Pairwise and Parallel: Enhancing the Key Mismatch Attacks on Kyber and Beyond

Mingyao Shao[2,3], Yuejun Liu[1], and Yongbin Zhou[1,2]

[1] School of Cyber Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
liuyuejun@njust.edu.cn

[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
shaomingyao@iie.ac.cn

[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Key mismatch attacks resilience is a great concern for KEMs in the NIST PQC standardization process. In key mismatch attacks, the adversary aims to recover the reused key by sending special form of ciphertexts to the target party and observing whether the shared key matches his guesses or not.

In this paper, we propose *pairwise-parallel key mismatch attacks* on Kyber and other lattice-based KEMs. The strategy is to recover partial information about multiple secret key coefficient-pairs in a parallel way per query. We realize the required multi-value key mismatch oracle in a simple key exchange scenario and experimentally validate our proposed attacks. Our attacks greatly reduce the number of queries required to recover the full secret key. Specifically, compared with state-of-the-art key mismatch attacks on CPA-secure Kyber, our attacks reduce the number of queries by 95% with computational complexity 2^{32}. Then we employ the post-processing with lattice reduction to further minimize the number of queries. The results show we only need 78 queries to recover the full secret key with a lattice reduction cost of 2^{32}. Moreover, our proposed *pairwise-parallel* attack method can be directly applied to enhance the PC oracle-based SCA against CCA-secure Kyber, reducing the number of queries/traces by 16.67%.

Keywords: Lattice-based cryptography · Key encapsulation mechanism · Key exchange · Key reuse · Key mismatch attacks · Kyber.

1 Introduction

In 1994, Shor [25] proposed a quantum algorithm capable of solving the integer factoring problem and discrete logarithm problem in polynomial time on a quantum computer. Consequently, widely used public key cryptographic algorithms based on these hard problems such as Diffie-Hellman key exchange (KE) become insecure once large-scale quantum computers are built.

In order to address these potential threats, the National Institute of Standards and Technology (NIST) launched a process in 2016 aimed at developing and standardizing post-quantum cryptography (PQC). Among these PQC candidate algorithms, lattice-based cryptography is regarded as a very promising one, as it provides both strong provable security and remarkable efficiency. Notably, the MLWE-based Kyber [24] was selected as the sole candidate for the standardization of Key Encapsulation Mechanisms (KEMs).

Most CCA-secure lattice-based KEMs follow a popular structure: combining a CPA-secure public key encryption (PKE) scheme with the Fujisaki-Okamoto (FO) transformation [9]. To enhance the efficiency, some KEMs opted for CPA-secure schemes without the FO transformation for key exchange. Before the practical deployment of these schemes, it is crucial to evaluate their security under realistic conditions, particularly when neglecting the FO transformation.

Currently, in the widely adopted Internet standards, the key reuse mode is commonly used. For example, in TLS 1.2 [5], the Diffie-Hellman KE provides static key mode that avoids recalculating the key for each session. And in TLS 1.3 [21], the pre-shared key (PSK) mode and zero round trip time (0-RTT) mode require the client and the server to maintain a long-term public key. However, CPA-secure lattice-based KEMs do not provide security guarantees when the public key is reused. If Alice (Decapsulator) reuses her public key, the adversary (Encapsulator) can recover her secret key by sending special ciphertexts and observing whether the shared key derived by Alice matches his guesses or not. This type of attacks is known as *key mismatch attacks* [1–3, 10, 14, 15, 18–20].

In 2015, Kirkwood et al. [11] revealed an issue in reconciliation-based RLWE KE. They proposed that if public key is reused, these schemes may be susceptible to an attack that can reveal the secret key through multiple key exchange executions. In 2016, Fluhrer [8] initiated key reuse attacks against some RLWE KE by sending a series of messages, and observing how the target party responds. Later, Ding et al. [3] extended the attacks to more schemes and proposed key mismatch attacks. It is known that, KE can be replaced by any secure KEM with an efficient key generation algorithm, and these attacks can be applied to many lattice-based KEMs. Bauer et al. [2] first proposed key mismatch attacks on NewHope [16]. Then Qin et al. [19] extended the attacks to Kyber [24]. In 2021, Qin et al. [20] proposed a unified method to evaluate the resilience of lattice-based KEMs against key mismatch attacks. They transformed the problem of finding the lower bound of the number of queries into finding an optimal binary recovery tree (BRT), and obtained the query bounds of recovering one secret key coefficient using Huffman coding. Later, Guo and Mårtensson [10] proposed multi-positional attacks which can recover partial information about multiple secret coefficients per query, breaking the Huffman bounds in [20]. Furthermore, Mi et al. [14] combined the key mismatch attack with a standard lattice attack, further significantly reducing the number of required queries.

Similar attacks can be launched on CCA-secure KEMs with the help of side-channel information [2, 7, 22, 23, 26, 27]. The adversary can construct a plaintext-checking (PC) oracle using side-channel information leakage from re-encryption. The PC oracle offers a binary response (correct or wrong) about the attacker's guess of the decrypted message for a given ciphertext. Recently, Rajendran et al. [23] and Tanaka et al. [26] proposed parallel PC oracle-based side-channel attacks (SCA), which can recover multiple bits of information about the secret key in a single query/trace.

The number of required queries plays a crucial role in key mismatch attacks, as it directly impacts both the efficiency and success rate of the attacks. How to recover the full secret key with fewer queries is still a very attractive topic.

Contributions. In this paper, we propose a new key mismatch attack to recover the full secret key with fewer queries. Our contributions are as follows:

1. We propose *pairwise-parallel key mismatch attacks* on Kyber , which can recover partial information about multiple coefficient-pairs of the secret key in a parallel way per query. Compared with state-of-the-art key mismatch attacks, we reduces the number of required queries by 95% with computational complexity 2^{32}, when parallel level $P = 26$.
2. We further employ the post-processing with lattice reduction to reduce the number of required queries and use the leaky LWE estimator in [6] to measure the lattice reduction cost. Specifically, when $P = 26$, we only need 78 queries to recover the full secret key with a lattice reduction cost of 2^{32}.
3. We realize a practical multi-value key mismatch oracle which can recover multiple bits of the decrypted message in a simple key exchange scenario and experimentally validate our proposed attacks.
4. Our *pairwise-parallel* attack method can be directly applied to enhance the efficiency of PC oracle-based SCA against CCA-secure Kyber, and reduce the number of required queries/traces by 16.67% for Kyber768/1024.

