracle call again and obtain $q^{-\frac{n_U}{2}} \sum_U |U\ \ V\ \ 0\ \ Z_U\rangle$
6: make a quantum Fourier transform on the U and Z registers and obtain

$$q^{-n_U - \frac{\#I}{2}} \sum_{U,\alpha,\beta} \omega_q^{(\alpha \cdot U) + (\beta \cdot Z_U)} |\alpha\ \ V\ \ 0\ \ \beta\rangle$$

7: measure the two active registers to get (α, β)
8: solve the linear system $\alpha_j + (\beta \cdot (e_j \times \mathsf{sk})_I) = 0$, $j = 1, \ldots, n_U$, where e_j is the U vector with coordinates set to 0 except the jth bit which is set to 1
9: **if** not solvable **then** abort
10: set s to the solution
11: **if** s not sparse **then** abort
12: **return** s

Fig. 9. GKZ-based key recovery with quantum access to a decryption oracle.

Hence, $Z_U = (U \times \mathsf{sk})_I + \psi_U$ with ψ_U small. Hence, the state after Step 4 resembles a quantum superposition of LWE entries, but with a spurious register pt. This spurious register has a dramatic impact on the probability of later measurements, as it will be shown below. It was noticed by Ambainis, Magnin, Roetteler, and Roland [3] that getting rid of it is hard in general. Fortunately, we can call the decryption oracle again to clear pt completely. Step 5 is doing it. This is why we need a double query to the decryption oracle.

We call a pair $(\alpha, \beta) \in S_U \times \mathbf{Z}_q^I$ *good* if $\beta \neq 0$ and α satisfies the property $\alpha_j + (\beta \cdot (e_j \times \mathsf{sk})_I) = 0$ for $j = 1, \ldots, n_U$. For all $\beta \neq 0$, we have a unique α such that (α, β) is good. When this property is satisfied, since $U = \sum_j U_j e_j$, we have $(U \times \mathsf{sk})_I = \sum_j U_j (e_j \times \mathsf{sk})_I$, thus

$$\begin{aligned}
(\alpha \cdot U) + (\beta \cdot Z_U) &= (\alpha \cdot U) + (\beta \cdot (U \times \mathsf{sk})_I) + (\beta \cdot \psi_U) \\
&= (\beta \cdot \psi_U) + \sum_j U_j \left((\alpha \cdot e_j) + (\beta \cdot (e_j \times \mathsf{sk})_I) \right) \\
&= (\beta \cdot \psi_U)
\end{aligned}$$

We compute the probability p_g to measure a good (α, β) pair. We have

$$\begin{aligned} p_g &= \sum_{(\alpha,\beta)\ \mathsf{good}} \left| q^{-n_U - \frac{\#I}{2}} \sum_U \omega_q^{(\alpha\cdot U)+(\beta\cdot Z_U)} \right|^2 \\ &= \sum_\beta \frac{1}{q^{2n_U+\#I}} \left| \sum_U \omega_q^{\beta\cdot\psi_U} \right|^2 - q^{-\#I} \end{aligned}$$

where the $q^{-\#I}$ term cancels the $\beta = 0$ term in the sum. By using $|z|^2 = z \times \overline{z}$ over complex numbers, we have

$$\begin{aligned} p_g &= \sum_\beta \frac{1}{q^{2n_U+\#I}} \left(\sum_U \omega_q^{\beta\cdot\psi_U} \right) \left(\sum_{U'} \omega_q^{-\beta\cdot\psi_{U'}} \right) - q^{-\#I} \\ &= \sum_\beta \frac{1}{q^{2n_U+\#I}} \sum_{U,U'} \omega_q^{\beta\cdot(\psi_U-\psi_{U'})} - q^{-\#I} \\ &= \frac{1}{q^{2n_U}} \sum_{U,U'} \frac{1}{q^{\#I}} \sum_\beta \omega_q^{\beta\cdot(\psi_U-\psi_{U'})} - q^{-\#I} \\ &= \frac{1}{q^{2n_U}} \sum_{U,U'} 1_{\psi_U=\psi_{U'}} - q^{-\#I} \\ &= \Pr[\psi_U = \psi_{U'}] - q^{-\#I} \\ &= 2^{-H_2(\psi_U)} - q^{-\#I} \end{aligned}$$

where H_2 is the collision entropy (or Rényi of degree 2).

Without the second decryption call, with the same method, we would have obtained $p_g = 2^{-H_2(\psi_U,\mathsf{pt}_U)} - q^{-\#I} 2^{-H_2(\mathsf{pt}_U)}$ which is too small. This shows the importance of this second call, just to clear one register, although it works against intuition when we are used to classical computing.

Our analysis is based on the decode function being defined on the entire domain S_V. However, instances of our construction may rely on a partially defined decode algorithm. In that case, we may have $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \perp$. By convention, we set $\mathsf{encode}(\perp) = 0$. The same analysis gives

$$\begin{aligned} p_g &= \Pr[\psi_U = \psi_{U'}] - q^{-\#I} \\ &\approx \Pr[\mathsf{pt}_U \neq \perp]^2 \left(\Pr[\psi_U = \psi_{U'} | \mathsf{pt}_U \neq \perp, \mathsf{pt}_{U'} \neq \perp] - q^{-\#I} \right) \end{aligned}$$

This may be too small if $\Pr[\mathsf{pt}_U \neq \perp]$ is small. This is the case with Lepton.

One crucial thing is that the distribution of ψ_U comes from (2), where U is uniform, sk comes from the key generation algorithm, and V is fixed. In particular, ψ_U does not follow the normal distribution defined by (1) from the encryption/decryption process which would have a lower $H_2(\psi_U)$. If we had a way to sample ψ from (1) without any spurious register, we would have a better attack.

What we obtain is the following result:

Theorem 16. *Let $I \subseteq \{1, \ldots, n_V\}$ be a set of indices, let $V \in S_V$ be arbitrarily fixed, let $\mathsf{sk} \in S_{\mathsf{sk}}$ following the random distribution of the cryptosystem, and let $U \in S_U$ be uniformly distributed. We define*

$$\psi_U = (V - U \times \mathsf{sk} - \mathsf{encode}(\mathsf{decode}(V - U \times \mathsf{sk})))_I$$

The algorithm of Fig. 9 gives a pair (α, β) at Step 7 such that $\beta \neq 0$ and

$$\forall j \in \{1, \ldots, n_U\} \qquad \alpha_j + (\beta \cdot (e_j \times \mathsf{sk})_I) = 0$$

with probability $2^{-H_2(\psi_U)} - q^{-\#I}$ and using 2 oracle calls. This is a set of n_U linear equations with n_{sk} unknowns over $\mathbf{Z}_q$.

