An Efficient Key Mismatch Attack on the NIST Third Round Candidate Kyber

Yue Qin[a], Chi Cheng [a,*], Jintai Ding[b] and Xinyi Huang[c]

[a]School of Computer Science, China University of Geosciences, Wuhan, 430074, China
[b]Department of Mathematical Sciences, University of Cincinnati, Cincinnati, 45219, USA
[c]College of Mathematics and Informatics, Fujian Normal University, Fuzhou, 350117, China

ARTICLE INFO

Keywords:
Modular-LWE
Kyber
key reuse attacks
security analysis

ABSTRACT

Just recently, Kyber was selected as the third round candidate for NIST's postquantum cryptography standard. In order to ensure the security of Kyber, it is very important to evaluate its security under various strict conditions. In this paper, we propose an efficient key mismatch attacks on Kyber, which can recover one participant's secret key if the public key is reused. We first define the oracles in which the adversary is able to launch the attacks. Then, we showed how the adversary recovered the coefficients in the secret key by accessing the oracle multiple times. Furthermore, we propose two strategies to reduce the queries and time in recovering the secret key. It turns out that using key mismatch attacks to break Kyber is much easier than NIST's second round candidate NewHope. Our implementations have demonstrated the efficiency of the proposed attacks and verified our findings. Another interesting observation from the attack is that in the most secure Kyber-1024, it is easier to recover each coefficient compared with that in Kyber-512 and Kyber-768. Specifically, for Kyber-512 on average we recover each coefficient with 2.7 queries, while in Kyber-1024 and 768, we only need 2.4 queries. This demonstrates further that implementations of LWE based schemes in practice is very delicate.

1. Introduction

In recent years, quantum computing has developed rapidly [18]. On one hand, the development brings us powerful computing and information processing capabilities. On the other hand, it threatens the security of cryptographic algorithms widely used in Internet and communication security today. For example, the polynomial-time quantum algorithm for integer factorization proposed by Shor in 1994 can be used to break the public-key cryptography widely used today (e.g. the RSA scheme). To deal with the security threats from quantum computers, the industry and academia have begun to pay attention to post-quantum cryptographic (PQC) algorithms and apply them in the real world to test their security. For example, Google applied PQC scheme to chrome to protect it from the potential threat of quantum computers [17]. Since 2016, the National Institute of Standards and Technology (NIST) has begun the process of standardizing PQC algorithms [24]. Until January 30, 2019, NIST announced that 26 candidates had entered the second round of NIST PQC standardization [3]. Most candidate algorithms are lattice-based, code-based, multi-variate-based, and symmetric-based. Among these, lattice-based ones are regarded as one of the most promising algorithms.

In 1996, Ajtai showed in his seminal work [1] that lattices can actually be used to construct cryptographic primitives. In addition, the advantages of the lattice-based structure include a strong safety guarantee based on the worst-case hardness and its relatively high efficiency. This has aroused great interest in lattice-based cryptography from researchers. The first lattice-based public key encryption scheme was proposed by Regev in 2005 [27], and he also introduced the Learning with Errors (LWE) problem. Since then, many follow-up works have been inspired by this, such as the Ring-Learning with Errors problem (R-LWE) proposed by Lyubashevsky, Peikert and Regev [23], and the Module-Learning with Errors problem (M-LWE) proposed by Langlois and Stehlé [22]. On the second-round list of the PQC selection, Frodo is based on the LWE problem [5], NewHope is based on the R-LWE problem [4], while Kyber is based on the M-LWE problem [7].

*Corresponding author
qy52hz@cug.edu.cn (Y. Qin); chengchi@cug.edu.cn (C.C.); jintai.ding@gmail.com (J. Ding); xyhuang@fjnu.edu.cn (X. Huang)
ORCID(s):

Kyber is a post-quantum key encapsulation mechanism (KEM), which is part of the CRYSTALS (CRYptographic SuiTe for Algebraic LatticesS). It was first proposed by Roberto, Joppe, Léo et al. in 2017 [6], and submitted to NIST's first round PQC standardization process. Then, in the second-round kyber is modified to remove the public-key compression due to the problem pointed out by NIST. Another tweaked part is to change the module q from 7681 to 3329 and update the definition of the NTT. Although the value of q becomes smaller, it is claimed that this does not affect its security, and it also ensures that NTT-based multiplication can be performed quickly. In addition, they also change the noise parameter η to 2, which makes the coefficients of Kyber's secret key in the range of [-2,2]. Kyber enjoys the benefits of flexibility, security and high performance. Just recently, NIST announced that Kyber entered into the third round of NIST PQC standardization [2]. In 2019, the IBM research team tested Kyber successfully on a state-of-the-art prototype IBM TS1160 tape drive, thus achieving the world's first quantum computing security tape drive [20]. Tape drives are the most advanced storage technologies today, and IBM is developing it with quantum safe to protect the long-term security of stored data.

To ensure the security of candidate algorithms, NIST requires that they must satisfy certain security goals, one of which is to thwart key reuse attacks [25]. The reason is that in some special situations in actual life, we need to reuse the key. For example, the 0-RTT (zero round-trip time) mode in TLS 1.3 [28]. In addition, Kirkwood et al. from the US National Security Agency (NSA) pointed out that if the key is reused, there may exist key reuse attacks against the lattice-based post-quantum key exchange protocols [21]. Therefore, we must consider the security threats that the key reuse attacks bring to the lattice-based candidate algorithms.

In a key reuse attack, the public and its corresponding private key of one of the participants must be reused. Fluhrer stated in [16] that the R-LWE-based key exchange protocol could be broken if the public key is reused. Next, in [13], Ding, Alsayigh and Saraswathy took the lead in verifying this idea. They found that the public signal message may reveal the information about the secret key, and applied the signal leakage attack to the key exchange protocol in [15]. Later, Ding, Fluhrer and Saraswathy proposed key mismatch attack in [14], which is another branch of key reuse attack. In the key mismatch attack, we can recover the private key by comparing whether the shared keys between the two participants match. In [9], Bauer et al. proposed a key mismatch attack against NewHope KEM [4]. But their attack cannot correctly recover the private keys. Later, Qin, Cheng, and Ding proposed an improved and complete key mismatch attack to solve their problems and recover all private keys in NewHope KEM with extremely high accuracy and efficiency [26]. In addition, Băetu et al. introduced an effective key reuse attack that can attack many weak versions of post-quantum cryptosystems in NIST first round [8]. Based on this work, Dumittan and Vaudenay designed a new key reuse attack that can thwart against the NIST second round candidates which are chosen-plaintext attack (CPA) securel[19]. [11] and [12] have also proposed the attacks that use decryption failures, which work against the chosen-ciphertext attack (CCA) secure lattice-based schemes.

So far as we know the aforementioned key mismatch attacks are proposed against the R-LWE-based KEMs. Since Kyber is designed in a different algebraic structure, it is appealing to assess Kyber's security when the public key is reused.

Differences of key mismatch attacks on NewHope and Kyber. We find that it is much easier to use key mismatch attacks to break Kyber than that on NewHope, due to the different design structures of Kyber and Newhope. First, both Compress and Decompress functions are used in Kyber and NewHope. But the additional Encode and Decode functions used in NewHope, which process four coefficients at the same time, bring challenges to the key mismatch attacks on NewHope. Second, the ranges of coefficients in Kyber's secret key is from -2 to 2, while in NewHope it is from -8 to 8. The smaller ranges of Kyber result in less queries to launch the attack. Specifically, on average we only need 2,475 queries to recover all the coefficients in Kyber-1024. While in Newhope-1024, which achieves the same security as that in Kyber-1024, on average we need $882,794$ queries. From another perspective, in Kyber-1024 on average we need 2.4 queries to recover each coefficient, while in Newhope-1024, 862.1 queries are needed.

Contributions. In this paper, we propose an efficient key reuse attack on the Kyber KEM. The main contributions of this paper include:

1. We precisely define the security oracle in which our proposed attack succeeds;
2. We propose key mismatch attacks on the Kyber-KEM with three security levels, i.e., the Kyber-512, Kyber-768, and Kyber-1024. Furthermore, we propose two strategies, to significantly reduce the needed queries and time in the original attack.
3. The implementations have demonstrated the efficiency of the proposed attacks. Our best record is that we can

Table 1
Parameter choices in Kyber

	n	k	q	$d_{\mathbf{P}_B}$	$d_{\mathbf{v}_B}$	Security
Kyber-512	256	2	3329	10	3	1 (AES-128)
Kyber-768	256	3	3329	10	4	3 (AES-192)
Kyber-1024	256	4	3329	11	5	5 (AES-256)

recover one coefficient in the secret key in only 2 queries. An interesting observation from the implementations is that in the most powerful Kyber-1024, it is easier to recover each coefficient compared with that in Kyber-512 and Kyber 768. Specifically, in Kyber-512 on average we recover each coefficient in 2.7 queries, while in Kyber-768 and 1024, we only need 2.4 queries.

Organization of this paper. In Section 2, we introduce the basic notations and the underlying hard problems, as well as the Kyber KEM. Security analysis of Kyber with three security levels are given in Section 3, 4, and 5, respectively. In Section 6 we go on proposing two improved attacks that effectively reduce the needed time and queries in the original attacks. The experiments in Section 7 demonstrate the efficiency of our proposed attacks, and the conclusion is given in Section 8.

2. Preliminaries

2.1. Notations

Set $\mathbb{Z}_q$ the ring with all elements are integers modulo q, and $\mathbb{Z}_q[x]$ the polynomial ring, where all the polynomials in $\mathbb{Z}_q[x]$ are with coefficients selected from $\mathbb{Z}_q$. Then, we further define the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n+1)$, in which for every polynomial $f(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1} \in \mathbb{R}_q$, each coefficient $a_i \in \mathbb{Z}_q$ $(0 \leq i \leq n-1)$ and the polynomial additions and multiplications are operated modulo $x^n + 1$. All polynomials are in bold-lower case, and we treat a polynomial $\mathbf{c} \in \mathcal{R}_q$ the same with its vector form $(\mathbf{c}[0], \cdots, \mathbf{c}[n-1])$, here $\mathbf{c}[i]$ $(0 \leq i \leq n-1)$ represents the i-th coefficient of the polynomial $\mathbf{c}$. By default, all the vectors will be column vectors. Bold upper-case letters represent matrices. For a matrix $\mathbf{A} \in R_q^{k\times k}$, we denote $\mathbf{A}^T \in R_q^{k\times k}$ as its transpose. The operation $\lfloor x \rfloor$ represents the maximum integer not exceeding x, and $\lceil x \rfloor$ is the rounding function, i.e. $\lceil x \rfloor = \lfloor x + \frac{1}{2} \rfloor$.