Organization. In Section 2, we introduce some basic preliminaries. In Section 3, we recap previous works and our motivations. In Section 4, we present our pairwise-parallel key mismatch attacks and show how to realize a practical multiple-value key mismatch oracle in key exchange scenario. Then we show the experimental evaluation in Section 5. In Section 6, we employ lattice reduction to further reduce the number of queries, and discuss the applications of our attacks on CCA-secure KEMs. Finally, Section 7 concludes this work.

2 Preliminaries

2.1 Notation

We denote the ring of integers modulo $q \in \mathbb{Z}^+$ as $\mathbb{Z}_q$. R_q represents the polynomial ring $\mathbb{Z}_q[x]/(x^n + 1)$, and polynomials in R_q are denoted using bold lower case letters (i.e.) $\mathbf{a} \in R_q$. The i^{th} coefficient of $\mathbf{a}$ is denoted as $\mathbf{a}[i]$. By default, all vectors will be column vectors. A matrix of polynomials in $R_q^{k\times l}$ are denoted using bold upper case letters (i.e.) $\mathbf{A}$. The transpose of a matrix $\mathbf{A}$ is denoted as $\mathbf{A}^T$. For any positive integer p, we define $r' = r\,mod^+\,p$ to be the unique element r' in the range $0 \le r' < p$ such that $r' = r\,mod\,p$. For an element $x \in \mathbb{Q}$, we denote by $\lceil x \rfloor$ rounding of x to the closest integer with ties being rounded up.

2.2 CPA-Secure Version of Kyber

Kyber [24] is the KEM proposal of cryptographic suite for algebraic lattices (CRYSTALS), based on the MLWE problem. The CCA-secure variant of Kyber KEM is constructed by a simple CPA-secure PKE, denoted as Kyber.CPAPKE, along with the FO transformation. To assess its key reuse resilience, we present a possible CPA-secure version of Kyber in Fig. 1. This variant can be viewed as a modification of Kyber.CPAPKE described in [24].

Alice		**Bob**
1. Kyber.CPAPKE.KeyGen()		
1.1 Generate matrix $\mathbf{A} \in R_q^{k\times k}$		
1.2 Sample $\mathbf{s}_A, \mathbf{e}_A \in \mathbf{B}_{\eta_1}^k$		2. $\mathbf{m} \xleftarrow{\$} \{0,1\}^{256}$
1.3 $\mathbf{P}_A \leftarrow \mathbf{A}\mathbf{s}_A + \mathbf{e}_A$		3. Kyber.CPAPKE.Enc($\mathbf{P}_A, \mathbf{m}$)
1.4 Output: $(\mathbf{s}_A, \mathbf{P}_A)$	$\xrightarrow{\mathbf{p}_A}$	3.1 Generate matrix $\mathbf{A} \in R_q^{k\times k}$
		3.2 Sample $\mathbf{s}_B \in \mathbf{B}_{\eta_1}^k, \mathbf{e}_B \in \mathbf{B}_{\eta_2}^k, \mathbf{e}'_B \in \mathbf{B}_{\eta_2}$
		3.3 $\mathbf{P}_B \leftarrow \mathbf{A}^T\mathbf{s}_B + \mathbf{e}_B$
5. Kyber.CPAPKE.Dec($\mathbf{s}_A, \mathbf{P}_B, \bar{\mathbf{c}}$)		3.4 $\mathbf{v}_B \leftarrow \mathbf{P}_A^T\mathbf{s}_B + \mathbf{e}'_B + \mathbf{Decompress}_q(\mathbf{m}, 1)$
5.1 $\mathbf{u}_A \leftarrow \mathbf{Decompress}_q(\mathbf{c}_1, d_u)$		3.5 $\mathbf{c}_1 \leftarrow \mathbf{Compress}_q(\mathbf{P}_B, d_u)$
5.2 $\mathbf{v}_A \leftarrow \mathbf{Decompress}_q(\mathbf{c}_2, d_v)$	$\xleftarrow{(\mathbf{p}_B,\bar{\mathbf{c}})}$	3.6 $\mathbf{c}_2 \leftarrow \mathbf{Compress}_q(\mathbf{v}_B, d_v)$
5.3 $\mathbf{m}' \leftarrow \mathbf{Compress}_q(\mathbf{v}_A - \mathbf{s}_A^T\mathbf{u}_A, 1)$		3.7 Output: $\bar{\mathbf{c}} := (\mathbf{c}_1, \mathbf{c}_2)$
5.4 Output: $\mathbf{m}'$		4. $K_B \leftarrow \mathbf{H}(\mathbf{m}\|\|(\mathbf{p}_B, \bar{\mathbf{c}}))$
6. $K_A \leftarrow \mathbf{H}(\mathbf{m}'\|\|(\mathbf{p}_B, \bar{\mathbf{c}}))$		

Fig. 1: The CPA-secure version of Kyber.

In Kyber, R_q represents the ring $\mathbb{Z}_q[x]/(x^n+1)$, where $n = 256$ and $q = 3329$. Another parameter k is set to be 2, 3 or 4, which is in accordance with the three different security levels Kyber512, Kyber768, and Kyber1024. All the secret keys and error vectors are sampled from a centered binomial distribution $\mathbf{B}_\eta$. Here $\mathbf{B}_\eta$ is generated using $\sum_{i=1}^{\eta}(a_i - b_i)$, where a_i and b_i are independently randomly sampled from $\{0,1\}$. In Kyber512, $\eta = 3$, and in Kyber768/1024, $\eta = 2$. The $\mathbf{Compress}/\mathbf{Decompress}_q(x, d)$ functions are shown below:

$$\begin{aligned} \mathbf{Compress}_q(x, d) &= \lceil (2^d/q) \cdot x \rfloor \; mod^+ 2^d, \\ \mathbf{Decompress}_q(x, d) &= \lceil (q/2^d) \cdot x \rfloor. \end{aligned} \tag{1}$$

2.3 Model of Key Mismatch Attacks

In the key mismatch attacks, we suppose honest Alice reuses her public key $\mathbf{P}_A$. The adversary acts as Bob to negotiate shared key with Alice. Bob sends a series of special ciphertexts and observes whether the shared key derived by Alice is consistent with his guesses or not. Based on the information, Bob can potentially recover Alice's secret key $\mathbf{s}_A$.