For $\#I = 1$ and ψ_U uniform in $[-\rho_+, +\rho_+]$, we can assume that $H_2(\psi_U) = \log_2(2\rho_+ + 1)$. We obtain that the success probability is roughly $\frac{1}{2\rho_+} - \frac{1}{q}$. Clearly, it requires $\rho_+ < \frac{q}{2}$.

Note that when this fails (with probability $1 - p_g$) and $n_{\mathsf{sk}} \leq n_U$, we can easily filter out those cases because we can eliminate the cases when the equations have no solution or when the solution is not sparse. Hence, either we recover part of sk, or we abort. Therefore, we can iterate this attack to recover (at least some part of) sk with a better probability.

For $n_{\mathsf{sk}} = n_U$, we iterate $1/p_g$ times on average until one sparse equation is found.

For $n_{\mathsf{sk}} > n_U$, we should, in general, treat the problem on a case-by-case basis by studying the structure of the equations we obtain but there are some general methods we can apply when n_{sk}/n_U is small. We could indeed iterate $\frac{n_{\mathsf{sk}}}{p_g n_U}$ times to be sure to get enough sets of n_U equations to recover the n_{sk} unknowns. To recover them, we can try all the $\left(\frac{n_{\mathsf{sk}}}{p_g n_U}\right)^{\frac{n_{\mathsf{sk}}}{n_U}}$ combinations of $\frac{n_{\mathsf{sk}}}{n_U}$ sets. Each combination gives n_{sk} equations. We can solve each combination until one sparse sk is found.

However, for each j, the equation $\alpha_j + (\beta \cdot (e_j \times \mathsf{sk})_I) = 0$ typically depends on a fixed subset of coordinates of sk and we should better apply those methods for each of these subset separately.

We assume $n_{\mathsf{sk}} \leq n_U$. If there is a single coordinate i of $U \times \mathsf{sk}$ which depends on all coordinates of sk, by using $I = \{i\}$, we recover the entire sk with probability p or abort. Hence, iterating p_g^{-1} times fully recover sk with $2p_g^{-1}$ decryption calls.

It is not always possible to find a coordinate i which depends on the entire sk. For instance, in the case of Frodo, $U \times \mathsf{sk}$ is a matrix multiplication so each coordinate i depends on one column of sk only.

Interestingly, $H_2(\psi_U) \leq H(\psi_U)$ so the number of queries when iterating is lower bounded by $2^{1+H(\psi_U)}$. This is a big difference with the previous classical attack, which recovers one $\mathbf{Z}_q$ element within only $H(\psi_U)$ queries. However, it is hard to compare an attack finding a piece of the key using two oracle calls and with probability p to an attack finding linear equations using many oracle calls and succeeding with probability 1. For applications where the number of key reuse is strictly limited, the former is more devastating.

Example 17 (Frodo continued). The Frodo-640 parameters are $n_{\mathsf{sk}} = n_U$ but we need to recover each of the $\bar{n} = 8$ columns separately if we take $\#I = 1$. We approximate $2^{-H_2(\psi_U)} \approx \frac{1}{2\rho_+}$. Since $\rho_+ = q2^{-\ell-1} = 2^{12}$ and $q = 2^{15}$, by using two oracle calls, we recover one column of 640 values with probability 2^{-13}. (By iterating, we need 2^{17} oracle calls to fully recover sk.) We can recover two columns at the same time with $\#I = 2$ in two oracle calls, but with probability 2^{-26}.

Example 18 (NewHope continued). With NewHope512, ψ_U on a single component has no information (this is due to $\rho_+ = \frac{q}{2}$ which does not characterize ψ_U compared to any other $\mathbf{Z}_q$ element). Indeed, we obtain $p = 0$ with $\#I = 1$. We can use $\#I = 2$ with $I = \{i, i + 256\}$. This uses two positions encoding the same bit. We can see that ψ_U encodes two values which are either both smaller than $\frac{q}{4}$ (with probability $\frac{1}{2}$) or with exactly one being smaller than $\frac{q}{4}$. Hence, two ψ_U pairs collide with probability $\frac{1}{4}\left(\frac{2}{q}\right)^2 + \frac{1}{8}\left(\frac{2}{q}\right)^2 = \frac{3}{2q^2}$. It gives $p_g \approx \frac{1}{2q^2}$. This means that with two oracle calls we recover the full secret with probability 2^{-28}. (By iterating, we need $4q^2$ decryption calls, i.e. 2^{29}.) Compressing V in NewHope does not modify our attack as V is constant.

Example 19 (Lepton continued). Lepton.CPA Light I considers $q = 2$. With $\#I = 1$, we obtain $p = 0$. The encoding function in Lepton is obtained by first applying a BCH encoding, then using a repetition code. We can focus on the repetition code and take two repeating bits in the set I with $\#I = 2$. The distribution of ψ_U from (2) should be of form $\Pr[\psi_U = 00] = \frac{1}{2}$, $\Pr[\psi_U = 01] = \Pr[\psi_U = 10] = \frac{1}{4}$. Hence, $\Pr[\psi_U = \psi_{U'}] = \frac{3}{8}$ and we obtain $p = \frac{1}{8}$. Hence, we could recover sk with probability $\frac{1}{8}$.[3] Unfortunately, decode is partially defined, due to the BCH code, and we have $\Pr[\mathsf{pt}_U \neq \perp] \approx 2^{-93}$ which is too low. That is why the attack does not work for Lepton.

5.2 AJOP-Based Attack

We let a and b be two integers. We partition $[a, a + q - 1]$ into c intervals $I_k = [a+kb, a+kb+b-1]$ for $k = 0, \ldots, c-2$ and $I_{c-1} = [a+(c-1)b, a+q-1]$, with $b = \lceil \frac{q}{c} \rceil$. We define $\mathsf{RF}(x) = k$ such that $x \in I_k$ modulo q. Hence, we can consider RF as a function from $\mathbf{Z}_q$ to $\mathbf{Z}_c$.