2.2. LWE & R-LWE & M-LWE

In [27] Regev introduced a reduction from solving the worst case problem in lattices to solving LWE problem in the average case. But the matrices used in the LWE problem incur heavy computation and communication costs. However, the R-LWE problem replaces the matrices with the polynomials in the ring, so these problems do not exist. The hardness of R-LWE is similar to LWE, which is based on a reduction from solving the worst case problem in idea lattices to solving R-LWE problem in the average case.

The M-LWE problem can be viewed as a combination of the LWE problem and Ring-LWE problem. Therefore, M-LWE enjoys the advantages of easy scalability in LWE problem and the high efficiency in R-LWE problem. To be specific, for a matrix $\mathbf{A} \in R_q^{k\times k}$, $\mathbf{s}, \mathbf{e} \in \mathcal{B}_\eta^k$ (Here $\mathcal{B}_\eta$ is a centered binomial distribution), the M-LWE problem is to distinguish $(\mathbf{A}, \mathbf{B} = \mathbf{A}\mathbf{s} + \mathbf{e}) \in R_q^{k\times k} \times R_q^k$ from uniformly selected $(\mathbf{A}, \mathbf{B}) \in R_q^{k\times k} \times R_q^k$. Kyber can change security levels by simply changing the value of k, and the lattice used in Kyber has less algebraic structure than that in Ring-LWE.

2.3. The Kyber KEM

Kyber is an IND-CCA2-secure KEM, which is a part of the Cryptographic Suite for Algebraic Cipher Suite (CRYSTALS) [6, 10, 7]. Kyber's security is based on the hardness of solving the M-LWE problem, which makes Kyber different from NewHope [4]. In Kyber, there is a public matrix $\mathbf{A}$ of size $k \times k$, and its elements are polynomials selected from R_q. The secret key $\mathbf{s}$ and the errors $\mathbf{e}$ are k-dimensional vectors with elements selecting from $\mathcal{B}_\eta$. Here $\mathcal{B}_\eta$ is a centered binomial distribution with $\eta = 2$, which can be simply calculated from $\sum_{i=1}^{2}(b_i - b_i')$, here b_i and b_i'

are randomly selected from $\{0, 1\}$. Specifically, the coefficients of the secret $\mathbf{s}$ are integers in the range $[-2, 2]$. This has the advantage that the security level of Kyber can be shifted by simply modifying k. More specifically, there are three security levels in Kyber: Kyber-512, Kyber-768 and Kyber-1024, corresponding to $k = 2$, $k = 3$, and $k = 4$, respectively. Table 1 lists the parameters choices for different security levels of Kyber. The parameter n is always set as 256 and q is always 3329.

In the following, we first give the definition of two functions: $\mathbf{Compress}_q(x, d)$ and $\mathbf{Decompress}_q(x, d)$.

Definition 1. *The Compression function:* $\mathbb{Z}_q \rightarrow \mathbb{Z}_{2^d}$*:*

$$\mathbf{Compress}(x, d)_q = \left\lceil \frac{2^d}{q} \cdot x \right\rfloor \pmod{2^d}. \tag{1}$$

Definition 2. *The Decompression function:* $\mathbb{Z}_{2^d} \rightarrow \mathbb{Z}_q$*:*

$$\mathbf{Decompress}(x, d)_q = \left\lceil \frac{q}{2^d} \cdot x \right\rfloor. \tag{2}$$

In the two functions, their input x is selected from $\mathbb{Z}_q$. When the input is a polynomial, it means we will operate coefficients of the polynomial one by one. Both equations (1) and (2) can be extended to the polynomial $\mathbf{f} = f_0 + f_1 x + \cdots + f_{n-1}x^{n-1}$ as follows:

$$\mathbf{Compress}(\mathbf{f}, d)_q = \left(\left\lceil \frac{2^d}{q} \cdot f_0 \right\rfloor \pmod{2^d}, \left\lceil \frac{2^d}{q} \cdot f_1 \right\rfloor \pmod{2^d}, \cdots, \left\lceil \frac{2^d}{q} \cdot f_{n-1} \right\rfloor \pmod{2^d} \right), \tag{3}$$

$$\mathbf{Decompress}(\mathbf{f}, d)_q = \left(\left\lceil \frac{q}{2^d} \cdot f_0 \right\rfloor, \left\lceil \frac{q}{2^d} \cdot f_1 \right\rfloor, \cdots, \left\lceil \frac{q}{2^d} \cdot f_{n-1} \right\rfloor \right). \tag{4}$$

Similarly, if the input is a matrix, we deal with each column one by one. For matrix $\mathbf{M}^T \in R_q^{k \times k} = (\mathbf{m}_0, \mathbf{m}_1, \cdots, \mathbf{m}_{k-1})^T$, equations (1) and (2) can be extended to:

$$\mathbf{Compress}(\mathbf{M}^T, d)_q = \left(\mathbf{Compress}(\mathbf{m}_0, d)_q, \mathbf{Compress}(\mathbf{m}_1, d)_q, \cdots, \mathbf{Compress}(\mathbf{m}_{k-1}, d)_q \right), \tag{5}$$

and

$$\mathbf{Decompress}(\mathbf{M}^T, d)_q = \left(\mathbf{Decompress}(\mathbf{m}_0, d)_q, \mathbf{Decompress}(\mathbf{m}_1, d)_q, \cdots, \mathbf{Decompress}(\mathbf{m}_{k-1}, d)_q \right). \tag{6}$$

The value of d is set as $d_{\mathbf{P}_B}$ or $d_{\mathbf{v}_B}$ in different security levels of Kyber in Table 1.

In Table 2, we describe the details of the Kyber IND-CCA2-secure KEM. To simplify the security analysis, we remove the number theory transformation (NTT) that has nothing to do with security and is only used to accelerate polynomial multiplication in Table 2. The three different functions G(·), H(·) and KDF(·) in Table 2 use SHA3-256, SHA3-512 and SHAKE-256, respectively. The Kyber KEM consists of three parts:

(1) Alice first randomly chooses a 32-bit z and call Gen() to generate its key pair. In Step 2 of Table 2, Alice first generates a matrix $\mathbf{A} \in R_q^{k \times k}$, then she will select $\mathbf{s}'_A$ and $\mathbf{e}_A$ uniformly at random from $\mathcal{B}_\eta$ to compute the public key $\mathbf{P}_A = \mathbf{A}\mathbf{s}'_A + \mathbf{e}_A$. The resulted $\mathbf{s}_A$ is computed as $(\mathbf{s}'_A || \mathbf{P}_A || \mathrm{H}(\mathbf{P}_A) || z)$. The output $(\mathbf{s}_A, \mathbf{P}_A)$ is the key pair and $\mathbf{P}_A$ will be sent to Bob.t

(2) After receiving $\mathbf{P}_A$ sent by Alice, Bob will first generate $\mathbf{m} \xleftarrow{\$} \{0, 1\}^{256}$ and $(K, r) = \mathrm{G}(\mathrm{H}(\mathbf{P}_A), \mathbf{m})$. Then he will use $\mathbf{P}_A, \mathbf{m}$ and r as input to call Enc(). In Step 6 of Table 2, we can see that, Bob will generate a matrix $\mathbf{A} \in R_q^{k \times k}$ first. Subsequently, he will select $\mathbf{s}_B$, $\mathbf{e}_B$ and $\mathbf{e}'_B$ uniformly at random, and compute a public key $\mathbf{P}_B = \mathbf{A}^T \mathbf{s}_B + \mathbf{e}_B$ as well as $\mathbf{v}_B = \mathbf{P}_A^T \mathbf{s}_B + \mathbf{e}'_B + \mathbf{Decompress}_q(\mathbf{m}, 1)$. After this, Bob will compress $\mathbf{P}_B$, $\mathbf{v}_B$ to $\mathbf{c}_1$, $\mathbf{c}_2$, respectively. In the end, he will send $\mathbf{P}_B$ and $(\mathbf{c}_1, \mathbf{c}_2)$ to Alice, and compute the shared key k_B.

(3) When Alice receives $\mathbf{P}_B$ and $(\mathbf{c}_1, \mathbf{c}_2)$ sent by Bob, she will generate $z \xleftarrow{\$} \{0, 1\}^{256}$ first. Then, she will use her secret key $\mathbf{s}_A$ and $(\mathbf{c}_1, \mathbf{c}_2)$ as inputs to call Dec() to get m'. According to Step 9 of Table 2, Alice can obtain $\mathbf{u}_A$ and $\mathbf{v}_A$ by decompressing $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively. With the output $\mathbf{m}' = \mathbf{Compress}_q(\mathbf{v}_A - \mathbf{s}_A^T \cdot \mathbf{u}_A, 1)$, Alice will use it to generate (K', r'). She then uses $\mathbf{P}_A, (K', r')$ as input to call Enc() and get the returned $(\mathbf{c}'_1, \mathbf{c}'_2)$. Finally, Alice calculates her shared key k_A after checking that $(\mathbf{c}_1, \mathbf{c}_2)$ and $(\mathbf{c}'_1, \mathbf{c}'_2)$ are equal.

In the following, we will use the key mismatch attack to assess the security of the Kyber KEM when the public key $\mathbf{P}_A$ is reused. Due to the different security parameters in the Compress and Decompress functions, we need to propose our attacks on these three security levels one by one.