To facilitate these attacks, the adversary typically utilizes a key mismatch oracle denoted as $\mathcal{K}$, as described in Algorithm 1. The input to $\mathcal{K}$, denoted as

$\mathbf{P}$, comprises the adversary's chosen-ciphertexts $\mathbf{P}_B$, $\bar{\mathbf{c}}$, as well as the shared key K_B. The output of $\mathcal{K}$ is either 1 or 0. To be specific, with the received $\mathbf{P}_B$ and $\bar{\mathbf{c}}$, $\mathcal{K}$ calls the function $KEM.Dec(\mathbf{P}_B, \bar{\mathbf{c}})$ and gets the shared key K_A. If the shared keys K_A and K_B match, $\mathcal{K}$ outputs 1, otherwise, it outputs 0.

Algorithm 1 The key mismatch oracle $\mathcal{K}$ and key mismatch attacks

```
Key mismatch oracle $\mathcal{K}$(P)
Input: $\mathbf{P} := (\mathbf{P}_B, \bar{\mathbf{c}}, K_B)$
Output: 0 or 1
1: $K_A \leftarrow KEM.Dec(\mathbf{P}_B, \bar{\mathbf{c}})$
2: if $K_A = K_B$ then
3:     Return 1
4: else
5:     Return 0
6: end if
```

```
Key mismatch attacks
Input: Alice's $\mathbf{P}_A$ and oracle $\mathcal{K}$
Output: 0 or 1
1: $\mathbf{s}'_A \leftarrow \mathrm{Bob}^{\mathcal{K}}(\mathbf{P}_A)$
2: if $\mathbf{s}'_A = \mathbf{s}_A$ then
3:     Return 1
4: else
5:     Return 0
6: end if
```

3 Previous Works and Our Motivations

Key mismatch attacks against lattice-based KEMs have been extensively studied. In the following discussion, we take Kyber1024 as an example to introduce the previous major works and our motivations.

3.1 One-Position Key Mismatch Attacks

To begin with, let us explain the mechanism of one-position key mismatch attacks which retrieving partial information about one secret key coefficient per query. Let's consider the case of recovering $\mathbf{s}_A[0]$. Firstly, Bob sets the message $\mathbf{m} = [1, 0, ..., 0]$. Then he selects special ciphertexts $(\mathbf{P}_B, \bar{\mathbf{c}})$ such that Alice's decrypted message is 0 by design on all position except for the position 0, whose value depends on the secret value $\mathbf{s}_A[0]$. By observing the output of key mismatch oracle, Bob obtains the value of $\mathbf{m}[0]$, then deduce partial information about the value of $\mathbf{s}_A[0]$. Repeating this process, Bob gradually narrows down the potential values of $s_A[0]$ until he identifies a unique value.

We present the specific ciphertexts selection scheme as follows. Firstly, Bob selects $\mathbf{P}_B = [\lceil \frac{q}{32} \rfloor, 0, ..., 0], \mathbf{c}_2 = [h, 0, ..., 0], h \in \mathbb{Z}_q$, then he compresses $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_u)$, and sends $(\mathbf{P}_B, \mathbf{c}_1, \mathbf{c}_2)$ to Alice. Secondly, Alice decompresses $\mathbf{u}_A = \mathbf{Decompress}_q(\mathbf{c}_1, d_u) = \mathbf{P}_B$, and $\mathbf{v}_A = \mathbf{Decompress}_q(\mathbf{c}_2, d_v) = [\lceil \frac{q}{32} h \rfloor, 0, ..., 0]$. Then Alice decrypts to get

$$\begin{aligned}\mathbf{m}'[i] &= \mathbf{Compress}_q((\mathbf{v}_A - \mathbf{s}_A^T \mathbf{u}_A)[i], 1) \\ &= \begin{cases} \left\lceil \frac{2}{q} \left(\lceil \frac{q}{32} h \rfloor - \mathbf{s}_A^T[0] \lceil \frac{q}{32} \rfloor \right) \right\rfloor \ mod\ 2, & i = 0, \\ \left\lceil \frac{2}{q} \left(0 - \mathbf{s}_A^T[i] \lceil \frac{q}{32} \rfloor \right) \right\rfloor \ mod\ 2 = 0, & i \geq 1. \end{cases}\end{aligned} \tag{2}$$

The value of $\mathbf{m}'[0]$ depends on both h and $\mathbf{s}_A[0]$ according to Table 1. For index $i \geq 1$, the expression of $\mathbf{m}'[i]$ within the outer rounding function is bounded in absolute value by $2/q \cdot \eta \cdot \lceil \frac{q}{32} \rfloor \approx 0.125 < 1/2$. Hence these values are always equal to 0 when rounded to the nearest integer. Thirdly, Bob and Alice derive the shared keys $K_B \leftarrow \mathbf{H}(\mathbf{m}||(\mathbf{p}_B, (\mathbf{c}_1, \mathbf{c}_2)))$ and $K_A \leftarrow \mathbf{H}(\mathbf{m}'||(\mathbf{p}_B, (\mathbf{c}_1, \mathbf{c}_2)))$ respectively.

With the help of the key mismatch oracle, Bob knows whether $K_A = K_B$ or not. When $K_A = K_B$, it means $\mathbf{m}' = \mathbf{m}$ and $\mathbf{m}'[0] = 1$, otherwise $\mathbf{m}'[0] = 0$. Then Bob can infer the range of $\mathbf{s}_A[0]$ according to Table 1. Finally, Bob can narrow the range of $\mathbf{s}_A[0]$ in half each time by querying repeatedly with different parameter h. Based on the distribution of the secret key, we list a possible querying order of h and corresponding BRT in Fig. 2. It shows that Bob needs 3 queries at most to recover one coefficient using binary search approach.