Lemma 20. *Given $f \in \mathbf{Z}_q^n$ such that $\{u \cdot f; u \in \mathbf{Z}_q^n\} = \mathbf{Z}_q$ (we call such f* regular*),*[4] *we define*

$$p_{q,c} = q^{-2n} \left| \sum_{u \in \mathbf{Z}_q^n} \omega_c^{-\mathsf{RF}(v-u\cdot f)} \omega_q^{u\cdot(-f)} \right|^2$$

[3] In this computation, we took the worst case for ambiguous decoding (e.g. when both 01 and 10 decode to 00). If now 01 decode to 00 and 10 decode to 11, the distribution of ψ_U becomes $\Pr[\psi_U = 00] = \Pr[\psi_U = 01] = \frac{1}{2}$ and we obtain $p = \frac{1}{4}$.

[4] For q prime, every nonzero f is regular. For $q = 2^n$, every f with at least one odd component is regular.

We write $\varepsilon = b - \frac{q}{c}$. For $c = \mathcal{O}(1)$ and $q \to +\infty$, we have

$$p_{q,c} \geq q^{-2} \left(\left| \frac{\sin \frac{\pi c \varepsilon}{q}}{\sin \frac{\pi \varepsilon}{q}} \times \frac{\sin \frac{\pi b}{q}}{\sin \frac{\pi}{q}} \right| - \left| \frac{\sin \frac{\pi c \varepsilon}{q}}{\sin \frac{\pi}{q}} \right| \right)^2 = \frac{c^2}{\pi^2} \sin^2 \frac{\pi b}{q} - o\left(\frac{1}{q}\right)$$

If c divides q, we have $p_{q,c} \geq \frac{c^2}{\pi^2} \sin^2 \frac{\pi}{c}$.

Proof. Due to the fact that $u \mapsto v - u \cdot f$ being balanced, we have

$$p_{q,c} = q^{-2} \left| \sum_{x \in \mathbf{Z}_q} \omega_c^{-\mathsf{RF}(x)} \omega_q^{x-v} \right|^2 = q^{-2} \left| \sum_{x \in \mathbf{Z}_q} \omega_c^{-\mathsf{RF}(x)} \omega_q^{x} \right|^2 = q^{-2} \left| \sum_{k \in \mathbf{Z}_c} \sum_{x \in I_k} \omega_c^{-k} \omega_q^{x} \right|^2$$

Hence,

$$\begin{aligned}
p_{q,c} &\geq q^{-2} \left(\left| \sum_{k \in \mathbf{Z}_c} \sum_{x=a}^{a+b-1} \omega_c^{-k} \omega_q^{x+kb} \right| - \left| \sum_{x=a+q-(c-1)b}^{a+b-1} \omega_c^{-c+1} \omega_q^{x+(c-1)b} \right| \right)^2 \\
&= q^{-2} \left(\left| \sum_{k \in \mathbf{Z}_c} \sum_{x=0}^{b-1} \omega_c^{-k} \omega_q^{x+kb} \right| - \left| \sum_{x=b-c\varepsilon}^{b-1} \omega_q^{x} \right| \right)^2 \\
&= q^{-2} \left(\left| \sum_{k \in \mathbf{Z}_c} \sum_{x=0}^{b-1} \omega_q^{k\varepsilon} \omega_q^{x} \right| - \left| \sum_{x=b-c\varepsilon}^{b-1} \omega_q^{x} \right| \right)^2 \\
&= q^{-2} \left(\left| \frac{\sin \frac{\pi c \varepsilon}{q}}{\sin \frac{\pi \varepsilon}{q}} \times \frac{\sin \frac{\pi b}{q}}{\sin \frac{\pi}{q}} \right| - \left| \frac{\sin \frac{\pi c \varepsilon}{q}}{\sin \frac{\pi}{q}} \right| \right)^2
\end{aligned}$$

by using $\omega_q = e^{\frac{2i\pi}{q}}$ and $\omega_c = e^{\frac{2i\pi}{c}}$. We have $0 \leq \varepsilon < 1$, $c = \mathcal{O}(1)$. When $q \to +\infty$, this bound tends towards $\frac{c^2}{\pi^2} \sin^2 \frac{\pi b}{q}$ with a $o(\frac{1}{q})$ difference. □

In Fig. 10, we adapt the AJOP algorithm to make a KR-CCA attack using a single query to a quantum oracle making

$$|U \;\; V \;\; z\rangle \mapsto |U \;\; V \;\; z + \mathsf{Dec}(\mathsf{sk}, U, V)\rangle$$

where the addition is in $\mathbf{Z}_c^{n_V}$. It works assuming a special form of the cryptosystem. Compared to our GKZ-based attack, this is more restrictive but it uses a single oracle call and has a better success probability.

Theorem 21. *We consider meta-PKC constructions of the following form. We assume that $\mathcal{M} = \mathbf{Z}_c^{n_V}$, $\mathsf{decode}(W)_j = \mathsf{RF}(W_j)$, and that we can write $U = (U_1, \ldots, U_m) \in \mathbf{Z}_q^{n_U^1} \times \cdots \mathbf{Z}_q^{n_U^m}$, $(U \times \mathsf{sk})_j = U_{g(j)} \cdot f_j(\mathsf{sk})$ for some functions g and f_j, for $j = 1, \ldots, n_V$. Given a subset J over which g is injective, the algorithm on Fig. 10 recovers all $f_j(\mathsf{sk})$ for $j \in J$ with probability $p \geq p_{q,c}^{\#J}$ with $p_{q,c}$ defined in Lemma 20, when they are regular.*

Proof. We compute the probability $\Pr[\alpha]$:

$$\begin{aligned}
\Pr[\alpha] &= q^{-2n_U} c^{-n_V} \left\| \sum_{U,z} \left(\prod_{j\in J} \omega_c^{z_j - (\mathsf{pt}_U)_j} \right) \omega_q^{U\cdot\alpha} |\alpha \;\; V \;\; z\rangle \right\|^2 \\
&= q^{-2n_U} c^{-n_V} \sum_z \left| \sum_U \left(\prod_{j\in J} \omega_c^{z_j - (\mathsf{pt}_U)_j} \right) \omega_q^{U\cdot\alpha} \right|^2 \\
&= q^{-2n_U} \left| \sum_U \left(\prod_{j\in J} \omega_c^{-(\mathsf{pt}_U)_j} \right) \omega_q^{U\cdot\alpha} \right|^2 \\
&= q^{-2n_U} \left| \sum_U \left(\prod_{j\in J} \omega_c^{-\mathsf{RF}(V_j - U_{g(j)}\cdot f_j(\mathsf{sk}))} \right) \omega_q^{U\cdot\alpha} \right|^2 \\
&= q^{-2n_U} \left| \sum_U \prod_{j\in J} \left(\omega_c^{-\mathsf{RF}(V_j - U_{g(j)}\cdot f_j(\mathsf{sk}))} \omega_q^{U_{g(j)}\cdot\alpha_{g(j)}} \right) \prod_{j\notin g(J)} \omega_q^{U_j\cdot\alpha_j} \right|^2
\end{aligned}$$