Table 2
Kyber IND-CCA2-Secure KEM

Alice		Bob
1. $z \xleftarrow{\$} \{0,1\}^{32}$		
2. $\triangleright$Gen()		
2.1 Generate matrix $\mathbf{A} \in R_q^{k\times k}$		
2.2 Sample $\mathbf{s}'_A, \mathbf{e}_A \in R_q^k$		
2.3 $\mathbf{P}_A = \mathbf{A}\mathbf{s}'_A + \mathbf{e}_A$		
2.4 Output $(\mathbf{s}'_A, \mathbf{P}_A)$		
3. $\mathbf{s}_A = (\mathbf{s}'_A \vert\vert \mathbf{P}_A \vert\vert \mathrm{H}(\mathbf{P}_A) \vert\vert z)$	$\xrightarrow{\mathbf{P}_A}$	4. $\mathbf{m} \xleftarrow{\$} \{0,1\}^{256}$
		5. $(K, r) = \mathrm{G}(\mathrm{H}(\mathbf{m}, \mathbf{P}_A))$
		6. $\triangleright\mathrm{Enc}(\mathbf{P}_A, \mathbf{m}, r)$
		6.1 Generate matrix $\mathbf{A} \in R_q^{k\times k}$
		6.2 Sample $\mathbf{s}_B, \mathbf{e}_B \in R_q^k$
		6.3 Sample $\mathbf{e}'_B \in R_q$
		6.4 $\mathbf{P}_B = \mathbf{A}^T\mathbf{s}_B + \mathbf{e}_B$
		6.5 $\mathbf{v}_B = \mathbf{P}_A^T\mathbf{s}_B + \mathbf{e}'_B + \mathbf{Decompress}_q(\mathbf{m}, 1)$
		6.6 $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$
		6.7 $\mathbf{c}_2 = \mathbf{Compress}_q(v_B, d_{v_B})$
		6.8 Output $(\mathbf{c}_1, \mathbf{c}_2)$
8. $\triangleright\mathrm{Dec}(\mathbf{s}_A, (\mathbf{c}_1, \mathbf{c}_2))$	$\xleftarrow{(\mathbf{c}_1,\mathbf{c}_2)}$	7. $k_B = \mathrm{KDF}(K, \mathrm{H}(\mathbf{c}_1, \mathbf{c}_2))$
8.1 $\mathbf{u}_A = \mathbf{Decompress}_q(\mathbf{c}_1, d_{\mathbf{P}_B})$		
8.2 $\mathbf{v}_A = \mathbf{Decompress}_q(\mathbf{c}_2, d_{v_B})$		
8.3 $\mathbf{m}' = \mathbf{Compress}_q(\mathbf{v}_A - \mathbf{s}_A^T \cdot \mathbf{u}_A, 1)$		
8.4 Output $\mathbf{m}'$		
9. $(K', r') = \mathrm{G}(\mathrm{H}(\mathbf{m}', \mathbf{P}_A))$		
10. $(\mathbf{c}'_1, \mathbf{c}'_2) = \mathrm{Enc}(\mathbf{P}_A, \mathbf{m}', r')$		
11. **if** $(\mathbf{c}_1, \mathbf{c}_2) = (\mathbf{c}'_1, \mathbf{c}'_2)$		
$k_A = \mathrm{KDF}(K', \mathrm{H}(\mathbf{c}'_1, \mathbf{c}'_2))$		
else		
$k_A = \mathrm{KDF}(z, \mathrm{H}(\mathbf{c}'_1, \mathbf{c}'_2))$		

3. The Key Mismatch Attack on Kyber-1024

In this section, we will propose the key mismatch attack on Kyber-1024. According to Table 1, we have $(d_{\mathbf{P}_B}, d_{\mathbf{v}_B}) = (11, 5)$ in Kyber-1024. We will first introduce how to build key mismatch Oracles and then describe the parameters choice of the adversary.

3.1. Key Mismatch Oracles

We build an oracle $\mathcal{O}$ that simulates Alice in Table 2. In Algorithm 1 we describe how the oracle $\mathcal{O}$ works. The input of $\mathcal{O}$ is $(\mathbf{c}_1, \mathbf{c}_2)$ and its output is 1 or 0. To be specific, first, $\mathcal{O}$ chooses a randomly selected 256-bit secret z. By receiving the inputs $(\mathbf{c}_1, \mathbf{c}_2)$, $\mathcal{O}$ will use $\mathbf{c}_1$ and $\mathbf{c}_2$ to calculate $\mathbf{u}_A$ and $\mathbf{v}_A$, separately. Then $\mathcal{O}$ uses these two values to calculate m' and (K', r'). Subsequently, it calls Enc() with inputs $\mathbf{P}_A, m', r'$ to get $(\mathbf{c}'_1, \mathbf{c}'_2)$, if $(\mathbf{c}_1, \mathbf{c}_2) = (\mathbf{c}'_1, \mathbf{c}'_2)$ holds. If $\mathcal{O}$ outputs 1, the shared keys k_A and k_B match, otherwise the shared keys mismatch.

According to Algorithm 1 , since $(K, r) = \mathrm{G}(\mathrm{H}(\mathbf{P}_A), \mathbf{m})$ and $(K', r') = G(\mathrm{H}(\mathbf{P}_A), \mathbf{m}')$, if $\mathbf{m} = \mathbf{m}'$, then $r = r'$. Notice that $(\mathbf{c}_1, \mathbf{c}_2)$ and $(\mathbf{c}'_1, \mathbf{c}'_2)$ are both generated by Enc(), and their inputs are $\mathbf{P}_A, \mathbf{m}', r'$ and $\mathbf{P}_A, \mathbf{m}, r$, respectively. It is easy to verify that if $\mathbf{m} = \mathbf{m}'$, $k_A = k_B$, and the output of $\mathcal{O}$ is 1. To propose our attack, we need to modify the Oracle $\mathcal{O}$ in Algorithm 1. We refer to this Oracle as $\mathcal{O}_m$ and the main process is shown in Algorithm 2. The main difference between the two oracles is that for the received $\mathbf{m}'$, the $\mathcal{O}_m$ directly check whether $\mathbf{m}'$ and $\mathbf{m}$ are equal or not. If so, $\mathcal{O}$ will return 1 otherwise return 0.

Algorithm 1: The Oracle $\mathcal{O}$

Input: $(\mathbf{c}_1, \mathbf{c}_2)$
Output: 1 or 0

$z \xleftarrow{\$} \{0,1\}^{256}$;
$\mathbf{u}_A = \mathbf{Decompress}_q(\mathbf{c}_1, d_{\mathbf{P}_B})$;
$\mathbf{v}_A = \mathbf{Decompress}_q(\mathbf{c}_2, d_{v_B})$;
$\mathbf{m}' = \mathbf{Compress}_q(\mathbf{v}_A - \mathbf{s}_A^T \cdot \mathbf{u}_A, 1)$;
$(K', r') = G(\mathrm{H}(\mathbf{P}_A), \mathbf{m}')$;
$(\mathbf{c}_1', \mathbf{c}_2') = \mathrm{Enc}(\mathbf{P}_A, \mathbf{m}', r')$;
if $(\mathbf{c}_1, \mathbf{c}_2) = (\mathbf{c}_1', \mathbf{c}_2')$ **then**
    $k_A = \mathrm{KDF}(K', \mathrm{H}(\mathbf{c}_1', \mathbf{c}_2'))$;
    Return 1;
else
    $k_A = \mathrm{KDF}(z, \mathrm{H}(\mathbf{c}_1', \mathbf{c}_2'))$;
    Return 0;
end

Algorithm 2: The Oracle $\mathcal{O}_m$

Input: $(\mathbf{c}_1, \mathbf{c}_2), m$
Output: 1 or 0

$\mathbf{u}_A = \mathbf{Decompress}_q(\mathbf{c}_1, d_{\mathbf{P}_B})$;
$\mathbf{v}_A = \mathbf{Decompress}_q(\mathbf{c}_2, d_{v_B})$;
$\mathbf{m}' = \mathbf{Compress}_q(\mathbf{v}_A - \mathbf{s}_A^T \cdot \mathbf{u}_A, 1)$;
if $\mathbf{m}' = \mathbf{m}$ **then**
    Return 1;
else
    Return 0;
end

3.2. Parameter choices of the adversary

We assume that Alice's public key $\mathbf{P}_A$ is reused, and the adversary $\mathcal{A}$'s goal is to recover Alice's secret key $\mathbf{s}_A$ by accessing Oracle $\mathcal{O}_m$ multiple times. Therefore, the most important thing for $\mathcal{A}$ is to properly select the input parameters $(\mathbf{c}_1, \mathbf{c}_2)$ and $\mathbf{m}$ in $\mathcal{O}_m$ in a way such that these parameters can be associated with the secret key $\mathbf{s}_A$ and help $\mathcal{A}$ recover it successfully.

First of all, $\mathcal{A}$ will select a 256-bit $\mathbf{m}$ as $(1, 0, \cdots, 0)$ rather than choose it uniformly at random. In this case, except $\mathbf{m}[0] = 1$ all the other $\mathbf{m}[i] = 0$ $(i = 1, 2, \ldots, 255)$.

Then, $\mathcal{A}$ will directly set $\mathbf{c}_2 = h$, and the range of h is from 0 to 31. This is because in Kyber-1024 $d_{v_B} = 5$ and $\mathbf{c}_2 = \mathbf{Compress}_q(\mathbf{v}_B, d_{v_B}) = \mathbf{Compress}(\mathbf{v}_B, 5)_q = \left\lceil \frac{32}{q} \cdot \mathbf{v}_B \right\rfloor \pmod{32}$.

Suppose that $\mathcal{A}$ wants to recover the k-th position in $\mathbf{s}_A^T$ and k is an integer in [0,255]. $\mathcal{A}$ will choose $\mathbf{P}_B$ carefully rather than calculating it as $\mathbf{P}_B = \mathbf{A}^T \mathbf{s}_B + \mathbf{e}_B$. First he let $\mathbf{P}_B = \mathbf{0}$, then in case $k = 0$, he set $\mathbf{P}_B[0] = \left\lceil \frac{q}{32} \right\rfloor$, otherwise he let $\mathbf{P}_B[256 - k] = -\left\lceil \frac{q}{32} \right\rfloor$. Next, $\mathcal{A}$ will calculate $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$.