Besides, if Bob aims to recover $\mathbf{s}_A[i], i \neq 0$, he can set $\mathbf{P}_B[256 - i] = -\lceil \frac{q}{32} \rfloor$. Then $\mathbf{m}'[0] = \lceil \frac{2}{q}(\lceil \frac{q}{32} h \rfloor - \mathbf{s}_A^T[i] \lceil \frac{q}{32} \rfloor) \rfloor \ mod\ 2$.

Table 1: The relationships of $\mathbf{m}'[0]$ with $\mathbf{s}_A[0]$ and h for Kyber1024.

$\mathbf{m}'[0]$ \ $\mathbf{s}_A[0]$ / h	-2	-1	0	1	2
7	1	0	0	0	0
8	1	1	0	0	0
9	1	1	1	0	0
10	1	1	1	1	0

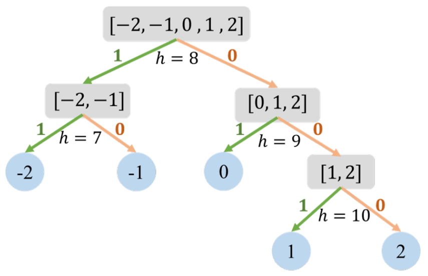


Fig. 2: The BRT for Kyber1024.

Complexity. For Kyber1024, recovering one coefficient at most requires 3 number of queries, as $2^2 < 5 < 2^3$. Therefore, the number of queries required for full secret key recovery, denoted as $\mathcal{Q}_{query}$ is given as:

$$\mathcal{Q}_{query} = 3 \cdot 256 \cdot k = 3000. \tag{3}$$

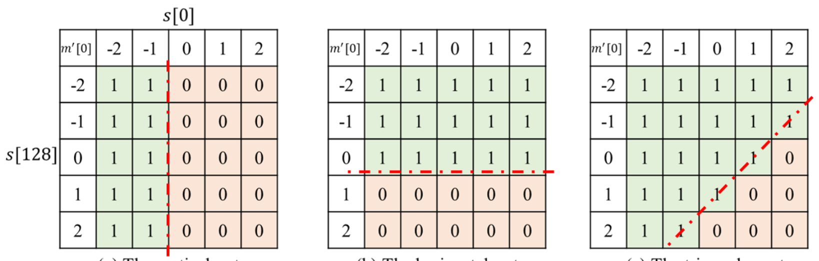
3.2 Two-Position Key Mismatch Attacks

The limitation of one-position key mismatch attacks is that the adversary only can recover partial information about one secret key coefficient per query. Then Guo and Mårtensson [10] proposed two-positional key mismatch attacks which can recover partial information about secret key coefficient-pair in each query.

The only difference from one-position key mismatch attacks is that, Bob sets $\mathbf{P}_B$ the value 0 on all positions, except that $\mathbf{P}_B[0] = b_1 \lceil \frac{q}{32} \rfloor$ and $\mathbf{P}_B[128] = b_2 \lceil \frac{q}{32} \rfloor$, where $b_1, b_2 \in \{-1, 0, 1\}$. Then Alice decompresses and decrypts to get

$$\mathbf{m}'[i] = \begin{cases} \left\lceil \frac{2}{q} \left(\left\lceil \frac{q}{32} h \right\rfloor - (b_1 \mathbf{s}_A^T[0] - b_2 \mathbf{s}_A^T[128]) \left\lceil \frac{q}{32} \right\rfloor \right) \right\rfloor mod\ 2, & i = 0, \\ \left\lceil \frac{2}{q} \left(0 - (b_1 \mathbf{s}_A^T[i] - b_2 \mathbf{s}_A^T[128+i]) \left\lceil \frac{q}{32} \right\rfloor \right) \right\rfloor mod\ 2 = 0, & i \geq 1. \end{cases} \tag{4}$$

The value of $\mathbf{m}'[0]$ is determined by h, b_1, b_2 and $\mathbf{s}_A[0], \mathbf{s}_A[128]$. For $\mathbf{m}'[i], i \geq 1$, the expression within the outer rounding function is bounded in absolute value by $2/q \cdot 2 \cdot \eta \cdot \lceil \frac{q}{32} \rfloor \approx 0.25 < 1/2$, thus $\mathbf{m}'[i] = 0$ when $i \geq 1$. We arrange the possible values of $(\mathbf{s}_A[0], \mathbf{s}_A[128])$ in a two-dimensional grid. Based on the different parameters b_1, b_2, we can obtain three distinct types of cuts on the grid: a *vertical cut*, a *horizontal cut*, and a *triangular cut*, as illustrated in Fig. 3.



$m'[0]$	-2	-1	0	1	2
-2	1	1	0	0	0
-1	1	1	0	0	0
0	1	1	0	0	0
1	1	1	0	0	0
2	1	1	0	0	0

$m'[0]$	-2	-1	0	1	2
-2	1	1	1	1	1
-1	1	1	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
2	0	0	0	0	0

$m'[0]$	-2	-1	0	1	2
-2	1	1	1	1	1
-1	1	1	1	1	1
0	1	1	1	1	0
1	1	1	1	0	0
2	1	1	0	0	0

(a) The vertical cut $b_1 = 1, b_2 = 0, h = 8$ (b) The horizontal cut $b_1 = 0, b_2 = -1, h = 9$ (c) The triangular cut $b_1 = 1, b_2 = -1, h = 10$

Fig. 3: The cuts with respect to $\mathbf{m}'[0]$ under different parameters (b_1, b_2, h).

Complexity. For Kyber1024, there are $5 \times 5 = 25$ possible values of $(\mathbf{s}_A[i], \mathbf{s}_A[i+128])$. According to the binary search approach, we need to perform at most 5 cuts to uniquely determine the value of one coefficient-pair, as $2^4 < 25 < 2^5$. Therefore, the number of queries required for full secret key recovery is given as:

$$\mathcal{Q}_{query} = 5 \cdot 256/2 \cdot k = 2500. \tag{5}$$

Compared with one-position key mismatch attacks, this attacks reduce the number of queries by 16.67%.

Shannon Bound. Obviously, there is a lower bound for the number of queries in key mismatch attacks according to the Shannon entropy:

$$H(X) = -\sum_x P(x) log_2[P(X)]. \tag{6}$$

Under key mismatch oracle, one query can only provide 1 bit information. Therefore the number of queries cannot be lower than the Shannon entropy of the secret key. According to the probability distribution of the secret key in Kyber1024, the entropy of one coefficient is $-(2\frac{1}{16}log_2\frac{1}{16} + 2\frac{1}{4}log_2\frac{1}{4} + \frac{3}{8}log_2\frac{3}{8}) \approx 2.03$. Therefore the Shannon bound of the number of queries required for recovering the full secret key in Kyber1024 is $1024 \times 2.03 \approx 2079$.