For α such that $\alpha_{g(j)} = -f_j(\mathsf{sk})$ for all $j \in J$ and $\alpha_j = 0$ for $j \notin g(J)$, we have $\Pr[\alpha] = p_{q,c}^{\#J}$ when the $f_j(\mathsf{sk})$ are regular. □

Example 22 (Frodo continued). Frodo has $c = 2^\ell$. We regroup U by rows, i.e., $U_{g(j)}$ is the $g(j)$th row of U and $f_j(\mathsf{sk})$ is the $g(j)$th column of sk. We have $\bar{m}$ columns in sk. We recover $\#J$ columns with probability at least $p_{q,c}^{\#J}$. The Frodo-640 parameters are $q = 2^{15}$, $\ell = 2$, and $\bar{m} = 8$. This gives $\varepsilon = 0$ and $p_{q,c} \geq 81\%$. We can be greedy with $\#J = \bar{m}$ and fully recover sk with probability greater than 18% which we approximate as 2^{-2} in the table.

Example 23 (NewHope continued). To adapt the attack to NewHope, we observe

$$\mathsf{decode}(W)_j = \mathsf{RF}(|W_j| + |W_{j+256}|) = 1_{|W_j|+|W_{j+256}| \leq \frac{q}{2}}$$

and we cut an interval of $2q$ values into $c = 2$ intervals. We use $\#J = 1$. For simplicity, we use $J = \{0\}$ so $j = 0$. We modify the algorithm by sampling $\bar{V}_0$ and $\bar{V}_{256}$ and letting all other components of $\bar{V}$ constant, and by making the Fourier transform on the $\bar{V}_0$ and $\bar{V}_{256}$ registers as well. We obtain that we measure the final state

$$q^{-n_U} p^{-2} \sum_{\substack{\alpha, U\in S_U \\ \bar{V}_0, \bar{V}_{256}, \beta_1, \beta_2 \in \mathbf{Z}_p}} c^{-\frac{n_V}{2}} \sum_{z\in \mathbf{Z}_c^{n_V}} (-1)^{z_0 - (\mathsf{pt}_U)_0} \omega_q^{U\cdot\alpha} \omega_p^{\bar{V}\cdot\beta} |\alpha \;\; \beta \;\; z\rangle$$

where $\bar{V}\cdot\beta$ means $\bar{V}_0\beta_1 + \bar{V}_{256}\beta_2$. By similar computation as before, we obtain

$$\Pr[\alpha, \beta] = q^{-2n_U} p^{-4} \left| \sum_{U, \bar{V}_0, \bar{V}_{256}} (-1)^{(\mathsf{pt}_U)_0} \omega_q^{U\cdot\alpha} \omega_p^{\bar{V}\cdot\beta} \right|^2$$

Input: $J \subseteq \{1, \ldots, n_V\}$ and $V \in S_V$
Decryption oracle: $|U \;\; V \;\; z\rangle \mapsto |U \;\; V \;\; z + \mathsf{Dec}(\mathsf{sk}, (U, V))\rangle$

1: prepare the state $|0 \;\; V \;\; (1_{j\in J})_{j=1,\ldots,n_V}\rangle$ in $\mathbf{Z}_q^{n_U} \times \mathbf{Z}_q^{n_V} \times \mathbf{Z}_c^{n_V}$

2: make a quantum Fourier transform on all registers except V and obtain

$$q^{-\frac{n_U}{2}} \sum_{U \in S_U} c^{-\frac{n_V}{2}} \sum_{z' \in \mathbf{Z}_c^{n_V}} \left(\prod_{j \in J} \omega_c^{z'_j} \right) |U \;\; V \;\; z'\rangle$$

3: apply the decryption oracle and get (by writing $z = z' + \mathsf{pt}_U$)

$$q^{-\frac{n_U}{2}} \sum_{U \in S_U} c^{-\frac{n_V}{2}} \sum_{z \in \mathbf{Z}_c^{n_V}} \left(\prod_{j \in J} \omega_c^{z_j - (\mathsf{pt}_U)_j} \right) |U \;\; V \;\; z\rangle$$

4: make a quantum Fourier transform on the first register and obtain

$$q^{-n_U} \sum_{\alpha, U \in S_U} c^{-\frac{n_V}{2}} \sum_{z \in \mathbf{Z}_c^{n_V}} \left(\prod_{j \in J} \omega_c^{z_j - (\mathsf{pt}_U)_j} \right) \omega_q^{U \cdot \alpha} |\alpha \;\; V \;\; z\rangle$$

5: measure the first register and obtain α with some probability $\Pr[\alpha]$

Fig. 10. AJOP-based key recovery with quantum access to a decryption oracle.

Let s be such that $U \cdot s = (U \times \mathsf{sk})_0$. Clearly, there is a one-to-one mapping between sk and s. The probability that $\alpha = -s$, $\beta_1 = 1$, $\beta_2 = 0$ is

$$\begin{aligned}
\Pr[-s, 1, 0] &= q^{-2n_U} p^{-4} \left| \sum_{U, \bar{V}_0, \bar{V}_{256}} (-1)^{(\mathsf{pt}_U)_0} \omega_q^{-(U \times \mathsf{sk})_0} \omega_p^{\bar{V}_0} \right|^2 \\
&= q^{-4} p^{-4} \left| \sum_{u, v \in \mathbf{Z}_q, \bar{V}_0, \bar{V}_{256} \in \mathbf{Z}_p} \left(1 - 2 \cdot 1_{|u|+|v| \le \frac{q}{2}}\right) \omega_q^{u - V'_0} \omega_p^{\bar{V}_0} \right|^2 \\
&= 4q^{-4} p^{-2} \left| \sum_{\substack{u, v \in \mathbf{Z}_q \\ |u|+|v| \le \frac{q}{2}}} \omega_q^u \sum_{\bar{V}_0 \in \mathbf{Z}_p} \omega_q^{-V'_0} \omega_p^{\bar{V}_0} \right|^2
\end{aligned}$$

with $V'_j = \lceil \frac{q}{p} \bar{V}_j \rfloor$, $u = V'_0 - (U \times \mathsf{sk})_0$, and $v = V'_{256} - (U \times \mathsf{sk})_{256}$. We can compute experimentally this sum as $\Pr[-s, 1, 0] \approx 16.4\%$. We can also compute literally for $p = q$ (i.e., we ignore compression). The terms of the inner sum are