In the following we briefly explain why $\mathcal{A}$ sets $\mathbf{P}_B$ in this way. We have already known that when $\mathcal{O}_m$ outputs 1 only if $\mathbf{m}' = \mathbf{m}$, that is $\mathbf{m}'[0] = \mathbf{m}[0] = 1$. Therefore, we only need to consider the first position in $\mathbf{m}'$. By calculating $\mathbf{u}_A = \mathbf{Decompress}_q(\mathbf{c}_1, d_{\mathbf{P}_B})$, $\mathcal{O}_m$ will get $\mathbf{u}_A = \mathbf{P}_B$ and $\mathbf{v}_A[0]$ can be calculated as

$$\mathbf{v}_A[0] = \mathbf{Decompress}_q(\mathbf{c}_2[0], d_{v_B}) = \mathbf{Decompress}_q(h, d_{v_B}) = \left\lceil \frac{q}{2^5} h \right\rfloor = \left\lceil \frac{q}{32} h \right\rfloor.$$

Next we discuss the results of $\mathbf{m}'[0]$ in cases $k = 0$ and $k \neq 0$, respectively.

1) If $k = 0$ and $\mathbf{P}_B[0] = \left\lceil \frac{q}{32} \right\rfloor$, we have

$$\mathbf{m}'[0] = \mathbf{Compress}_q((\mathbf{v}_A - \mathbf{s}_A^T \mathbf{u}_A)[0], 1) = \mathbf{Compress}_q(\mathbf{v}_A[0] - (\mathbf{s}_A^T \mathbf{u}_A)[0], 1) = \left\lceil \frac{2}{q} \left(\mathbf{v}_A[0] - (\mathbf{s}_A^T \mathbf{u}_A)[0] \right) \right\rfloor \bmod 2.$$

(7)

Since $(\mathbf{s}_A^T\mathbf{u}_A)[0] = \mathbf{s}_A^T[0]\mathbf{u}_A[0] = \mathbf{s}_A^T[0]\left\lceil \frac{q}{32} \right\rfloor$,

$$\mathbf{m}'[0] = \left\lceil \frac{2}{q}\left(\left\lceil \frac{q}{32}h \right\rfloor - \mathbf{s}_A^T[0]\left\lceil \frac{q}{32} \right\rfloor\right) \right\rfloor \bmod 2. \tag{8}$$

When h ranges from 0 to 15, according to equation (8) we can deduce that

$$\mathbf{m}'[0] = \left\lceil \frac{2}{q}\left\lceil \frac{q}{32} \right\rfloor (h - \mathbf{s}_A^T[0]) \right\rfloor \bmod 2. \tag{9}$$

2) If $k = 1, 2, \cdots, 255$ and $\mathbf{P}_B[256-k] = -\left\lceil \frac{q}{32} \right\rfloor$, then the constant term in $\mathbf{s}_A^T\mathbf{u}_A$ is

$$\mathbf{s}_A^T[k]x^k\mathbf{u}_A[256-k]x^{256-k} = \mathbf{s}_A^T[k]\left(-\left\lceil \frac{q}{32} \right\rfloor\right)x^{256} = \mathbf{s}_A^T[k]\left\lceil \frac{q}{32} \right\rfloor, \tag{10}$$

the last equation holds since $x^{256} = -1$ in $\mathcal{R}_q$.

Then we have

$$\mathbf{m}'[0] = \left\lceil \frac{2}{q}\left(\left\lceil \frac{q}{32}h \right\rfloor - \mathbf{s}_A^T[k]\left\lceil \frac{q}{32} \right\rfloor\right) \right\rfloor \bmod 2. \tag{11}$$

When h ranges from 0 to 15, according to equation (11) we can conclude that

$$\mathbf{m}'[0] = \left\lceil \frac{2}{q}\left\lceil \frac{q}{32} \right\rfloor (h - \mathbf{s}_A^T[k]) \right\rfloor \bmod 2. \tag{12}$$

Since equations (9) and (12) are the same, in this way the adversary $\mathcal{A}$ successfully associates his deliberately chosen parameters $\mathbf{m}$, $\mathbf{c}_1$ and $\mathbf{c}_2$ with $\mathbf{m}'$ in $\mathcal{O}_m$. Then $\mathcal{A}$ accesses $\mathcal{O}_m$ multiple times and get useful information from the feedbacks to recover $\mathbf{s}_A$. It is worth noting that when h ranges from 16 to 31, equations (9) and (12) do not hold, but this will not affect the recovery of the secret key.

3.3. The Proposed Attack

Algorithm 3: Key-recovery-Kyber-1024

```
Output: s'
Set m = {1, 0, ··· , 0}^256;
for i := 0 to 3 do
    for k := 0 to 255 do
        P_B = 0;
        if k = 0 then
            P_B[0] = ⌈q/32⌋;
        else
            P_B[256 − k] = −⌈q/32⌋;
        c_1 = Compress_q(P_B, d_{P_B});
        for h := 0 to 31 do
            c_2 = 0 except c_2[0] = h;
            t = Oracle_m((c_1, c_2), m);
            if t = 1 then
                break ;
        end
        s'[i ∗ 256 + k] = Check(h);
    end
end
Return s'
```

To launch the attack, the adversary $\mathcal{A}$ deliberately selects the parameters $\mathbf{m}$, $\mathbf{c}_1$ and $\mathbf{c}_2$ as aforementioned. In the following, we will briefly introduce our method to recover the exact value of $\mathbf{s}_A$ in an efficient way. The key mismatch attack includes four steps:

Table 3
The relationship between sk and h in Kyber-1024

sk	0	1	-1	2	-2
h	9	10	8	11	7

Step 1: As we know, in Kyber-1024, $\mathbf{s}_A^T \in R_q^4$ consists of four 256-bit polynomials, and each polynomial is with 256 coefficients. In this step, the adversary $\mathcal{A}$ chooses one of the 4 polynomials as the target. As shown in line 2 of Algorithm 3, i represents the i-th polynomial chosen by $\mathcal{A}$.
Step 2: In this step, the adversary $\mathcal{A}$ also chooses a position in the i-th polynomial of $\mathbf{s}_A^T$, and he tries to recover the value of the secret key corresponding to this position. More specifically, k represents the coefficient in each polynomial, which is shown in line 3 of Algorithm 3.
Step 3: In this step, the adversary $\mathcal{A}$ tries to find a proper h by accessing the Oracle $\mathcal{O}_m$ multiple times to help recover the secret key.

Before accessing the $\mathcal{O}_m$, $\mathcal{A}$ chooses the parameters $\mathbf{m}$, $\mathbf{c}_1$ and $\mathbf{c}_2$ as described previously. Then $\mathcal{A}$ sets $\mathbf{P}_B[0] = \mathbf{0}$ first, if $k = 0$, $\mathbf{P}_B[0] = \left\lceil \frac{q}{32} \right\rfloor$, otherwise $\mathbf{P}_B[256-k] = -\left\lceil \frac{q}{32} \right\rfloor$. Next, we only analyze the case $k = 0$ in detail.

When $\mathcal{A}$ accesses $\mathcal{O}_m$, it will return 0 or 1. The adversary $\mathcal{A}$ starts from setting $h = 0$ to $h = 31$. When $h = 0$, according to equation (9) we have

$$\mathbf{m}'[0] = \left\lceil \frac{2}{q} \left\lceil \frac{q}{32} \right\rfloor (h - \mathbf{s}_A^T[0]) \right\rfloor \bmod 2 = \left\lceil \frac{2}{q} \left\lceil \frac{q}{32} \right\rfloor (-\mathbf{s}_A^T[0]) \right\rfloor \bmod 2. \tag{13}$$

Since $\frac{2}{q} \left\lceil \frac{q}{32} \right\rfloor \approx 0.0624812$, regardless of the value of $\mathbf{s}_A^T[0]$, $\mathbf{m}'[0]$ is always 0. Since $\mathbf{m}[0] = 1$, in the beginning the output of $\mathcal{O}_m$ is always 0. As h increases, the output may become 1 at some point. The adversary $\mathcal{A}$ records the value of h when the output of $\mathcal{O}_m$ changes from 0 to 1. Later, $\mathcal{A}$ uses the recorded h to recover the secret key.

The main process in this step is shown between lines 4 to 15 in the Algorithm 3.
Step 4: In this step, the adversary $\mathcal{A}$ wants to get the exact value of the secret key based on h and Table 3. In Table 3, sk represents the possible value of the coefficient in $\mathbf{s}_A^T[0]$.

With the recorded h in Step 3, $\mathcal{A}$ is able to recover sk corresponding to h in Table 3. The process of looking up the table according to the value of h is the $Check(h)$ in the line 16 of Algorithm 3. For example, if $h = 9$, then the operation of $Check(h)$ is to find its corresponding $sk = 0$ according to Table 3.

In the following we show why we can recover the secret key in this way.

According to equation (9) we have

$$\mathbf{m}'[0] = \left\lceil \frac{2}{q} \left\lceil \frac{q}{32} \right\rfloor (h - \mathbf{s}_A^T[0]) \right\rfloor \bmod 2.$$

We set $f = \frac{2}{q} \left\lceil \frac{q}{32} \right\rfloor (h - \mathbf{s}_A^T[0])$, then the above equation becomes

$$\mathbf{m}'[0] = \lceil f \rfloor \bmod 2.$$

Since $\frac{2}{q} \left\lceil \frac{q}{32} \right\rfloor \approx 0.0624812$, we further have

$$f = 0.0624812(h - \mathbf{s}_A^T[0]) \tag{14}$$

and

$$\mathbf{m}'[0] = \lceil 0.0624812(h - \mathbf{s}_A^T[0]) \rfloor \bmod 2. \tag{15}$$

1. If the element in $\mathbf{s}_A^T$ we want to recover is 0, according to equations (14) and (15), we have

$$f = 0.0624812 \cdot h, \tag{16}$$

and

$$\mathbf{m}'[0] = \lceil f \rfloor \bmod 2 = \lceil 0.0624812 \cdot h \rfloor \bmod 2. \tag{17}$$

At the beginning, $h = 0$, so $f = 0$ and $\mathbf{m}'[0] = 0$. As h increases, the value of $\mathbf{m}'[0]$ turns to 1. From equation (16), we can check that when $h = 9$, $f = 0.0624812 \cdot 9 = 0.56233$ and $\mathbf{m}'[0]$ becomes 1.