3.3 Parallel PC Oracle-based SCA

For CCA-secure KEMs, Rajendran et al. [23] and Tanaka et al. [26] proposed parallel PC oracle-based SCA, which can recover a generic P number of bits of information about the secret key in a single query/trace. And they realized the required parallel PC oracle utilizing the EM side-channel information leakage from the re-encryption procedure in FO transformation, which is capable of recovering first P number of bits of the plaintext.

We find that this parallel PC oracle-based attacks can be applied to key mismatch attacks with a slight modification. Specifically, Bob selects $\mathbf{P}_B = [\lceil \frac{q}{32} \rfloor, 0, ..., 0], \mathbf{c}_2 = [h_0, h_1, ..., h_{P-1}, 0, ..., 0]$, then he compresses $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, d_u)$, and sends$(\mathbf{P}_B, \mathbf{c}_1, \mathbf{c}_2)$ to Alice. Then Alice decompresses $\mathbf{u}_A = \mathbf{Decompress}_q(\mathbf{c}_1, d_u) = \mathbf{P}_B$, $\mathbf{v}_A = \mathbf{Decompress}_q(\mathbf{c}_2, d_v) = [\lceil \frac{q}{32} h_0 \rfloor, \lceil \frac{q}{32} h_1 \rfloor, ..., \lceil \frac{q}{32} h_{P-1} \rfloor, 0, ..., 0]$, and decrypts to get

$$\mathbf{m}'[i] = \begin{cases} \left\lceil \frac{2}{q} \left(\lceil \frac{q}{32} h_i \rfloor - \mathbf{s}_A^T[i] \lceil \frac{q}{32} \rfloor \right) \right\rfloor \; mod \; 2, & i < P, \\ \left\lceil \frac{2}{q} \left(0 - \mathbf{s}_A^T[i] \lceil \frac{q}{32} \rfloor \right) \right\rfloor \; mod \; 2 = 0, & i \geq P. \end{cases} \tag{7}$$

The values of $\mathbf{m}'[i], i < P$ depend on h_i and $\mathbf{s}_A[i]$ correspondingly. For index $i \geq P$, $\mathbf{m}'[i] = 0$ as that in one-position key mismatch attacks.
Complexity. For Kyber1024, with the help of parallel PC oracle, Bob can recover P number of coefficients of $\mathbf{s}_A$ in parallel within 3 queries. Therefore, the number of queries required for full secret key recovery is given as:

$$\mathcal{Q}_{query} = 3 \cdot \lceil \frac{256}{P} \rceil \cdot k. \tag{8}$$

This result significantly breaks the Shannon bound of the number of queries under key mismatch oracle.

4 Pairwise-Parallel Key Mismatch Attacks

In order to further reduce the number of required queries, we propose *pairwise-parallel key mismatch attacks* with the help of multi-value key mismatch oracle. And we realize the multi-value key mismatch oracle in a simple CPA-secure KEM-based key exchange scenario.

4.1 *P*-Level Pairwise-Parallel Key Mismatch Attacks

The core idea of our attacks lies in constructing special ciphertexts such that the first P number of bits of $\mathbf{m}'$ depend upon the P corresponding coefficient-pairs of the secret key (i.e.) $(\mathbf{s}_A[i], \mathbf{s}_A[i+128]), 0 \leq i < P$ respectively as Fig. 6.

Let us focus on Kyber1024. Firstly, Bob lets $\mathbf{P}_B$ the value 0 on all positions, except that $\mathbf{P}_B[0] = b_1 \lceil \frac{q}{32} \rfloor$ and $\mathbf{P}_B[128] = b_2 \lceil \frac{q}{32} \rfloor$, where $|b_1| + |b_2| \leq 3$. Then he sets $\mathbf{c}_2 = [h_1, h_2, ..., h_{P-1}, 0, ..., 0]$, compresses $\mathbf{c}_1 =$

$\mathbf{Compress}_q(\mathbf{P}_B, d_u)$, and sends $(\mathbf{P}_B, \mathbf{c}_1, \mathbf{c}_2)$ to Alice. Secondly, Alice decompresses $\mathbf{u}_A = \mathbf{Decompress}_q(\mathbf{c}_1, d_u) = \mathbf{P}_B$, $\mathbf{v}_A = \mathbf{Decompress}_q(\mathbf{c}_2, d_v) = [\lceil \frac{q}{32} h_0 \rfloor, \lceil \frac{q}{32} h_1 \rfloor, ..., \lceil \frac{q}{32} h_{P-1} \rfloor, 0, ..., 0]$. Then Alice decrypts to get

$$\mathbf{m}'[i] = \begin{cases} \left\lceil \frac{2}{q} \left(\lceil \frac{q}{32} h_i \rfloor - (b_1 \mathbf{s}_A^T[i] - b_2 \mathbf{s}_A^T[128+i]) \lceil \frac{q}{32} \rfloor \right) \right\rfloor mod\ 2, & i < P, \\ \left\lceil \frac{2}{q} \left(0 - (b_1 \mathbf{s}_A^T[i] - b_2 \mathbf{s}_A^T[128+i]) \lceil \frac{q}{32} \rfloor \right) \right\rfloor mod\ 2 = 0, & i \geq P. \end{cases} \tag{9}$$

For $\mathbf{m}'[i]$, $i \geq P$, the expression within the outer rounding function is bounded in absolute value by $2/q \cdot 3 \cdot \eta \cdot \lceil \frac{q}{32} \rfloor \approx 0.375 < 1/2$ when $\eta = 2$. Hence these values are always equal to 0 when rounded to the nearest integer. The value of $\mathbf{m}'[i], i < P$ depends on h_i and $(b_1 \mathbf{s}_A[i] - b_2 \mathbf{s}_A[128+i])$, which can be obtained with the help of multi-value key mismatch oracle. For a separate coefficient-pair $(\mathbf{s}_A[i], \mathbf{s}_A[i+128])$, it is easy to find the suitable parameters to determine the unique value within 5 cuts according to Section 3.2. However, when we want the recover P number of coefficient-pairs of the secret key in a parallel way, can we still do it? We give a positive answer, next we will show how to select the parameters.