1. We have

$$\begin{aligned}\Pr[-s,1,0] &= 4q^{-4}\left|2\sum_{v=0}^{\frac{q-1}{2}}\sum_{u=-\frac{q-1}{2}+v}^{\frac{q-1}{2}-v}\omega_q^u-\sum_{u=-\frac{q-1}{2}}^{\frac{q-1}{2}}\omega_q^u\right|^2\\ &= 4q^{-4}\left|2\sum_{v=0}^{\frac{q-1}{2}}\frac{\omega_q^{\frac{q}{2}-v}-\omega_q^{-\frac{q}{2}+v}}{\omega_q^{\frac{1}{2}}-\omega_q^{-\frac{1}{2}}}-\frac{\omega_q^{\frac{q}{2}}-\omega_q^{-\frac{q}{2}}}{\omega_q^{\frac{1}{2}}-\omega_q^{-\frac{1}{2}}}\right|^2\\ &= 4q^{-4}\left|2\frac{\left(\omega_q^{\frac{q+1}{4}}-\omega_q^{-\frac{q+1}{4}}\right)^2}{\left(\omega_q^{\frac{1}{2}}-\omega_q^{-\frac{1}{2}}\right)^2}-\frac{\omega_q^{\frac{q}{2}}-\omega_q^{-\frac{q}{2}}}{\omega_q^{\frac{1}{2}}-\omega_q^{-\frac{1}{2}}}\right|^2\end{aligned}$$

Since $\omega_q^{\frac{q}{2}}=-1$ and $\omega_q^{\frac{q}{4}}=i$, we obtain

$$\Pr[-s,1,0]=16q^{-4}\left|\frac{\omega_q^{\frac{1}{4}}+\omega_q^{-\frac{1}{4}}}{\omega_q^{\frac{1}{2}}-\omega_q^{-\frac{1}{2}}}\right|^4=\left|\frac{2}{q\left(\omega_q^{\frac{1}{4}}-\omega_q^{-\frac{1}{4}}\right)}\right|^4=\frac{1}{q^4\sin^4\frac{\pi}{2q}}\sim\left(\frac{2}{\pi}\right)^4$$

which is also 16.4%. We also have $\Pr[s,-1,0]=\Pr[-s,1,0]$. Similarly, let s' be such that $U\cdot s'=(U\times\mathsf{sk})_{256}$. We have $\Pr[\mp s',0,\pm 1]=\Pr[s,1,0]$ and all four cases reveal sk. Hence, we recover sk with probability about 66%. The compression of V in NewHope does not modify the attack.

6 Conclusion

We have shown how to make efficient key recovery attacks against the weak version of many post-quantum cryptosystems under classical PCA mode or quantum CCA mode, even with one or two CCA queries.

When trying to adapt this attack strategy to various algorithms, we observed that a binary code followed by an encoding in the high bits in $\mathbf{Z}_q$ makes the attack more difficult. However, the repetition code helps the attacker quite a lot.

Our attacks do not work for some algorithms. For instance, KINDI does not encode the plaintext but rather a seed (our attack only works because decryption outputs the seed for some reason). Compressing V does not harm quantum attacks but makes our classical attack impossible. Compressing U sometimes makes the quantum attack harder (this is the case in Kyber [6] but not in Lizard [9]). Lepton does not decode on the entire domain. We let as future work to investigate if attacks are still possible in those cases. Attacking other NIST applications is also left as an open problem.

References

1. E. Alkim, L. Ducas, T. Pöppelmann, P. Schwabe. Post-Quantum Key Exchange - a New Hope. Available on https://eprint.iacr.org/2015/1092

2. G. Alagic, S. Jeffery, M. Ozols, A. Poremba. On Quantum Chosen Ciphertext Attacks and Learning with Errors. Available on https://eprint.iacr.org/2018/1185
3. A. Ambainis, L. Magnin, M. Roetteler, J. Roland. Symmetry-Assisted Adversaries for Quantum State Generation. *CoRR*, vol. abs/1012.2112, 2010. https://arxiv.org/pdf/1012.2112.pdf
4. R. El Bansarkhani. Kindi. http://kindi-kem.de/
5. A. Bauer, H. Gilbert, G. Renault, M. Rossi. Assessment of the Key-Reuse Resiience of NewHope. In *Topics in Cryptology CT-RSA'2019*, San Francisco, CA, USA, Lecture Notes in Computer Science 11405, pp. 272–292, Springer-Verlag, 2019. Full version: https://eprint.iacr.org/2019/075
6. J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J.M. Schanck, P. Schwabe, G. Seiler, D. Stehlé. Crystals-Kyber: a CCA-Secure Module-Lattice-Based KEM. In *IEEE European Symposium on Security and Privacy EuroS&P'2018*, London, UK, pp. 353–367, IEEE, 2018. Full version: https://eprint.iacr.org/2017/634
7. E. Bernstein, U. Vazirani. Quantum Complexity Theory. *SIAM Journal of Computing*, vol. 26(5), pp. 1411–1473, 1997.
8. J.W. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, D. Stebila. Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE. In *23rd ACM Conference on Computer and Communications Security*, Vienna, Austria, pp. 1006–1018, ACM Press, 2016. Full version: https://eprint.iacr.org/2016/659
9. J.H. Cheon, D. Kim, J. Lee, Y. Song. Lizard: Cut Off the Tail! A Practical Post-quantum Public-Key Encryption from LWE and LWR. In *Security and Cryptography for Networks SCN'2018*, Amalfi, Italy, Lecture Notes in Computer Science 11035, pp. 160–177, Springer-Verlag, 2018. Full version: https://eprint.iacr.org/2016/1126
10. J. Ding, S. Alsayigh, S. RV, S. Fluhrer, X. Lin. Leakage of Signal Function with Reuse Keys in RLWE Key Exchange. In *IEEE International Conference on Communications ICC'2017*, Paris, France, pp. 1–6, IEEE, 2017.
11. S. Fluhrer. Cryptanalysis of Ring-LWE Based Key Exchange with Key Share Reuse. Available on https://eprint.iacr.org/2016/085
12. E. Fujisaki, T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Advances in Cryptology CRYPTO'99*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 1666, pp. 537–554, Springer-Verlag, 1999.
13. E. Fujisaki, T. Okamoto. *Journal of Cryptology*, vol. 26, pp. 80–101, 2013.
14. A.B. Grilo, I. Kerenidis, T. Zijlstra. Learning with Errors is Easy with Quantum Samples. *CoRR*, vol. abs/1702.08255, 2017. https://arxiv.org/pdf/1702.08255.pdf
15. D. Hofheinz, K. Hövelmanns, E. Kiltz. A Modular Analysis of the Fujisaki-Okamoto Transformation. In *Theory of Cryptography TCC'17*, Baltimore MD, USA, Lecture Notes in Computer Science 10677–10678, pp. 341–371 vol. 1, Springer-Verlag, 2017. Full version: https://eprint.iacr.org/2017/604
16. D. Kirkwood, B.C. Lackey, J. McVey, M. Motley, J.A. Solinas, D. Tuller. Failure is not an Option: Standardization Issues for Post-Quantum Key Agreement. Presented at the NIST Workshop on Cybersecurity in a Post-Quantum World 2015 https://www.nist.gov/news-events/events/2015/04/workshop-cybersecurity-post-quantum-world. Available at https://csrc.nist.gov/csrc/media/events/workshop-on-cybersecurity-in-a-post-quantum-world/documents/presentations/session7-motley-mark.pdf