Therefore, if $h = 9$, we can recover the coefficient to be 0.

2. Suppose the coefficient in $\mathbf{s}_A^T$ we want to recover is ± 1, ± 2, ± 3 or ± 4. Since they are similar, we only consider the case when the coefficient in $\mathbf{s}_A^T$ is ± 1.

(1) If the coefficient in $\mathbf{s}_A^T$ is 1, according to equations (14) and (15), we have

$$f = 0.0624812(h - 1), \tag{18}$$

and

$$\mathbf{m}'[0] = \lceil f \rfloor \bmod 2 = \lceil 0.0624812(h - 1) \rfloor \bmod 2. \tag{19}$$

When $h = 0$, we can get $f = -0.0624812$ and $\mathbf{m}'[0] = 0$. As h increases, the value of $\mathbf{m}'[0]$ also changes to 1 when $f \geq 0.5$. To be specific, when $h = 10$ in equation (18), we can check that $f = 0.0624812(10 - 1) = 0.0624812 \cdot 9 = 0.56233$ and $\mathbf{m}'[0]$ equals to 1.

So, if the recorded $h = 10$ we can determine that the coefficient is 1.

(2) Suppose the coefficient in $\mathbf{s}_A^T$ is -1, according to equations (14) and (15), we have

$$f = 0.0624812(h + 1), \tag{20}$$

and

$$\mathbf{m}'[0] = \lceil f \rfloor \bmod 2 = \lceil 0.0624812(h + 1) \rfloor \bmod 2. \tag{21}$$

When $h = 0$, we can get $f = 0.0624812$ and $\mathbf{m}'[0] = 0$. As h increases, if $f \geq 0.5$ the value of $\mathbf{m}'[0]$ will change to 1. When $h = 8$ in equation (20), we can have $f = 0.0624812(8 + 1) = 0.0624812 \cdot 9 = 0.56233$ and $\mathbf{m}'[0]$ turns to 1.

Therefore, if $h = 8$, we conduce that the coefficient in the secret key is -1.

4. The Key Mismatch Attack on Kyber-768

In this section, we will analyze the security of the Kyber-768 and introduce the key mismatch attack if the public key $\mathbf{P}_A$ is reused. The main difference between Kyber-768 and Kyber-1024 is that their parameter choice is different. According to Table 1, we know that $(d_{\mathbf{P}_B}, d_{\mathbf{v}_B}) = (10, 4)$ in Kyber-768. Specifically, this will change the range of $\mathbf{c}_1$ and $\mathbf{c}_2$, so the adversary needs to reconsider the choice of these two parameters, and the $Check(h)$ also needs to be changed.

4.1. Parameters choices

We still use the same Oracle O_m in Algorithm 1. At first, $\mathcal{A}$ still selects $\mathbf{m}$ as $\{1, 0, \cdots, 0\}$ rather than choosing it uniformly at random. In this case, except $\mathbf{m}[0] = 1$ all the other $\mathbf{m}[i] = 0$ $(i = 1, 2, \ldots, 255)$.

Since $d_{v_B} = 4$ in Kyber-768, we have $\mathbf{c}_2 = \mathbf{Compress}_q(\mathbf{v}_B, d_{v_B}) = \mathbf{Compress}\ (\mathbf{v}_B, 4)_q = \left\lceil \frac{16}{q} \cdot \mathbf{v}_B \right\rfloor \pmod{16}$. So, $\mathcal{A}$ directly sets $\mathbf{c}_2$ as h, which ranges from 0 to 15.

Similarly, $\mathcal{A}$ directly sets $\mathbf{P}_B = \mathbf{0}$ first, then if $k = 0$, $\mathbf{P}_B[0] = \left\lceil \frac{q}{16} \right\rfloor$, otherwise $\mathbf{P}_B[0] = -\left\lceil \frac{q}{16} \right\rfloor$. The resulted $\mathbf{c}_1$ is $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$. Next, we also only analyze the case $k = 0$.

By calculating $\mathbf{u}_A = \mathbf{Decompress}_q(\mathbf{c}_1, d_{\mathbf{P}_B})$, the $\mathcal{O}_m$ has $\mathbf{u}_A = \mathbf{P}_B$, and the resulted $\mathbf{v}_A[0]$ is

$$\mathbf{v}_A[0] = \mathbf{Decompress}_q(\mathbf{c}_2[0], d_{v_B}) = \mathbf{Decompress}_q(h, d_{v_B}) = \left\lceil \frac{q}{2^4} h \right\rfloor = \left\lceil \frac{q}{16} h \right\rfloor.$$

We further have

$$\begin{aligned}\mathbf{m}'[0] =&\mathbf{Compress}_q((\mathbf{v}_A - \mathbf{s}_A^T\mathbf{u}_A)[0], 1)\\ =&\mathbf{Compress}_q(\mathbf{v}_A[0] - (\mathbf{s}_A^T\mathbf{u}_A)[0], 1)\\ =&\left\lceil \frac{2}{q}\left(\mathbf{v}_A[0] - (\mathbf{s}_A^T\mathbf{u}_A)[0]\right)\right\rfloor \bmod 2\\ =&\left\lceil \frac{2}{q}\left(\left\lceil \frac{q}{16}h\right\rfloor - \left(\mathbf{s}_A^T[0]\left\lceil \frac{q}{16}\right\rfloor\right)\right)\right\rfloor \bmod 2.\end{aligned} \tag{22}$$

When h ranges from 0 to 7, according to equation (22) we can deduce

$$\mathbf{m}'[0] = \left\lceil \frac{2}{q}\left\lceil \frac{q}{16}\right\rfloor (h - \mathbf{s}_A^T[0])\right\rfloor \bmod 2. \tag{23}$$

We also note that when h ranges from 8 to 15, the equation (23) is not true. However, this has little impact on the recovery of the secret key, which we will explain in the next subsection.

4.2. The Proposed Attack

Algorithm 4: Key-recovery-Kyber-768

Output: s'
1 Set $\mathbf{m} = \{1, 0, \cdots, 0\}^{256}$;
2 **for** $i := 0$ *to* 2 **do**
3   **for** $k := 0$ *to* 255 **do**
4     $\mathbf{P}_B = \mathbf{0}$;
5   **if** $k = 0$ **then**
6       $\mathbf{P}_B[0] = \left\lceil \frac{q}{16}\right\rfloor$;
7   **else**
8       $\mathbf{P}_B[256 - k] = -\left\lceil \frac{q}{16}\right\rfloor$;
9     $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$;
10   **for** $h := 0$ *to* 31 **do**
11       $\mathbf{c}_2 = \mathbf{0}$ except $\mathbf{c}_2[0] = h$;
12       $t = Oracle_m((\mathbf{c}_1, \mathbf{c}_2), m)$;
13   **if** $t = 1$ **then**
14   break ;
15   **end**
16     $s'[i * \text{Kyber\_N} + k] = Check(h)$;
17   **end**
18 **end**
19 **Return** $\mathbf{s}'$

The key mismatch attack on Kyber-768 is similar to that on Kyber-1024, which also consists of four steps. Therefore, we only present **Step 3** and **Step 4**.

Step 3: In this step, the adversary $\mathcal{A}$ record the value of h by accessing the Oracle $\mathcal{O}_m$ multiple times to help him recover the secret key.

Before $\mathcal{A}$ accesses $\mathcal{O}_m$ he chose the parameters $\mathbf{m}, \mathbf{c}_1$ and $\mathbf{c}_2$ as previously described. That is, if $k = 0$, $\mathcal{A}$ sets $\mathbf{P}_B[0] = \left\lceil \frac{q}{16}\right\rfloor$, otherwise $\mathcal{A}$ sets $\mathbf{P}_B[256 - k] = -\left\lceil \frac{q}{16}\right\rfloor$.

The adversary $\mathcal{A}$ also starts from $h = 0$, according to equation (23), it holds that

$$\mathbf{m}'[0] = \left\lceil \frac{2}{q}\left\lceil \frac{q}{16}\right\rfloor (h - \mathbf{s}_A^T[0])\right\rfloor \bmod 2 = \left\lceil \frac{2}{q}\left\lceil \frac{q}{16}\right\rfloor (-\mathbf{s}_A^T[0])\right\rfloor \bmod 2. \tag{24}$$

Since $\frac{2}{q}\left\lceil \frac{q}{16}\right\rfloor \approx 0.1249625$, regardless the value of $\mathbf{s}_A^T[0]$, $\mathbf{m}'[0]$ is always 0. Similarly, $\mathcal{A}$ increases h from 0 to 15, and record the value of h when the output of $\mathcal{O}_m$ changes from 0 to 1.

The main process of this step is shown between line 4 to line 15 in Algorithm 4.

Step 4: In this step, the adversary $\mathcal{A}$ wants to recover the coefficient of the secret key based on h and Table 4.

In Table 4, we demonstrate the relationship between sk and h. By looking up the Table 4, we can efficiently recover all the coefficients in the secret key.

Table 4
The relationship between sk and h in Kyber-768

sk	0	1	-1	2	-2
h	5	6	4	7	3

5. The Key Mismatch Attack on Kyber-512

In this section, we assess the security of the Kyber-512 under the key mismatch attack. According to Table 1, we know $(d_{\mathbf{P}_B}, d_{\mathbf{v}_B}) = (10, 3)$. The Oracle we use in this section is still O_m in Algorithm 1. Since the theoretical analysis is similar as that in Kyber-768 and Kyber-1024, the rest of this section only introduces $\mathcal{A}$'s choice of parameters $\mathbf{P}_B$ and $\mathbf{c}_2$, as well as the $check(h)$ in Algorithm 5.

