# An Efficient Key Mismatch Attack on the NIST Second Round Candidate Kyber

Yue Qin, Chi Cheng, *Member, IEEE*, and Jintai Ding

***Abstract*—Kyber is a KEM based their security on the Modular Learning with Errors problem and was selected in the second round of NIST Post-quantum standardization process. Before we put Kyber into practical application, it is very important to assess its security in hard practical conditions especially when the Fujisaki-Okamoto transformations are neglected. 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 show that by accessing the oracle multiple times, the adversary is able to recover the coefficients in the secret key. Furthermore, we propose two strategies to reduce the queries and time in recovering the secret key. It turns out that it is actually much easier to use key mismatch attacks to break Kyber than NewHope, another NIST second round candidate, due to their different design structures. 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 powerful 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.**

***Index Terms*—Modular-LWE, Kyber, key reuse attacks, security analysis;**

## I. Introduction

Public key cryptographic algorithms such as public key encryption (PKE), signatures and key exchange schemes have played fundamental roles in securing communications over the Internet. However, these widely used building blocks base their security on difficult problems like big integer factorization and discrete logarithms problems, which could be broken if there come the quantum computers. With recently reported rapid progresses in quantum technology [1], the transition from the currently used public key cryptographic blocks to their post-quantum counterparts has become urgent.

Since 2016 [2], the National Institute of Standards and Technology (NIST) has began collecting post-quantum cryptographic algorithms from all over the world. As pointed out in their 2016 report (NISTIR 8015 [3]), the goal of post-quantum cryptography is to establish cryptographic systems that are secure against both quantum and classical computers, integrating with existing communication protocols and networks. On December 20, 2017, 69 algorithms were accepted as the first-round candidates, which also marked the beginning of the NIST post-quantum cryptography standardization process. On January 30, 2019, in the report (NISTIR 8240 [4]) NIST announced the second round of candidates. In the past August NIST hosted the second NIST PQC standardization conference.

The Key Encapsulation Mechanisms (KEMs), which allow the communicating parties to obtain a shared key by using a public key mechanism, have gained significant attention in the competition. Among these 26 PKE/KEM candidates in the second round of the NIST standardization process, 12 of them are based on lattice [4]. The advantages of lattice-based constructions include strong security guarantees based on worst-case hardness, as well as its high efficiency in implementations. The main difficulty in constructing a lattice-based Diffie-Hellman-like key exchange protocol is how to effectively reconcile errors in order to negotiate a consistent shared key. In [5], Ding, Xie, and Lin first proposed a "robust extractor" to reconcile the errors, and one of the participants needs to send an additional signal message to the other party, so that the two participants can agree on a shared key. Ding, Xie, and Lin's schemes base their security on the Learning with Errors (LWE) problem or Ring Learning with Errors (R-LWE) problem. The latter can be seen as the polynomial version of the former. On NIST's second-round list, there are LWE-based KEM Frodo, and R-LWE-based KEM NewHope. Kyber is another candidate, whose security is based on the Module Learning with Errors (M-LWE) problem. The M-LWE problem can be viewed as the combination of the LWE and R-LWE problems.

However, before we put these lattice-based KEMs into practical application, it is of great importance to assess their security in hard practical conditions. From history we know that it happens frequently that the important secure transformations such as the Fujisaki-Okamoto transformation are neglected in the implementation, either for the reason of efficiency or being lazy to avoid the additional work. Therefore, in these cases we should keep in mind how to deal with the key reuse attacks against them. In PQCrypto 2019 [6], NIST defined the safety targets of the PKE/KEM schemes, which require that these schemes need to thwart the key reuse attacks. Reusing the key may cause the leakage of private keys, but in some situations the key must be reused, such as that in TLS 1.3 [7]. The 0-RTT (Zero Round Trip Time) mode is a major innovation in TLS 1.3 that saves one round trip time for connections during the handshake phase. This requires both the client and the server to maintain a long-term public key.

In 2015, Kirkwood et al. from the US National Security A-

Yue Qin and Chi Cheng are with the School of Computer Science, China University of Geosciences, Wuhan, 430074, China. E-mail: {chengchi,qy52hz}@cug.edu.cn.