17. T. Lepoint. Algorithmic of LWE-Based Submissions to NIST Post-Quantum Standardization Effort. Presented at the Post-Scryptum Spring School 2018 https://postscryptum.lip6.fr/. Available at https://postscryptum.lip6.fr/tancrede.pdf
18. V. Lyubashevsky, C. Peikert, O. Regev. On Ideal Lattices and Learning With Errors Over Rings. In *Advances in Cryptology EUROCRYPT'10*, French Riviera, France, Lecture Notes in Computer Science 6110, pp. 54–72, Springer-Verlag, 2010.
19. L.T. Phong, T. Hayashi, Y. Aono, S. Moriai. LOTUS. https://www2.nict.go.jp/security/lotus/index.html
20. O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of the ACM*, vol. 56(6), 2009.
21. M. Seo, J.H. Park, D.H. Lee, S. Kim, S.-J. Lee. Emblem. https://pqc-emblem.org
22. P.W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, Santa Fe, New Mexico, U.S.A., pp. 124–134, IEEE, 1994.
23. N.P. Smart, M.R. Albrecht, Y. Lindell, E. Orsini, V. Osheter, K.G. Paterson, G. Peer. Lima 1.1: a PQC Encryption Scheme. https://lima-pq.github.io
24. R. Steinfeld, A. Sakzad, R.K. Zhao. Titanium. http://users.monash.edu.au/ rste/Titanium.html
25. E.E. Targhi, D. Unruh. Post-Quantum Security of the Fujisaki-Okamoto and OAEP Transforms. In *Theory of Cryptography TCC'16B*, Beijing, China, Lecture Notes in Computer Science 9985–9986, pp. 192–216 vol. 2, Springer-Verlag, 2016. Full version: https://eprint.iacr.org/2015/1210
26. Y. Yu, J. Zhang. Lepton: Key Encapsulation Mechanisms from a Variant of Learning Parity with Noise. NIST Round 1 submission to Post-Quantum Cryptography, 2017. Available from https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

A Post-Quantum Cryptosystems

We list here several algorithms for which we could adapt our attacks. The algorithms are available from

https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

For the KR-PCA attack, we estimate to $(\log_q \#S_V)\log_2(2\rho_+ + 1)$ the number of oracle calls. For the GKZ-based attack, the probability of success is estimated to $\frac{1}{(2\rho_+)^{\#I}} - \frac{1}{q^{\#I}}$. For the AJOP-based attack, the probability of success is $p_{q,c}^{\#J}$.

EMBLEM. EMBLEM-CPA [21] works with $S_A = \mathbb{Z}_q^{m\times n}$, $S_{\mathsf{sk}} = \mathbb{Z}_q^{n\times k}$, $S_B = \mathbb{Z}_q^{m\times k}$, $S_t = \mathbb{Z}_q^{v\times m}$, $S_U = \mathbb{Z}_q^{v\times n}$ and $S_V = \mathbb{Z}_q^{v\times k}$. The bilinear mappings are matrix multiplications. The message space is $\{0,1\}^\ell$ and a message is encoded by t-bit chunks. Each block of t-bits is padded with a 1 bit and 0 bits to match a length of $\log_2(q)$ bits. Then, all $\frac{\ell}{t}$ blocks are arranged in a $v \times k$ matrix. Thus, for a message pt, each $\log_2(q)$-bits element of the matrix $M = \mathsf{encode}(\mathsf{pt})$ is $\mathsf{pt}_{i,j}\|1\|00\ldots0$, where $\mathsf{pt}_{i,j}$ is a t-bit block of the original message. Decoding takes the t most significant bits of each element and concatenate them to obtain

the original message. Therefore, we have $\rho_- = \rho_+ = q2^{-t-1}$. Components of sk, t are sampled in $[-B, B]$ uniformly at random and components of d, e, f are sampled from the discrete Gaussian distribution on $\mathbb{Z}$ with standard deviation σ. This is similar to Frodo. Hence, we have nk unknowns and each δ gives vk equations. The GKZ-based attack with $\#I = 1$ recovers one column of n unknowns. The AJOP-based attack uses $\varepsilon = 0$, $c = 2^t$, and $\#J = k$. For 128-bit security, the following parameters are used: $m = 1\,003$, $n = 770$, $\ell = 256$, $q = 2^{24}$, $\sigma = 25$, $t = 8$, $B = 1$, v and k can be tuned such that $v \times k \times t = \ell = 256$, typically $v = 32, k = 1$. We compute $p_{q,c} \approx 1$.