5.1. Parameters choices

Similarly, $\mathcal{A}$ first selects a 256-bit $\mathbf{m}$ as $\{1, 0, \cdots, 0\}$. Then $\mathcal{A}$ directly sets $\mathbf{c}_2 = h$. Here the range of h is from 0 to 7, since $d_{\mathbf{v}_B} = 3$ in Kyber-512. Subsequently, $\mathcal{A}$ directly set $\mathbf{P}_B = \mathbf{0}$ first, then if $k = 0$, $\mathbf{P}_B[0] = \left\lceil \frac{q}{8} \right\rfloor$ and $\mathbf{P}_B[256-k] = -\left\lceil \frac{q}{8} \right\rfloor$ otherwise. The resulted $\mathbf{c}_1$ can be calculated using $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$.

With the above parameters, $\mathcal{O}_m$ needs to calculate $\mathbf{u}_A = \mathbf{Decompress}_q(\mathbf{c}_1, d_{\mathbf{P}_B})$ and the result is $\mathbf{u}_A = \mathbf{P}_B$. Next, $\mathcal{O}_m$ goes on computing

$$\mathbf{v}_A[0] = \mathbf{Decompress}_q(\mathbf{c}_2[0], d_{v_B}) = \mathbf{Decompress}_q(h, d_{v_B}) = \left\lceil \frac{q}{2^3} h \right\rfloor = \left\lceil \frac{q}{8} h \right\rfloor.$$

And this will result in

$$\begin{aligned}
\mathbf{m}'[0] =& \mathbf{Compress}_q((\mathbf{v}_A - \mathbf{s}_A^T \mathbf{u}_A)[0], 1) \\
=& \mathbf{Compress}_q(\mathbf{v}_A[0] - (\mathbf{s}_A^T \mathbf{u}_A)[0], 1) \\
=& \left\lceil \frac{2}{q} \left(\mathbf{v}_A[0] - (\mathbf{s}_A^T \mathbf{u}_A)[0] \right) \right\rfloor \bmod 2 \\
=& \left\lceil \frac{2}{q} \left(\left\lceil \frac{q}{8} h \right\rfloor - \mathbf{s}_A^T[0] \left\lceil \frac{q}{8} \right\rfloor \right) \right\rfloor \bmod 2.
\end{aligned} \tag{25}$$

When h changes from 0 to 7, according to equation (25) we can deduce that

$$\mathbf{m}'[0] = \left\lceil \frac{2}{q} \left\lceil \frac{q}{8} \right\rfloor (h - \mathbf{s}_A^T[0]) \right\rfloor \bmod 2. \tag{26}$$

We could note that when h changes from 4 to 7, equation (26) does not hold any more, which could affect the recovery of the secret key.

5.2. The Proposed Attack

The same as before, the key mismatch attack on Kyber-512 consists of four steps. We only briefly describe the changes in **Step 3** and **Step 4**, and explain how to deal with values of $Check(h)$ in the line 16 of Algorithm 5.

Step 3: In this step, the adversary $\mathcal{A}$ wants to find a value h by accessing the Oracle $\mathcal{O}_m$ multiple times to help him recover the secret key, and he will choose the parameters $m, \mathbf{c}_1$ and $\mathbf{c}_2$ as described previously.

When $\mathcal{A}$ accesses $\mathcal{O}_m$ with $h = 0$, from (26) we further have

$$\mathbf{m}'[0] = \left\lceil \frac{2}{q} \left\lceil \frac{q}{8} \right\rfloor (h - \mathbf{s}_A^T[0]) \right\rfloor \bmod 2 = \left\lceil \frac{2}{q} \left\lceil \frac{q}{8} \right\rfloor (-\mathbf{s}_A^T[0]) \right\rfloor \bmod 2 = 0 \tag{27}$$

since $\frac{2}{q} \left\lceil \frac{q}{8} \right\rfloor \approx 0.249924$. Therefore, when the output of $\mathcal{O}_m$ becomes 1, $\mathcal{A}$ uses the recorded h to recover the coefficient.

The main process in this step is shown in the Algorithm 5.

Algorithm 5: Key-recovery-Kyber-512

Output: s'
1 Set $\mathbf{m} = \{1, 0, \cdots, 0\}^{256}$;
2 **for** $i := 0$ *to* 1 **do**
3   **for** $k := 0$ *to* 255 **do**
4         $\mathbf{P}_B = \mathbf{0}$;
5   **if** $k = 0$ **then**
6             $\mathbf{P}_B[0] = \left\lceil \frac{q}{8} \right\rfloor$;
7   **else**
8             $\mathbf{P}_B[256 - k] = -\left\lceil \frac{q}{8} \right\rfloor$;
9         $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$;
10   **for** $h := 0$ *to* 7 **do**
11             $\mathbf{c}_2 = \mathbf{0}$ except $\mathbf{c}_2[0] = h$;
12             $t = Oracle_m((\mathbf{c}_1, \mathbf{c}_2), m)$;
13   **if** $t = 1$ **then**
14   break ;
15   **end**
16   **if** $h = 4$ **then**
17             $\mathbf{P}_B = \mathbf{0}$;
18   **if** $k = 0$ **then**
19                 $\mathbf{P}_B[0] = -\left\lceil \frac{q}{8} \right\rfloor$;
20   **else**
21                 $\mathbf{P}_B[256 - k] = \left\lceil \frac{q}{8} \right\rfloor$;
22             $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$;
23   **for** $h_1 := 1$ *to* 2 **do**
24                 $\mathbf{c}_2 = \mathbf{0}$ except $\mathbf{c}_2[0] = h_1$;
25                 $t = Oracle_m((\mathbf{c}_1, \mathbf{c}_2), m)$;
26   **if** $t = 1$ **then**
27   break ;
28   **end**
29   **if** $h_1 = 1$ **then**
30                 $s'[i * \text{Kyber\_N} + k] = 2$;
31   **else**
32                 $s'[i * \text{Kyber\_N} + k] = 1$;
33   **else**
34             $s'[i * \text{Kyber\_N} + k] = Check(h)$;
35   **end**
36 **end**

Table 5
The relationship between sk and h, h_1 on Kyber-512

sk	0	-1	-2	1	2
h	3	2	1	4	4
h_1				2	1

Step 4: In this step, the adversary $\mathcal{A}$ also recovers the coefficients by simply looking up h in Table 5.

But there are special cases we have to pay attention to. When the integer h is in the interval $[4, 7]$, the equation (26) does not hold. So, we need additional information to further decide the exact sk.

We first let $\mathbf{P}_B[0] = \left\lceil -\frac{q}{8} \right\rfloor$, if $k = 0$, and $\mathbf{P}_B[256 - k] = \left\lceil \frac{q}{8} \right\rfloor$ otherwise. Then we repeat the processes from lines 9 to 15 in Algorithm 5 until the value of h_1 is obtained. And next, we can determine the exact value of sk based on h and h_1 in Table 5. For example, if $h = 4$ and $h_1 = 2$, we can determine $sk = 1$, otherwise if $h_1 = 1$, the value of sk is 2.

In sum, in this step, if $h = 1, 2$ or 3, we can directly recover sk according to Table 5. Otherwise, we will recover the value of sk based on h and h_1 in Table 5.

6. The Improved Key Mismatch Attack

In this Section, we propose our improved attack to reduce the number of queries and improve efficiency in the previous attacks.

Table 6
The relationship between sk & h & O_m in Kyber-1024

sk	0		-1			-2				1		2		
h	9	8	9	8	7	9	8	7	6	9	10	9	10	11
O_m	1	0	1		0	1			0	0	1	0		1

In the previous sections we introduce the key mismatch attacks on three security levels of Kyber. Our key observation is that in the above three attacks all the hs start from $h = 0$, which is not efficient. By analyzing Tables 3, 4 and 5, we find that h is fixed within a certain range. Specifically, in Kyber-1024 the recorded h ranges from 7 to 11, while in Kyber-768 and Kyber-512, h ranges from 3 to 7 and 1 to 5, respectively.

Therefore, in the improved attack we limit the range of h, that is, h starts from 7 in Kyber-1024 and the range of h is $[7, 11]$. We only need to modify the line 10 in Algorithm 3. Similarly, in Kyber-768, we let h in Algorithm 4 in the range $[3, 7]$. And the range of h in Kyber-512 is $[1, 5]$. We refer to the improved attack as the $Improved\ attack\ v_1$.

Can we further reduce the needed queries? In the following, we introduce our $Improved\ attack\ v_2$. Our key idea here is to use the information about the distribution of the coefficients in the secret key. As we know, the coefficients in the secret key obey the centered binomial distribution. That is, the probability of the occurrence of 0 in sk is the largest, followed by $\pm 1, \pm 2$. So when recovering sk, we first determine if it is 0, if not we go on verifying whether it is ± 1 or ± 2. The specific key mismatch attacks are shown in Algorithms 6, 7 and 8, respectively.

6.1. The Improved attack v_2 on Kyber-1024

In this subsection, we take $sk = 0$ and $sk = -1$ as examples to explain our idea. The specific details of this attack are shown in Table 6 and Algorithm 6.