Jintai Ding is with Department of Mathematical Sciences, University of Cincinnati, Cincinnati, 45219, USA. E-mail: jintai.ding@gmail.com.

gency (NSA) announced that there may exist key reuse attacks against the lattice-based post-quantum key exchange protocol, without giving any details [8]. Later, Fluhrer showed in [9] that if the public key of the R-LWE-based key exchange is reused, then this protocol could be broken. In [10], Ding, Alsayigh and Saraswathy first applied the signal leakage attack to the key exchange protocol in [5] by using the leaked information about the secret key from the signal messages. Then, Liu, Zheng and Zou proposed another signal leakage attack [11] against the reconciliation-based NewHope-Usenix key exchange protocol [12]. Zhao and Gong proposed a small field attack for R-LWE-Based one-pass authenticated key exchange using the algebraic structure in R-LWE [13].

The idea of key mismatch attack was first proposed by Ding, Fluhrer and Saraswathy [14], which is proposed against a one pass case of the protocol in [5]. In a key mismatch attack, a participant's public key is reused and its private key is recovered by comparing whether the shared keys between two participants match or not. In [15], Bauer et al. proposed a key mismatch attack against NewHope KEM [16]. Unfortunately, there are some problems with the attack which cannot recover all the private keys ranging from $-6$ to 4 as they wish. In [17], Qin, Cheng and Ding proposed an improved and complete key mismatch attack to recover all private keys ranging from $-8$ to 8 in NewHope KEM [16] with extremely high accuracy and efficiency. In EUROCRYPT 2019, Băetu et al. gave a bound on the queries needed in the key mismatch attack on the chosen plaintext attack (CPA) versions of a number of postquantum cyrptosystem, as well as a attack on the chosen ciphertext attack (CCA) versions when they have quantum access to the decryption oracle [18]. But they did not analysis the security of Kyber under key mismatch attacks. Since Kyber is designed in a different algebraic structure, it is appealing to assess Kyber's security when the public key is reused.

**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 demonstrate the efficiency of the proposed attacks. Our best record is that we can 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.

**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.

**Organization of this paper.** In Section II, 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 III, IV, and V, respectively. In Section VI we go on proposing two improved attacks that effectively reduce the needed time and queries in the original attacks. The experiments in Section VII demonstrate the efficiency of our proposed attacks, and the conclusion is given in Section VIII.

## II. Preliminaries

### A. 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 $a_i \in \mathbb{Z}_q$ $(0 \le i \le n-1)$ 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 $\mathbf{c}[i]$ $(0 \le i \le n-1)$ represents the $i\text{-}th$ 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. $\lceil x \rfloor = \lfloor x + \frac{1}{2} \rfloor$.

### B. LWE & R-LWE & M-LWE

In the groundbreaking work of Ajtai in 1996 [19], he pointed out that we can use lattice to build cryptosystems. In 2005, Regev introduced the LWE problems [20] and 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. In order to solve this problem, Lyubashesky, Peikert and Regev in 2010 proposed the R-LWE problem, which is to use the polynomials in the ring [21]. 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 Module-LWE 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.

### C. The Kyber KEM

TABLE I
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) |

Kyber is an IND-CCA2-secure KEM, which is a part of the Cryptographic Suite for Algebraic Cipher Suite (CRYSTALS) [22–24]. Kyber's security is based on the hardness of solving the M-LWE problem, which makes Kyber different from NewHope [16]. 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 I 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: $\mathbb{Z}_q \to \mathbb{Z}_{2^d}$:

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

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

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

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:

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

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

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:

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

and

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

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

In Table II, 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 II. The three different functions $\mathrm{G}(\cdot)$, $\mathrm{H}(\cdot)$ and $\mathrm{KDF}(\cdot)$ in Table II 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 $\mathrm{Gen}()$ to generate its key pair. In Step 2 of Table II, 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 $\mathrm{Enc}()$. In Step 6 of Table II, 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 $\mathrm{Dec}()$ to get $m'$. According to Step 9 of Table II, 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 II
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. $\mathbf{s}_A = (\mathbf{s}'_A \| \mathbf{P}_A \| \mathrm{H}(\mathbf{P}_A) \| z)$ | $\xrightarrow{\mathbf{P}_A}$ | 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. $k_B$ = 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** $(\mathbf{c}_1, \mathbf{c}_2)$ = $(\mathbf{c}'_1, \mathbf{c}'_2)$ | | |
| $k_A$ = KDF$(K', \mathrm{H}(\mathbf{c}'_1, \mathbf{c}'_2))$ | | |
| **else** | | |
| $k_A$ = KDF$(z, \mathrm{H}(\mathbf{c}'_1, \mathbf{c}'_2))$ | | |

**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
    $k_A$ = KDF$(K', \mathrm{H}(\mathbf{c}'_1, \mathbf{c}'_2))$;
    Return 1;
else
    $k_A$ = KDF$(z, \mathrm{H}(\mathbf{c}'_1, \mathbf{c}'_2))$;
    Return 0;
end
```

## III. The Key Mismatch Attack on Kyber-1024

In this section, we will propose the key mismatch attack on Kyber-1024. According to Table I, 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.

### *A. Key Mismatch Oracles*

We build an oracle $\mathcal{O}$ that simulates Alice in Table II. 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 $\mathrm{Enc}()$ 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.

**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
```

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 $\mathrm{Enc}()$, 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.

## *B. 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 $\mathbf{P}_B[0] = \lceil \frac{q}{32} \rfloor$, otherwise he let $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{32} \rfloor$. 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

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

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

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

$$\begin{aligned}\mathbf{m}'[0] =& \mathbf{Compress}_q((\mathbf{v}_A - \mathbf{s}_A^T\mathbf{u}_A)[0], 1) \\ =& \mathbf{Compress}_q(\mathbf{v}_A[0] - (\mathbf{s}_A^T\mathbf{u}_A)[0], 1) \\ =& \left\lceil \frac{2}{q} \left( \mathbf{v}_A[0] - (\mathbf{s}_A^T\mathbf{u}_A)[0] \right) \right\rfloor \bmod 2.\end{aligned} \tag{7}$$

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

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

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 $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{32} \rfloor$, then the constant term in $\mathbf{s}_A^T\mathbf{u}_A$ is

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

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.

## *C. The Proposed Attack*

**Algorithm 3:** Key-recovery-Kyber-1024

```
Output: $s'$
Set $\mathbf{m} = \{1, 0, \cdots, 0\}^{256}$;
for $i := 0$ to $3$ do
    for $k := 0$ to $255$ do
        $\mathbf{P}_B = \mathbf{0}$;
        if $k = 0$ then
            $\mathbf{P}_B[0] = \lceil \frac{q}{32} \rfloor$;
        else
            $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{32} \rfloor$;
        $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$;
        for $h := 0$ to $31$ do
            $\mathbf{c}_2 = \mathbf{0}$ except $\mathbf{c}_2[0] = h$;
            $t = Oracle_m((\mathbf{c}_1, \mathbf{c}_2), m)$;
            if $t = 1$ then
                break;
        end
        $s'[i * 256 + k] = Check(h)$;
    end
end
Return $\mathbf{s}'$
```

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:

**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.t 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 $k = 0$, $\mathbf{P}_B[0] = \lceil \frac{q}{32} \rfloor$, otherwise $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{32} \rfloor$. 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

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

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 III. In Table III, $sk$ represents the possible value of the coefficient in $\mathbf{s}_A^T[0]$.

TABLE III
THE RELATIONSHIP BETWEEN $sk$ AND $h$ IN KYBER-1024

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

With the recorded $h$ in Step 3, $\mathcal{A}$ is able to recover $sk$ corresponding to $h$ in Table III. 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 III.

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

(1) If the 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 $\mathbf{m}'[0]$ truns to 1.