R.EMBLEM-CPA is a variant of EMBLEM where the variables are considered as polynomials in X modulo $X^n + 1$ with coefficients in $\mathbb{Z}_q$. It has $S_{\mathsf{sk}} = S_A = S_B = S_t = S_U = \mathbb{Z}_q^n$ and $S_V = \mathbb{Z}_q^{\ell/t}$ with L_∞ norm. The bilinear mappings are polynomial multiplications. A message $m \in \{0,1\}^\ell$ is encoded as in EMBLEM-CPA, except that now the $\frac{\ell}{t}$ encoded blocks are polynomial coefficients and not matrix entries. As before, we have $\rho_- = \rho_+ = q2^{-t-1}$. There is a small subtlety at encryption and decryption: since $\mathsf{encode}(m) \in \mathbb{Z}_q^{\ell/t}$, we compute $V = \mathsf{trunc}(t \times B + f, \ell/t) + \mathsf{encode}(m)$ and $W = V - \mathsf{trunc}(U \times \mathsf{sk}, \ell/t)$, where $\mathsf{trunc}(x, l)$ takes only the first ℓ components of a vector x. Coefficients of sk, t are sampled in $[-B, B]$ uniformly at random and coefficients of d, e, f are sampled from a discrete Gaussian distribution on $\mathbb{Z}$ with standard deviation σ. For 128-bit security, the following parameters are proposed: $n = 463$, $\ell = 256$, $q = 2^{25}$, $\sigma = 25$, $t = 1$, $B = 1$. We have n unknowns and each δ give them all. The number of oracle calls is about $n(\log_2 q - t)$ in the classical attack. The probability of success in the quantum attack is $\frac{2^t}{q}$ for the GKZ-based one, and $p_{q,c}$ for the AJOP-based one.

KINDI. KINDI-CPA [4] works with the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$. It has $S_A = \mathcal{R}_q^{\ell^2}$, $S_{\mathsf{sk}} = S_B = S_t = S_U = \mathcal{R}_q^\ell$ and $S_V = \mathcal{R}_q$. The norm is L_∞. The bilinear mappings are matrix multiplications and scalar product when the elements are vectors, where elements are considered as polynomials in $\mathcal{R}_q$. The public key B is compressed by dropping the k least significant bits of all coefficients.

The encoding of a message pt is more complex than in other LWE schemes. A random polynomial s_1 with binary coefficients is uniformly sampled from $\mathcal{R}_2$. This polynomial is used as a seed for a PRNG function (Shake) that returns a one time-pad $\bar{u}$ and the value t. The message is encrypted into $u = \bar{u} \oplus \mathsf{pt}$ by one-time pad and encoded in a value $e \in \mathcal{R}_q^\ell$ and $f \in \mathcal{R}_q$. The ciphertexts are computed as $(U, V) = (t \times A + e, t \times B + f + \mathsf{encode}(s_1))$ where $\mathsf{encode}(s_1) = L \cdot s_1$ with $L = \frac{q}{2}$. Then, the decryption $V - U \times \mathsf{sk}$ recovers s_1 thus t then e and f, then u. The value of s_1 also gives $\bar{u}$ which decrypts u into pt. We have $\rho_- = \rho_+ = \frac{q}{4}$. Elements of A are sampled uniformly at random from $\mathcal{R}_q$, elements of sk, d, t and the one-time pad are sampled uniformly at random from $\mathcal{R}_q$ where the coefficients of the polynomials are in $[-p, p)$ and e, f are derived from the message xored with the one-time pad. For KINDI256-CPA, the parameters used are $n = 256, \ell = 3, p = 4, k = 2, q = 2^{14}$. In our KR-PCA attack, we have to be aware that tampering V results in having junk decryption in the last bits, so we

must assume that the PCO oracle ignores those last bits. Adapting the quantum attacks may not be possible because they need s_1 and we cannot recover s_1 from pt. Surprisingly, the decryption in KINDI kindly returns s_1 in addition to the plaintext. So, the quantum attacks work well, with $\varepsilon = 0$, $c = 2$, $\#I = \#J = 1$.

LIMA. LIMA-CPA [23] has $S_{\mathsf{sk}} = S_A = S_B = S_t = S_U = S_V = \mathbb{Z}_q^n$ with the L_∞ norm. Elements are considered as polynomials in $\mathbb{Z}_q[X]/\langle g\rangle$. LIMA-CPA comes in two variants, namely LIMA-2p and LIMA-sp. In LIMA-2p, the polynomial g is $X^n + 1$ with $q \equiv 1 \mod 2n$ and in LIMA-sp, g is a trinomial of degree $n = p - 1$ and p is a safe prime (i.e. $p = 2q + 1$ for a prime q). Each bit of a message is encoded into a 0 or $q/2$. Therefore, we have $\rho_- = \rho_+ = \frac{q}{4}$. The sparse elements sk, d, t, e, f are sampled in $\{-B, \ldots, B\}$ from an approximation of a centered discrete Gaussian distribution of standard deviation $\sigma = \sqrt{(B+1)/2}$. A subtlety is that a pair (t, e) is accepted only if for $y_i = t_i + e_i$, it has

$$\left|\sum_{i=0}^{n-1} y_i\right| \leq 11 \times \sqrt{2 \times n} \times \sigma$$

for LIMA-2p and

$$\left|\sum_{i=0}^{k} y_i + \sum_{i=1}^{n-1} y_i + \sum_{i=k+2}^{n-1} y_i\right| \leq 11 \times \sqrt{4 \times n} \times \sigma$$

for LIMA-sp and any $k \in \{0, \ldots, n-1\}$. For a classical 227-bit security LIMA-2p-CPA, the parameters used are $B = 19, n = 1\,024, q = 133\,121$. For a classical 152-bit security, LIMA-sp-CPA uses $B = 19, n = 1\,018$ and $q = 12\,521\,473$. The quantum attacks work with $c = 2$, $\#I = \#J = 1$, and $p_{q,c} = 41\%$.