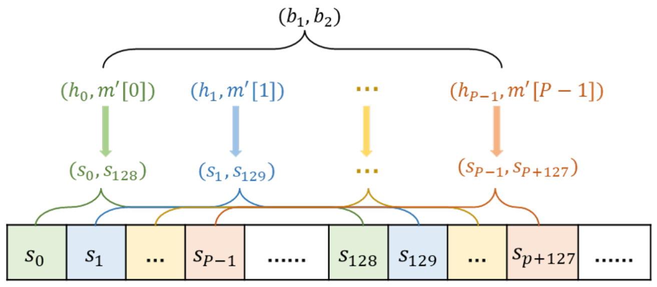


Fig. 4: Parallel dependencies between secret key coefficient-pairs $(\mathbf{s}_A[i], \mathbf{s}_A[i+128])$ and parameters $h_i, \mathbf{m}'[i], b_1, b_2$, $0 \leq i < P$ for Kyber1024.

Parameters Selection Strategies. There are three principles needed to be followed in parameters selection procedure:

1. Full Coverage: For the two-dimensional gird, there are $5 \times 5 = 25$ possible values of $(\mathbf{s}_A[i], \mathbf{s}_A[i+128])$. In each cut, We must adaptively choose the parameters according to the value of $\mathbf{m}'[i]$ (1 or 0) and parameters of previous rounds, so that every value can be uniquely determined.
2. Minimum Query: According to binary search method, in each cut, we should divide the region into two parts with the same size as possible, so as to use minimum number of queries to determine the value of each coefficient-pair.
3. Parallelism: According to the structure of the above chosen-ciphertexts, unique parameters (b_1, b_2) must be fixed and suitable for P number of coefficient-pairs in each cut.

Following these strategies, we present a feasible parameters selection scheme for Kyber1024, so that the P number of coefficient-pairs can be determined within 5 cuts. We put the specific diagram in appendix A.1. At Step 1, a triangular cut is chosen by setting $b_1 = 2, b_2 = -1$. At Step 2, a horizontal cut is chosen by setting $b_1 = 0, b_2 = -1$. At Step 3, a triangular cut is chosen by setting $b_1 = 1, b_2 = 1$. At Step 4, a triangular cut is chosen by setting $b_1 = 1, b_2 = 1$. At Step 5, a vertical cut is chosen by setting $b_1 = 1, b_2 = 0$. Then we show how to choose h_i for every state and how state changes based on $\mathbf{m}'[i]$ in Table 2. After each cut, we can narrow the range of $(\mathbf{s}_A[i], \mathbf{s}_A[i+128])$ based on $\mathbf{m}'[i]$ and parameters. Finally, we can derive the value based on the path of $\mathbf{m}'[i]$ as Table. 3. Additionally, we put the cuts diagram for Kyber512 in appendix A.2.

Table 2: The choice of h_i for every state and how the states change.

State	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
h_i	8	8	9	9	6	11	8	8	10	5	7	10
$\mathbf{m}'[i] = 1$	S2	S4	S6	S8	S10	S12	S14	(-2,-1)	S17	(-2,2)	S18	S19
$\mathbf{m}'[i] = 0$	S3	S5	S7	S9	S11	S13	S15	S16	(0,-2)	(-2,1)	(-1,0)	S20
State	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
h_i	12	7	9	7	8	7	9	10	10	8	9	10
$\mathbf{m}'[i] = 1$	S21	S22	S24	(-2,-2)	(-1,-2)	(-2,0)	(0,0)	(1,-1)	(1,-2)	(-1,2)	(0,1)	(1,1)
$\mathbf{m}'[i] = 0$	(2,-2)	S23	(2,1)	(-1,-1)	(0,-1)	(-1,1)	(1,0)	(2,0)	(2,-1)	(0,2)	(1,2)	(2,2)

Now we take parallel level $P = 2$ as an example, and try to recover $(\mathbf{s}_A[0], \mathbf{s}_A[128]) = (-1, 0)$ and $(\mathbf{s}_A[1], \mathbf{s}_A[129]) = (2, -1)$ following the above parameters selection scheme. We start from state1, set $(b_1, b_2, h_1, h_2) = (2, -1, 8, 8)$, thus $\mathbf{m}'[i] = \left\lceil \frac{2}{q} \left(\left\lceil \frac{q}{4} \right\rfloor - (2\mathbf{s}_A^T[i] + \mathbf{s}_A^T[128+i]) \left\lceil \frac{q}{32} \right\rfloor \right) \right\rfloor \ mod\ 2 = 1$ only if $(2\mathbf{s}_A^T[i] + \mathbf{s}_A^T[128+i]) \leq -1$. Thus we get $\mathbf{m}'[0] = 1, \mathbf{m}'[1] = 0$ as Fig.5. At the following steps, we change states and choose parameters based on Table 2. We show the cuts process in Fig. 6. After 5 steps, we can get two paths of the value of $\mathbf{m}'$, which are $\mathbf{m}'[0]$: 1000 and $\mathbf{m}'[1]$: 01010. Then we can derive the value of $(\mathbf{s}_A[0], \mathbf{s}_A[128]) = (-1, 0)$ and $(\mathbf{s}_A[1], \mathbf{s}_A[129]) = (2, -1)$ based on Table 3.

Table 3: The path of $\mathbf{m}'[i]$ within 5 cuts for Kyber1024.

Path	-2	-1	0	1	2
-2	11101	11011	1100	01011	0100
-1	1111	11100	11010	01101	01010
0	10011	1000	01111	01110	01100
1	1010	10010	00101	00011	0000
2	1011	00111	00110	00100	00010

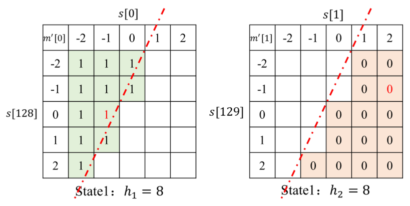


Fig. 5: Step1. $(b_1 = 2, b_2 = -1)$.