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

## IV. 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 I, 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.

### A. Parameters choices

We still use the same Oracle $\mathrm{O}_m$ 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 $\mathbf{c}_2 = \mathbf{Compress}_q(\mathbf{v}_B, d_{v_B}) = \mathbf{Compress}\ (\mathbf{v}_B, 4)_q = \left\lceil \frac{16}{q} \cdot \mathbf{v}_B \right\rfloor$ (mod 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

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

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.

### *B. 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              $\mathbf{P}_B[0] = \lceil \frac{q}{16} \rfloor$;
7          else
8              $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{16} \rfloor$;
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 will chose the parameters $\mathbf{m}, \mathbf{c}_1$ and $\mathbf{c}_2$ as previously described. That is, if $k = 0$, $\mathcal{A}$ sets $\mathbf{P}_B[0] = \lceil \frac{q}{16} \rfloor$, otherwise $\mathcal{A}$ sets $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{16} \rfloor$.

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

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

Since $\frac{2}{q} \lceil \frac{q}{16} \rfloor \approx 0.1249625$, 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 IV.

TABLE IV
THE RELATIONSHIP BETWEEN $sk$ AND $h$ IN KYBER-768

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

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

## V. 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 I, we know $(d_{\mathbf{P}_B}, d_{\mathbf{v}_B}) = (10, 3)$. The Oracle we use in this section is still $\mathrm{O}_m$ 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.

### *A. 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 $k = 0$, $\mathbf{P}_B[0] = \lceil \frac{q}{8} \rfloor$ and $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{8} \rfloor$ otherwise. The resulted $\mathbf{c}_1$ can be caclulated 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

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

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.

## B. The Proposed Attack

**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              $\mathbf{P}_B[0] = \lceil \frac{q}{8} \rfloor$;
7          else
8              $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{8} \rfloor$;
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         $s'[i * \text{Kyber\_N} + k] = Check(h)$;
17     end
18 end
```

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

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

since $\frac{2}{q}\lceil\frac{q}{8}\rfloor \approx 0.249924$. 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.

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

TABLE V
THE RELATIONSHIP BETWEEN $sk$ AND $h$, $h_1$ ON KYBER-512

| $sk$ | 0 | -1 | -2 | 1 | 2 | |
|---|---|---|---|---|---|---|
| $h$ | 3 | 2 | 1 | 4 | 4 | 5 |
| $h_1$ | | | | 2 | 1 | 1 |

But there 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 $\mathbf{P}_B[0] = \lceil -\frac{q}{8} \rfloor$, if $k = 0$, and $\mathbf{P}_B[256-k] = \lceil \frac{q}{8} \rfloor$ 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 V. 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 V. Otherwise, we will recover the value of $sk$ based on $h$ and $h_1$ in Table V.

## VI. 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.

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 $h$s start from $h = 0$, which is not efficient. By analysing Tables III, IV and V, 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 $\pm 1$ or $\pm 2$. The specific key mismatch attacks are shown in Algorithms 6, 7 and 8, respectively.

### A. 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 VI and Algorithm VI.

TABLE VI
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 | 11 | 10 |
| $O_m$ | 1 | 0 | 1 | | 0 | 1 | | | 0 | 1 | | 1 | | 0 |

According to Table III 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:

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

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