According to Table 3 we can see that, if $sk = 0$, when h increase to 9, the output of the Oracle becomes 1, but if $h = 8$, Oracle's output is still 0. Therefore, we can determine $sk = 0$ using only 2 queries:

Algorithm 6: Key-recovery-Kyber-1024-v2

```
Output: s'
Set m = {1, 0, ··· , 0}^256;
for i := 0 to 3 do
    for k := 0 to 255 do
        P_B = 0;
        if k = 0 then
            P_B[0] = ⌈q/32⌋ ;
        else
            P_B[256 − k] = −⌈q/32⌋;
        c_1 = Compress_q(P_B, d_{P_B}); c_2 = 0
        if c_2[0] = 9 and O_m((c_1, c_2), m) = 1 then
            if c_2[0] = 8 and O_m((c_1, c_2), m) = 0 then
                s'[i ∗ 256 + k] = 0 ;
            else if c_2[0] = 7 and O_m((c_1, c_2), m) = 0 then
                s'[i ∗ 256 + k] = −1 ;
            else
                s'[i ∗ 256 + k] = −2 ;
        else if c_2[0] = 9 and O_m((c_1, c_2), m) = 0 then
            if c_2[0] = 10 and O_m((c_1, c_2), m) = 1 then
                s'[i ∗ 256 + k] = 1 ;
            else
                s'[i ∗ 256 + k] = 2 ;
        end
    end
end
```

1. We set $h = 9$ and the output of Oracle is 1;
2. If 1) holds, we let $h = 8$ and Oracle returns 0.

Table 7
The relationship between sk & h & O_m in Kyber-768

<table>
<tr><td>sk</td><td colspan="2">0</td><td colspan="3">-1</td><td colspan="4">-2</td><td colspan="2">1</td><td colspan="3">2</td></tr>
<tr><td>h</td><td>5</td><td>4</td><td>5</td><td>4</td><td>3</td><td>5</td><td>4</td><td>3</td><td>2</td><td>5</td><td>6</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>O_m</td><td>1</td><td>0</td><td colspan="2">1</td><td>0</td><td colspan="3">1</td><td>0</td><td>0</td><td>1</td><td colspan="2">0</td><td>1</td></tr>
</table>

The purpose of the second query is to confirm that the output of the Oracle becomes 1 when $h = 9$ instead of $h = 8$.

Similarly, if we want to determine $sk = -1$, then only 3 queries are needed:

1. We set $h = 9$ and the output of the Oracle is 1;
2. If 1) holds, then we set $h = 8$ and the output of the Oracle is still 1;
3. If 2) holds, then we go on setting $h = 7$ and the Oracle returns 0.

6.2. The Improved attack v_2 on Kyber-768

The improved key mismatch attack v_2 on Kyber-768 is the same as that in Kyber-1024, except the relationship between h and sk. The specific sk recovery process is shown in Appendix 7. And the relationship between sk and h, O_m is given in Table 7.

6.3. The Improved attack v_2 on Kyber-512

The improved attack v_2 on Kyber-512, in which we need to use another h_1 to collect enough information, differs from the attacks against Kyber-1024 and Kyber-768. Specifically, in the improved attack v_2 on Kyber-512, we can recover $sk = 0, -1$ and -2 with h, but $sk = 1$ and 2 with h and h_1. If the sk we want to recover is -2, we can determine it through 4 queries:

1. We set $h = 3$ and the output of the Oracle is 1;
2. If 1) holds, then we set $h = 2$ and the output of the Oracle is 1;
3. If 2) holds, then we let $h = 1$ and Oracle will return 1;
4. If 3) holds, we will set $h = 0$, the Oracle returns 0.

Similarly, we can determine $sk = 2$ as follows:

1. We set $h = 3$ and the output of the Oracle is 0;
2. If 1) holds, then we set $h = 4$ and the output of the Oracle becomes 1;
3. If 2) holds, we reset $\mathbf{P}_B$ and let $h_1 = 1$, the Oracle returns 1.

The relationship between sk and h, h_1 is shown in Table 8 and the specific sk recovery process is shown in Appendix B.

7. Experiments

In our experiments, we use a 2.7 GHz Intel Core i7 processor with an 8 GB RAM and run it in the macOS High Sierra with version 10.13.6. All experiments are in C and compiled with gcc version 4.2.1. We implement the proposed key mismatch attacks on Kyber's source code submitted to NIST, and then compile it with the same makefile in the source code. We first analyzed Kyber's source code for three different security levels that are submitted to NIST, namely the Kyber-512, Kyber-768 and Kyber-1024. We implemented three sets of experiments. The first set is the original attack described in Sections 3, 4, and 5. The second is the *Improved attack* v_1 introduced in Section 6. The difference between the Original Attack and the *Improved attack* v_1 is that in the Original Attack h starts from $h = 0$.

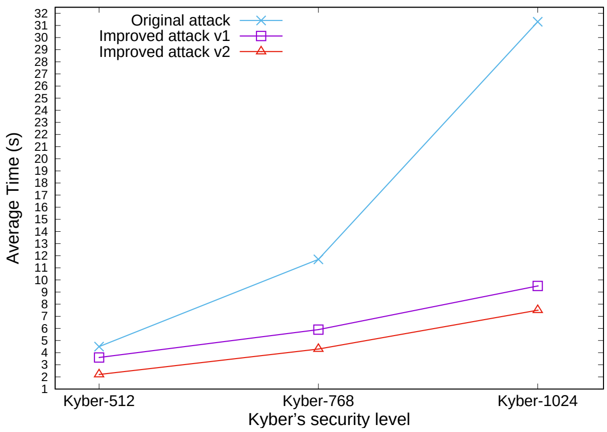
Table 8
The relationship between sk & h & h_1& O_m in Kyber-512

sk	0		-1			-2				1		2	
h	3	2	3	2	1	3	2	1	0	3	4	3	4
O_m	1	0	1		0	1			0	0	1	0	1
h_1										1		1	
O_m										0		1	

Table 9
Average Queries & Time

	Kyber-512			Kyber-768			Kyber-1024		
	Original Attack	Improved Attack v_1	Improved Attack v_2	Original Attack	Improved Attack v_1	Improved Attack v_2	Original Attack	Improved Attack v_1	Improved Attack v_2
Average Queries	$2,596$	$2,086$	$1,401$	$4,654$	$2,351$	$1,855$	$10,303$	$3,132$	$2,475$
Average Time (s)	4.5	3.61	2.2	11.7	5.88	4.3	31.3	9.51	7.5

Figure 1: The Average Time (s)



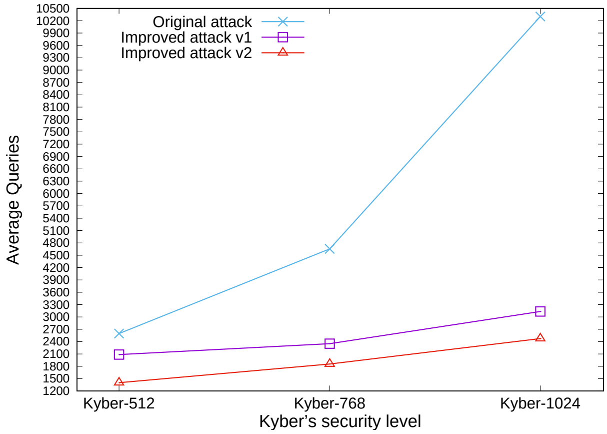
The third set corresponds to Algorithms 6, 7 and 8 in the *Improved attack* v_2 described in Section 6. In this attack, we redesign the recovery strategy, applying information on distribution probability of coefficients in the secret key.

In our experiments, we first use Kyber's source code to generate 100 secret keys. We evaluate the performance of these three attacks on both queries and time. Whenever the adversary accesses Oracle once, the number of quires increases by one. The number of average queries is reported by counting the number of queries the adversary needs to recover a whole secret key. For example, in the attack on Kyber-1024, we count the number of queries the adversary needs to recover 1024 coefficients in each secret key. Each attack is repeated 100 times and then the number is averaged.

The reported time in the experiments starts from the initial generation of the secret key $\mathbf{s}_A$ by Alice, to the recovery of the whole secret key by accessing the Oracle multiple times. The average time refers to recording the whole time required for the adversary to recovering a complete $\mathbf{s}_A$ 100 times and taking an average. The comparison of time spending in the three sets of experiments is given in Figure 1. We can conclude that the *Improved attack* v_2 is the fastest, which is nearly twice as fast as that in the Original Attack.

The comparison of the queries of the three sets of experiments is shown in Figure 2. We can see that the average number of queries is significantly reduced in *Improved attack* v_1 and *Improved attack* v_2, compared with that in the

Figure 2: The Average Number of Queries



Original Attack. For example, in the $Improved\ attack\ v_1$, the average number of queries on Kyber-1024 is reduced to 30% of the Original Attack. Comparing the results of the $Improved\ attack\ v_2$ and the Original Attack, the average number of queries in Kyber-512 is reduced by a half, and the queries in Kyber-768 and Kyber-1024 occupy only 40% and a quarter of that in the Original Attack, respectively.

In Table 9, we illustrate the queries and time needed in recovering the whole key. From the experiments, an interesting observation is that as Kyber's security level increases, the key mismatch attack becomes even simpler. Specifically, in Kyber-512 on average we recover each coefficient in 2.7 queries, while in Kyber-1024 and 768, we only need 2.4 queries. The reason is that in Kyber-1024 and Kyber-768, all the coefficients can be recovered directly using h. But in Kyber-512 the key mismatch attacks require calculating both h and h_1, and then recover sk with the recorded h and h_1.

In [26], we propose our key mismatch attacks on another NIST second-round candidate NewHope. Comparing these two attacks, we conclude that it is much easier to use key mismatch attacks to break Kyber than that on NewHope. To be specific, in Kyber-1024, we need an average of $2,475$ queries to recover all the coefficients in a key. While in NewHope-1024 we need $882,794$ queries. The number of queries for NewHope-1024 is 356.7 times larger than that of Kyber-1024. From another perspective, in Kyber-1024 on average we need 2.4 queries to recover each coefficient, while in Newhope-1024, 862.1 queries are needed.

The main reason of the differences of key mismatch attacks on NewHope and Kyber comes from their different design structures. First, in Kyber only Compress and Decompress functions are used to process coefficients one by one. But in NewHope, additional Encode and Decode functions are used, which operate on a quadruple of coefficients at a time, and this makes the attack on NewHope particularly complicated. Second, the ranges of coefficients in Kyber are much smaller than that in NewHope. To be specific, in NewHope the parameter η of the centered binomial distribution $\mathcal{B}_\eta$ is set as 8, while in Kyber $\eta = 2$. Equivalently, the coefficients of the secret key in NewHope ranges from -8 to 8, while in Kyber the range is from -2 to 2. The smaller ranges of Kyber result in less queries to launch the attack. In addition, while recovering the secret key in NewHope, we need to find 50 favorable cases, which also incurs a lot of additional queries.