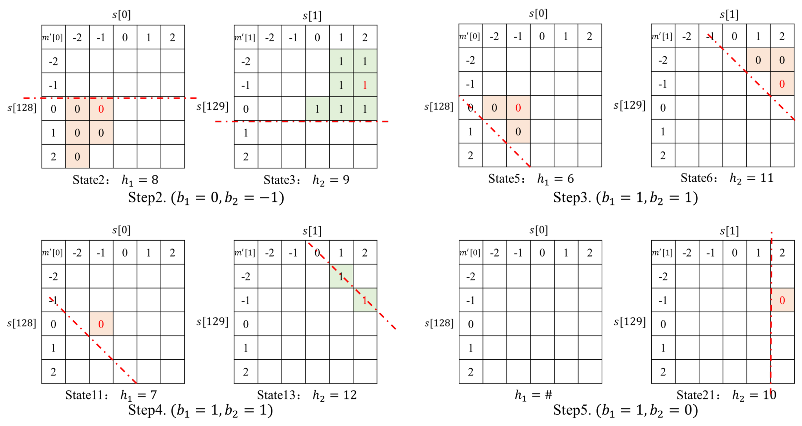


Fig. 6: The cuts process and parameters at Step2 to Step5

4.2 Multi-Value Key Mismatch Oracle in Key Exchange

In this section, we construct the multi-value key mismatch oracle in a simple CPA-secure KEM-based key exchange scenario. In key exchange protocol, Alice and Bob first negotiate the same shared key. Then they use the shared key (or the value derived from the shared key) as the session key for the next symmetric encrypted communication.

The construction consists of two stages. At the first stage, we assume that Bob interacts honestly with Alice, and negotiates the same shared key. Then Bob requests and decrypts to get a target data, denoted as **Target**. If Bob already knows **Target** in advance, this stage can be omitted.

At the second stage, Bob follows the above P-level pairwise-parallel key mismatch attacks, sending chosen-ciphertext to Alice. Then Bob requests the same target data, and receives a message encrypted by the shared key of Alice $\mathbf{c}_A = Enc(K_A, \mathbf{Target})$, where $K_A = \mathbf{H}(\mathbf{m}'||(\mathbf{P}_B, (\mathbf{c}_1, \mathbf{c}_2)))$. According to the analysis described above, $\mathbf{m}'$ has only 2^P number of possible values, corresponding to 2^P number of possible K_A. Next, Bob exhaustively tries the 2^P number of possible values of $\mathbf{m}'$, and computes the shared key $K_B = \mathbf{H}(\mathbf{m}'||(\mathbf{P}_B, (\mathbf{c}_1, \mathbf{c}_2)))$. Then Bob decrypts $\mathbf{c}_A$ using K_B, until the decryption result $Target' = Dec(K_B, \mathbf{c}_A) = \mathbf{Target}$. Finally, Bob obtains the values of $\mathbf{m}'[0] \sim \mathbf{m}'[P-1]$. Now, we have realized the multi-value key mismatch oracle, and we describe the key processes in Algorithm 2.

It is noted that the P-value key mismatch oracle requires 2^{P-1} number of offline decryption operations to achieve.

4.3 Complexity Analysis

Supposing we have an access to a multi-value key mismatch oracle, for Kyber768/1024, $\mathbf{s}_A \leftarrow \mathbf{B}_\eta, \eta = 2$, we can recover P number of coefficient-pairs (i.e.)

Algorithm 2 Multi-value key mismatch oracle and pairwise-parallel key mismatch attacks

```
Multi-value key mismatch oracle $\mathcal{O}$
Input: $\mathbf{P} := (P, \mathbf{c}_A, \mathbf{Target}, \mathbf{P}_B, \bar{\mathbf{c}})$
Output: $\mathbf{m}'[0] \sim \mathbf{m}'[P-1]$
1: for $i := 0$ to $2^P - 1$ do
2:     $\mathbf{m}' = bin(i)||0$
3:     $K_B = \mathbf{H}(\mathbf{m}'||(\mathbf{P}_B, \bar{\mathbf{c}}))$
4:     $Target' = Dec(K_B, \mathbf{c}_A)$
5:     if $Target' = \mathbf{Target}$ then
6:         Return $\mathbf{m}'[0] \sim \mathbf{m}'[P-1]$
7: end for
```

```
Pairwise-parallel key mismatch attacks
Input: Alice's $\mathbf{P}_A$ and oracle $\mathcal{O}$
Output: 0 or 1
1: $\mathbf{s}'_A \leftarrow \mathrm{Bob}^{\mathcal{O}}(\mathbf{P}_A)$
2: if $\mathbf{s}'_A = \mathbf{s}_A$ then
3:     Return 1
4: else
5:     Return 0
6: end if
```

$(\mathbf{s}_A[i], \mathbf{s}_A[i+128]), 0 \leq i < P$ in a parallel way within 5 queries. Therefore, the number of queries required for full secret key recovery is given as:

$$\mathcal{Q}_{query} = 5 \cdot \lceil \frac{256}{2P} \rceil \cdot k. \tag{10}$$

For Kyber512, $\eta = 3$, there are $7 \times 7 = 49 < 2^6$ kinds of value for coefficient-pair $(\mathbf{s}_A[i], \mathbf{s}_A[i+128])$. Thus we need 6 cuts to recover P number of coefficient-pairs. The number of queries required for full secret key recovery is given as:

$$\mathcal{Q}_{query} = 6 \cdot \lceil \frac{256}{2P} \rceil \cdot k. \tag{11}$$

Besides, to realize a P-value key mismatch oracle, an average of 2^{P-1} computational complexity is required. Therefore, the computational complexity required for recovering the full secret key, denoted as $\mathcal{Q}_{compute}$ is given as:

$$\mathcal{Q}_{compute} = 2^{P-1}\mathcal{Q}_{query}. \tag{12}$$

Although the compute process can be performed offline and not the key indicator of key mismatch attacks, it cannot exceed the actual computing power limit. For different parallel level P, the number of queries and corresponding computational complexity needed in our attacks are list in Table 4.

Table 4: The number of queries and corresponding computational complexity required in pairwise-parallel key mismatch attacks.