```
Output: $s'$
Set $m = \{1, 0, \cdots, 0\}^{256}$;
for $i := 0$ to $3$ do
    for $k := 0$ to $255$ do
        $\mathbf{P}_B = \mathbf{0}$;
        if $k = 0$ then
            $\mathbf{P}_B[0] = \lceil \frac{q}{32} \rfloor$;
        else
            $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{32} \rfloor$;
        $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$; $\mathbf{c}_2 = \mathbf{0}$ ;
        if $\mathbf{c}_2[0] = 9$ and $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 1$ then
            if $\mathbf{c}_2[0] = 8$ and $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 0$ then
                $s'[i * 256 + k] = 0$;
            else if $\mathbf{c}_2[0] = 7$ 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 $\mathbf{c}_2[0] = 9$ and $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 0$ then
            if $\mathbf{c}_2[0] = 10$ 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
```

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.

## B. 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 Algorithm VII.

TABLE VII
THE RELATIONSHIP BETWEEN $sk$ & $h$ & $O_m$ IN KYBER-768

| $sk$ | 0 | | -1 | | | -2 | | | | 1 | | 2 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $h$ | 5 | 4 | 5 | 4 | 3 | 5 | 4 | 3 | 2 | 5 | 6 | 5 | 6 | 7 |
| $O_m$ | 1 | 0 | 1 | | 0 | 1 | | | 0 | 0 | 1 | 0 | | 1 |

## C. 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 0;
2) If 1) holds, then we set $h = 4$ and the output of the Oracle is 0;
3) If 2) holds, then we let $h = 5$ and Oracle will return 1;
4) If 3) holds, we will reset $\mathbf{P}_B$ and $h_1 = 0$, the Oracle returns 0.

TABLE VIII
THE RELATIONSHIP BETWEEN $sk$ & $h$ & $h_1$& $O_m$ IN KYBER-512

| $sk$ | 0 | | -1 | | | -2 | | | | 1 | | 2 | | | 2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $h$ | 3 | 2 | 3 | 2 | 1 | 3 | 2 | 1 | 0 | 3 | 4 | 3 | 4 | 5 | 3 | 4 |
| $O_m$ | 1 | 0 | 1 | | 0 | 1 | | | 0 | 0 | 1 | 0 | | 1 | 0 | 1 |
| $h_1$ | | | | | | | | | | 1 | | 0 | | | 1 | |
| $O_m$ | | | | | | | | | | 0 | | 0 | | | 1 | |

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.

**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
            $\mathbf{P}_B[0] = \lceil \frac{q}{16} \rfloor$;
        else
            $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{16} \rfloor$;
        $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$;
        $\mathbf{c}_2 = \mathbf{0}$ ;
        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
```

The relationship between $sk$ and $h$, $h_1$ is shown in Table VIII and the specific $sk$ recovery process is shown in Algorithm 8.

## VII. 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

TABLE IX
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 $(s)$ | $4.5$ | $3.61$ | $2.2$ | $11.7$ | $5.88$ | $4.3$ | $31.3$ | $9.51$ | $7.5$ |