Lizard. Lizard-CPA [9] has $S_A = \mathbb{Z}_q^{m \times n}$, $S_{\mathsf{sk}} = \{-1, 0, 1\}^{n \times \ell}$, $S_B = \mathbb{Z}_q^{m \times \ell}$, $S_t = \{-1, 0, 1\}^m$, $S_U = \mathbb{Z}_p^n$, and $S_V = \mathbb{Z}_p^\ell$. The norm is L_∞. Bilinear mappings are matrix multiplications in these structures. Each bit of a message is encoded into 0 or $q/2$ but U, V are scaled by a p/q factor, then $\mathsf{pt} \in \{0,1\}^\ell$ is encoded into 0 or $p/2$. Therefore, we have $\rho_- = \rho_+ = p/4$. Actually, encryption is based on the LWR problem, hence with deterministic e and f. Decryption has form $\mathsf{Dec}(\mathsf{sk}, U, V) = \lceil \frac{2}{p}(V - U \times \mathsf{sk}) \rfloor$, which fits the quantum attacks. Elements of sk are sampled from the distribution $\Pr[x = 1] = \Pr[x = -1] = \gamma/2$, $\Pr[x = 0] = 1 - \gamma$, elements of d are sampled in $\mathbb{Z}_q$ from a discrete Gaussian distribution of parameter $\sigma = \alpha q$, t is sampled uniformly at random in $\{x \in \{-1, 0, 1\}^m : \mathsf{HW}(x) = h\}$, where $\mathsf{HW}(x)$ counts the number of non-zero elements of x, and e, f are zero. Proposed parameters are $n = 544$, $m = 840$, $q = 1\,024$, $p = 256$, $\ell = 256$, $\gamma = \frac{1}{2}$, $\alpha = \frac{1}{171}$, and $h = 128$. The quantum attacks work with $\varepsilon = 0$, $c = 2$, $\#I = \#J = 1$.

RLizard-CPA is a variant of Lizard which works with rings. It has $S_A = S_B = \mathbb{Z}_q^n, S_U = S_V = \mathbb{Z}_p^n$, and $S_{\mathsf{sk}} = S_t = \{-1, 0, 1\}^n$. Elements are considered as polynomials in these structures and bilinear mappings are polynomial multiplications

in the corresponding ring. Messages are encoded similarly as in Lizard-CPA. Elements sk, t are sampled uniformly at random in $\{x \in \{-1,0,1\}^m : \mathsf{HW}(x) = h\}$ with $h = h_{\mathsf{sk}}$ and $h = h_t$, respectively. Coefficients of d are sampled according to a discrete Gaussian distribution of parameter σ in Z_q. Proposed parameters are $n = 1\,024$, $q = 1\,024$, $p = 256$, $\alpha = \frac{1}{154}$ and $h_{\mathsf{sk}} = h_t = 128$.

LOTUS. LOTUS-PKE-CPA [19] is the same as Lindner-Peikert scheme. We have $S_A = \mathbb{Z}_q^{n \times n}$, $S_{\mathsf{sk}} = S_B = \mathbb{Z}_q^{n \times \ell}$, $S_t = S_U = \mathbb{Z}_q^n$, and $S_v = \mathbb{Z}_q^\ell$ with the L_∞ norm. Each bit of a message is multiplied by $\lfloor \frac{q}{2} \rfloor$. Elements of sk, d, t, e, f are sampled from a centered discrete Gaussian distribution of standard deviation σ. Therefore, we have $\rho_+ = \rho_- = \lfloor \frac{q}{4} \rfloor$. For LOTUS128-CPA, we have $n = 576$, $q = 8\,192$, $\ell = 128$, $\sigma = 3$. For key recovery, we have $n \times \ell$ unknowns and ℓ equations for each sample δ_i, hence we need n samples. The quantum attacks work with $\varepsilon = 0$, $c = 2$, $\#I = \#J = 1$.

Titanium. Let $\mathcal{R}_{q,n}$ be the set of polynomials in X with degree less than n and coefficients in $\mathbb{Z}_q$. Titanium has $S_A = \mathcal{R}_{q,n}^m$, $S_{\mathsf{sk}} = \mathcal{R}_{q,n+d+k-1}$, $S_B = \mathcal{R}_{q,d+k}^m$, $S_t = \mathcal{R}_{q,k+1}^m$, $S_U = \mathcal{R}_{q,n+k}$ and $S_V = \mathcal{R}_{q,d}$ with the L_∞ norm. The bilinear mappings use the middle product $\odot$ defined as follows: Let $a \in \mathcal{R}_{q,d_a}$ and $b \in \mathcal{R}_{q,d_b}$ s.t. $d_a + d_b - 1 = d + 2k$ for some integers d_a, d_b, d, k. The middle product $\odot_d : \mathcal{R}_{q,d_a} \times \mathcal{R}_{q,d_b} \to \mathcal{R}_{q,d}$ is the map

$$a \odot_d b = \left\lfloor \frac{(a \times b) \mod X^{k+d}}{X^k} \right\rfloor$$

i.e. we take the d terms of $a \times b$ of degree $k, k+1, \ldots, k+d-1$ and divide by X^k. Titanium extends it to vector multiplication as the dot product with $\odot_d$ for component multiplications and to polynomial-vector multiplication as the component-wise middle product with the polynomial. All bilinear mappings are middle products as described above, except for the $S_t \times S_A \to S_U$, which is the dot product with polynomial multiplication in $\mathbb{Z}_q[X]$. A message pt is encoded as a polynomial in $\mathcal{R}_{2,d}$ with each coefficient scaled by $\lfloor \frac{q}{p} \rfloor$. Therefore, we have $\rho_- = \rho_+ = \lfloor \frac{q}{p} \rfloor / 2$. The secret key sk is sampled uniformly at random in S_t and d is sampled by taking the difference of the Hamming weight of two uniformly distributed η-bits values, this approximates a discrete Gaussian distribution. For t, $N_t = (k+1) \times m$ coefficients need to be sampled in $\mathbb{Z}_q$. In order to tune the variance, N_1 of them are sampled uniformly in $\{-B_1/2, \ldots, B_1/2\} \setminus \{0\}$ and $N_t - N_1$ of them are sampled uniformly in $\{-B_2/2, \ldots, B_2/2\} \setminus \{0\}$. The elements e, f are null. For TitaniumStd128-CPA [24] with NIST security level I, the parameters are $n = 1\,024$, $k = 511$, $d = 256$, $m = 9$, $q = 86\,017$, $p = 2$, $\eta = 4$, $N_1 = 3\,816$, $B_1 = 2^6$, $B_2 = 2^7$. The quantum attacks work with $c = p$ and $\#I = \#J = 1$.