8. Conclusion

In this paper, we have proposed key mismatch attacks on the Kyber KEM with three security levels. We need to emphasize that the CCA2-secure Kyber KEM is still secure, but if we implement it in a wrong way, here comes the attacks. To show the efficiency of the proposed attacks, we have implemented them using Kyber's source code. From the experiments, an interesting observation is that as Kyber's security level increases, the key mismatch attack becomes even simpler. We hope that our proposed attack may help in assessing the security of Kyber, reminding us not to implement Kyber in the wrong way.

Acknowledgments

The work presented in this paper was supported in part by the National Natural Science Foundation of China under Grant no. 61672029. Jintai Ding would like to thank the partial support of USA Air Force and NSF.

References

[1] Ajtai, M., 1996. Generating hard instances of lattice problems, in: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, ACM. pp. 99–108.

[2] Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Kelsey, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., et al., 2020. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. US Department of Commerce, National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf. Last accessed 23 July 2020.

[3] Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Liu, Y.K., Miller, C., Moody, D., Peralta, R., et al., 2019. Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process. US Department of Commerce, National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf. Last accessed 26 February 2019.

[4] Alkim, E., Avanzi, R., Bos, J., Ducas, L., de la Piedra, A., Pöppelmann, T., Schwabe, P., Stebila, D., 2019a. Newhope: Algorithm specification and supporting documentation - version 1.03. https://newhopecrypto.org/data/NewHope_2019_07_10.pdf. Last accessed 12 July 2019.

[5] Alkim, E., Bos, J., Ducas, L., Longa, P., Mironov, I., Naehrig, M., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D., 2019b. Frodokem learning with errors key encapsulation: Algorithm specification and supporting documentation. https://frodokem.org/files/FrodoKEM-specification-20190702.pdf. Last accessed 12 July 2019.

[6] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D., 2017. Crystals-ckyber: Algorithm specification and supporting documentation. Submission to the NIST post-quantum project (2017) https://pq-crystals.org/kyber. Last accessed 24 June 2019.

[7] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D., 2019. CrystalsÅąckyber: Algorithm specification and supporting documentation (version 2.0). Submission to the NIST post-quantum project (2019). https://pq-crystals.org/kyber. Last accessed 24 June 2019.

[8] Băetu, C., Durak, F.B., Huguenin-Dumittan, L., Talayhan, A., Vaudenay, S., 2019. Misuse attacks on post-quantum cryptosystems, in: Ishai, Y., Rijmen, V. (Eds.), Advances in Cryptology – EUROCRYPT 2019, Springer International Publishing, Cham. pp. 747–776.

[9] Bauer, A., Gilbert, H., Renault, G., Rossi, M., 2019. Assessment of the key-reuse resilience of newhope, in: CryptographersÂąÂŕ Track at the RSA Conference, Springer. pp. 272–292.

[10] Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D., 2018. Crystals-kyber: a cca-secure module-lattice-based kem, in: 2018 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE. pp. 353–367. https://ieeexplore.ieee.org/document/8406610. Last accessed 29 August 2020.

[11] D'Anvers, J.P., Guo, Q., Johansson, T., Nilsson, A., Vercauteren, F., Verbauwhede, I., 2019. Decryption failure attacks on ind-cca secure lattice-based schemes, in: IACR International Workshop on Public Key Cryptography, Springer. pp. 565–598.

[12] D'Anvers, J.P., Rossi, M., Virdia, F., 2020. (one) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer. pp. 3–33.

[13] Ding, J., Alsayigh, S., Saraswathy, R., Fluhrer, S., Lin, X., 2017. Leakage of signal function with reused keys in rlwe key exchange, in: 2017 IEEE International Conference on Communications (ICC), IEEE. pp. 1–6.

[14] Ding, J., Fluhrer, S., Rv, S., 2018. Complete attack on rlwe key exchange with reused keys, without signal leakage, in: Australasian Conference on Information Security and Privacy, Springer. pp. 467–486.

[15] Ding, J., Xie, X., Lin, X., 2012. A simple provably secure key exchange scheme based on the learning with errors problem. IACR Cryptology EPrint Archive https://eprint.iacr.org/2012/688.pdf. Last accessed 26 February 2019.

[16] Fluhrer, S.R., 2016. Cryptanalysis of ring-lwe based key exchange with key share reuse. IACR Cryptology ePrint Archive http://eprint.iacr.org/2016/085. Last accessed 18 February 2019.

[17] Google, 2016. Experimenting with post-quantum cryptography. https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html. Last accessed 29 August 2020.

[18] Gyongyosi, L., Imre, S., 2019. A survey on quantum computing technology. Computer Science Review 31, 51–71.

[19] Huguenin-Dumittan, L., Vaudenay, S., 2020. Classical misuse attacks on nist round 2 pqc: The power of rank-based schemes. Cryptology ePrint Archive, Report 2020/409. https://eprint.iacr.org/2020/409.

[20] IBM, 2019. WorldâĂŹs first quantum computing safe tape drive. https://www.ibm.com/blogs/research/2019/08/crystals/. Last accessed 29 August 2020.

[21] Kirkwood, D., Lackey, B.C., McVey, J., Motley, M., Solinas, J.A., Tuller, D., 2015. Failure is not an option: standardization issues for post-quantum key agreement, in: Workshop on Cybersecurity in a Post-Quantum World. https://csrc.nist.gov/csrc/media/events/workshop-on-cybersecurity-in-a-post-quantum-world/documents/presentations/session7-motley-mark.pdf. Last accessed 12 July 2019.

[22] Langlois, A., Stehlé, D., 2015. Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography 75, 565–599.

[23] Lyubashevsky, V., Peikert, C., Regev, O., 2010. On ideal lattices and learning with errors over rings, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer. pp. 1–23.

[24] Moody, D., 2016. Post Quantum Cryptography Standardization: Announcement and outline of NIST's Call for Submissions. PQCrypto 2016,

Fukuoka, Japan. https://csrc.nist.gov/Presentations/2016/Announcement-and-outline-of-NIST-s-Call-for-Submis. Last accessed 11 July 2019.

[25] Moody, D., 2019. Round 2 of the nist pqc “competition" - what was nist thinking?

[26] Qin, Y., Cheng, C., Ding, J., 2019. A complete and optimized key mismatch attack on nist candidate newhope., in: Computer Security – ESORICS 2019, Springer International Publishing, Cham. pp. 504–520. doi:doi: 10.1007/978-3-030-29962-0_24.

[27] Regev, O., 2005. On lattices, learning with errors, random linear codes, and cryptography, in: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing, ACM. pp. 84–93. URL: http://doi.acm.org/10.1145/1060590.1060603.

[28] Rescorla, E., 2018. The transport layer security (TLS) protocol version 1.3. Technical Report. IETF. http://www.rfc-editor.org/info/rfc8446. Last accessed 26 February 2019.

A. The improved attack v_2 on Kyber-768

Algorithm 7: Key-recovery-Kyber-768-v2

Output: s'

Set $m = \{1, 0, \cdots, 0\}^{256}$;
for $i := 0$ *to* 2 **do**
    for $k := 0$ *to* 255 **do**
        $\mathbf{P}_B = \mathbf{0}$;
        if $k = 0$ **then**
            $\mathbf{P}_B[0] = \left\lceil \frac{q}{16} \right\rfloor$;
        else
            $\mathbf{P}_B[256 - k] = -\left\lceil \frac{q}{16} \right\rfloor$;
        $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$;
        $\mathbf{c}_2 = \mathbf{0}$;
        if $\mathbf{c}_2[0] = 5$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 1$ **then**
            if $\mathbf{c}_2[0] = 4$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 0$ **then**
                $s'[i * 256 + k] = 0$;
            else if $\mathbf{c}_2[0] = 3$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 0$ **then**
                $s'[i * 256 + k] = -1$;
            else
                $s'[i * 256 + k] = -2$;
        else if $\mathbf{c}_2[0] = 5$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 0$ **then**
            if $\mathbf{c}_2[0] = 6$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 1$ **then**
                $s'[i * 256 + k] = 1$;
            else
                $s'[i * 256 + k] = 2$;
        end
    end
end

B. The improved attack v_2 on Kyber-512

Algorithm 8: Key-recovery-Kyber-512-v2

Output: s'

Set $m = \{1, 0, \cdots, 0\}^{256}$;
for $i := 0$ *to* 1 **do**
    for $k := 0$ *to* 255 **do**
        $\mathbf{P}_B = \mathbf{0}$;
        if $k = 0$ **then**
            $\mathbf{P}_B[0] = \left\lceil \frac{q}{8} \right\rfloor$;
        else
            $\mathbf{P}_B[256 - k] = -\left\lceil \frac{q}{8} \right\rfloor$;
        $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$;
        $\mathbf{c}_2 = \mathbf{0}$;
        if $\mathbf{c}_2[0] = 3$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 1$ **then**
            if $\mathbf{c}_2[0] = 2$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 0$ **then**
                $s'[i * 256 + k] = 0$;
            else if $\mathbf{c}_2[0] = 1$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 0$ **then**
                $s'[i * 256 + k] = -1$;
            else
                $s'[i * 256 + k] = -2$;
        else if $\mathbf{c}_2[0] = 3$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 0$ **then**
            if $\mathbf{c}_2[0] = 4$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 1$ **then**
                if $k = 0$ **then**
                    $\mathbf{P}_B[0] = -\left\lceil \frac{q}{8} \right\rfloor$;
                else
                    $\mathbf{P}_B[256 - k] = \left\lceil \frac{q}{8} \right\rfloor$;
            if $\mathbf{c}_2[0] = 1$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 0$ **then**
                    $s'[i * 256 + k] = 1$;
            else if $\mathbf{c}_2[0] = 1$ *and* $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 1$ **then**
                    $s'[i * 256 + k] = 2$;
        end
    end
end