**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
            $\mathbf{P}_B[0] = \lceil \frac{q}{8} \rfloor$;
        else
            $\mathbf{P}_B[256 - k] = -\lceil \frac{q}{8} \rfloor$;
        $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$;
        $\mathbf{c}_2 = \mathbf{0}$ ;
        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
                    $\mathbf{P}_B[0] = -\lceil \frac{q}{8} \rfloor$;
                else
                    $\mathbf{P}_B[256 - k] = \lceil \frac{q}{8} \rfloor$;
            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
                $s'[i * 256 + k] = 2$;
            else if $\mathbf{c}_2[0] = 5$ and $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 1$ then
                if $k = 0$ then
                    $\mathbf{P}_B[0] = -\lceil \frac{q}{8} \rfloor$;
                else
                    $\mathbf{P}_B[256 - k] = \lceil \frac{q}{8} \rfloor$;
                if $\mathbf{c}_2[0] = 0$ and $O_m((\mathbf{c}_1, \mathbf{c}_2), m) = 0$ then
                    $s'[i * 256 + k] = 2$;
            end
        end
    end
end
```

Fig. 1. The Average Time (s)

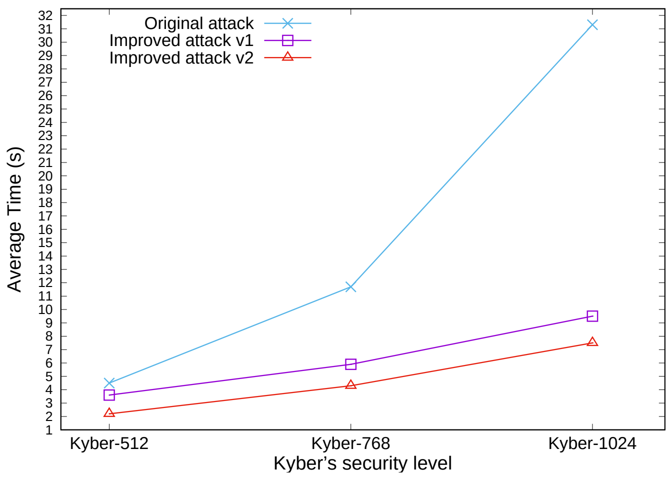
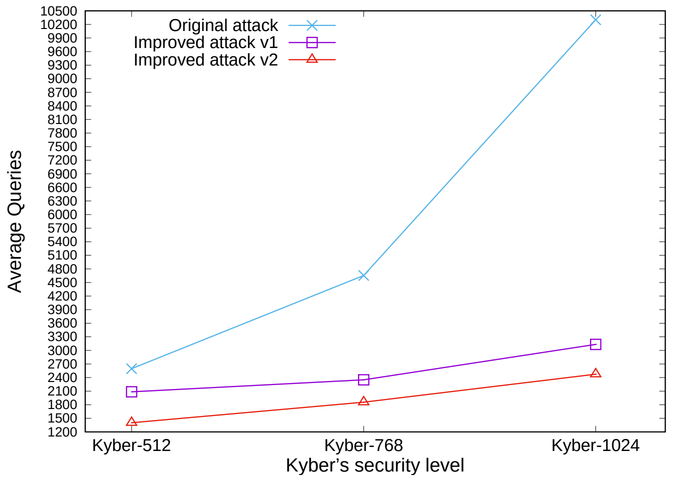


Fig. 2. The Average Number of Queries



of experiments. The first set is the original attack described in Sections III, IV, and V. The second is the *Improved attack* $v_1$ introduced in Section VI. The difference between the Original Attack and the *Improved attack* $v_1$ is that in the Original Attack $h$ starts from $h = 0$. The third set corresponds to Algorithms 6, 7 and 8 in the *Improved attack* $v_2$ described in Section VI. 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 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 IX, 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 [17], 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 $\eta$ 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.

## VIII. 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.

## References

[1] L. Gyongyosi and S. Imre, "A survey on quantum computing technology," *Computer Science Review*, vol. 31, pp. 51–71, 2019.

[2] D. Moody, *Post Quantum Cryptography Standardization: Announcement and outline of NIST's Call for Submissions.* PQCrypto 2016, Fukuoka, Japan, 2016, https://csrc.nist.gov/Presentations/2016/Announcement-and-outline-of-NIST-s-Call-for-Submis. Last accessed 11 July 2019.

[3] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography.* US Department of Commerce, National Institute of Standards and Technology, 2016.

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

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

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

[7] E. Rescorla, "The transport layer security (tls) protocol version 1.3," Tech. Rep., 2018, http://www.rfc-editor.org/info/rfc8446. Last accessed 26 February 2019.

[8] D. Kirkwood, B. C. Lackey, J. McVey, M. Motley, J. A. Solinas, and D. Tuller, "Failure is not an option: standardization issues for post-quantum key agreement," in *Workshop on Cybersecurity in a Post-Quantum World*, 2015, 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.

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

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

[11] C. Liu, Z. Zheng, and G. Zou, "Key reuse attack on newhope key exchange protocol," in *International*

*Conference on Information Security and Cryptology*. Springer, 2018, pp. 163–176.

[12] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchangeła new hope," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 327–343.

[13] B. Gong and Y. Zhao, "Cryptanalysis of rlwe-based one-pass authenticated key exchange," in *International Workshop on Post-Quantum Cryptography*. Springer, 2017, pp. 163–183.

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

[15] A. Bauer, H. Gilbert, G. Renault, and M. Rossi, "Assessment of the key-reuse resilience of newhope," in *Cryptographers Track at the RSA Conference*. Springer, 2019, pp. 272–292.

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

[17] Y. Qin, C. Cheng, and J. Ding, "A complete and optimized key mismatch attack on nist candidate newhope." in *European Symposium on Research in Computer Security, accepted*. Springer, 2019, https://eprint.iacr.org/2019/435.pdf. Last accessed 12 July 2019.

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

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

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

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

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

[23] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber: a cca-secure module-lattice-based kem," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 353–367, https://eprint.iacr.org/2017/634. Last accessed 24 June 2019.

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