P	2		4		8		16		32	
Kyber512	768	$2^{10.6}$	384	$2^{11.6}$	192	$2^{14.6}$	96	$2^{21.6}$	48	$2^{36.6}$
Kyber768	960	$2^{11.2}$	480	$2^{12.2}$	240	$2^{15.2}$	120	$2^{22.2}$	60	$2^{36.9}$
Kyber1024	1280	$2^{11.6}$	640	$2^{12.6}$	320	$2^{16.6}$	160	$2^{22.6}$	80	$2^{37.3}$

5 Experimental Evaluation

In this section, we conduct experiments to confirm our attack's correctness and efficiency. Our experiments are implemented on a desktop equipped with four 0.90 GHz Intel Core m3-6Y30 CPUs and a 7.6 GB RAM.

5.1 Setup

As shown in Algorithm 3, we use Kyber1024 to illustrate the details of *pairwise-parallel key mismatch attacks.* We record the number of queries for different parallel level P under a perfect multi-value key mismatch oracle.

Algorithm 3: Pseudocode of pairwise-parallel key mismatch attacks on Kyber1024

```
Input: Parallel level $P$ and multi-value key mismatch oracle $\mathcal{O}$
Output: $\mathbf{s}'_A$
1.  $(\mathbf{s}_A, \mathbf{P}_A) \leftarrow$ Kyber.CPAPKE.KeyGen();
2.  Set $queries = 0$;
3.  for (i=0; i<4; i++) do
4.      for (k=0; k<256/2; k+=$P$) do
5.          for (r=0; r<5; r++) do
6.              $\mathbf{P}_B = \mathbf{0}$
7.              if $k = 0$ then $\mathbf{P}_B[0] = b_1\lceil \frac{q}{32} \rfloor$, $\mathbf{P}_B[128] = b_2\lceil \frac{q}{32} \rfloor$;
8.              else $\mathbf{P}_B[256-k] = -b_1\lceil \frac{q}{32} \rfloor$, $\mathbf{P}_B[128-k] = -b_2\lceil \frac{q}{32} \rfloor$;
9.              $\mathbf{c_1} = \mathbf{Compress}_q(\mathbf{P}_B, d_{\mathbf{P}_B})$;
10.             $\mathbf{c_2} = \mathbf{0}$ except $\mathbf{c_2}[k+i] = h_i,\ 0 \leq i < P$;
11.             $\mathbf{m}' =$ Kyber.CPAPKE.Dec$(\mathbf{c_1}, \mathbf{c_2})$;
12.             $K_A = H(\mathbf{m}'||(\mathbf{P}_B, \mathbf{c_1}, \mathbf{c_2}))$;
13.             $\mathbf{c}_A = Enc(K_A, \mathbf{Target})$;
14.             $\mathbf{m}' = \mathcal{O}(P, \mathbf{c}_A, \mathbf{Target}, \mathbf{P}_B, \mathbf{c_1}, \mathbf{c_2})$;
15.             $queries$ ++;
16.             Update $h_i, b_1, b_2$ based on Table 2;
17.         end;
18.         for (j=0; j<$P$; j++) do
19.             Output $\mathbf{s}'_A[i * 256 + k + j]$ based on Table 3;
20.         end;
21.     end;
22. end ;
```

5.2 Results and Comparisons

In this section, we show the experiment results , and compare with the state-of-the-art key mismatch attacks in [20, 10]. We consider three different parallel levels $P = 8/16/26$, corresponding to the computational complexity options $2^{16}/2^{23}/2^{32}$, and denote the number of required queries as Our Result 1/2/3 respectively. We present the results in Table 5.

For CPA-secure Kyber, we recover the full secret key with 100% success probability. The results show our attacks can reduce the number of queries by 85% with computational complexity 2^{16}, 92% with computational complexity 2^{23}, and 95% with computational complexity 2^{32}.

And then we simulate the symmetric encryption/decryption process using *AES256-ECB* to realize the multi-value key mismatch oracle. When parallel

Table 5: Our results compared with state-of-the-art key mismatch attacks.

	Result in [20]	Result in [10]	Our Result 1	Our Result 2	Our Result 3
Kyber512	1312	1205	192	96	60
Kyber768	1776	1589	240	120	75
Kyber1024	2368	2118	320	160	100

level $P = 8$, an average of 131 decryption operations are needed to realize an 8-value key mismatch oracle. For Kyber1024, recovering the full secret key needs 320 queries, costing 1.5 seconds in total. Besides, when parallel level $P = 16$, an average of 34310 decryption operations are needed to realize a 16-value key mismatch oracle. For Kyber1024, recovering the full secret key needs 160 queries, costing 152.8 seconds in total.

6 Discussions

6.1 Post-Processing with Lattice Reduction

The above complexity analysis is the number of queries required to recover the full secret key. Actually, the adversary can only recover partial coefficients, then recover the remaining coefficients using lattice reduction in an offline manner.

Based on the research of Dachman-Soled et al. [6], when the adversary obtains a single coefficient of $\mathbf{s}_A$ (i.e.) $\mathbf{s}_A[0]$, he can get a *perfect hint* in the form of $\langle \mathbf{s}, \mathbf{v} \rangle = \mathbf{s}_A[0]$. Here $\mathbf{s} = (\mathbf{s_A}, \mathbf{e})$, and $\mathbf{v}$ represents a vector with all elements being 0, except the first coefficient, which is set to 1. By incorporating this perfect hint into a lattice reduction algorithm, the lattice's dimension is reduced by one, and its volume increases by a factor of $\sqrt{1 + \mathbf{s}_A^2[0]}$. Consequently, the cost of a standard lattice attack is lowered. Additionally, the design of the schemes allows the extraction of a *short vector hint*. Specifically, the so-called q-vector $(q, 0, 0, ..., 0)$ and its permutations. Integrating this hint into a lattice reduction algorithm also reduces the computational cost of the lattice attack.

For Kyber1024, we use *leaky LWE estimator* in [6] to estimate the lattice reduction cost, and plot the relationship with the number of queries for different parallel level P in Fig. 7. We see that the number of queries can be greatly reduced by using lattice reduction. For instance, when $P = 26$, we only need 78 queries to recover the full secret with a lattice reduction cost of 2^{32}.

6.2 Improved PC Oracle-based SCA on CCA-secure Kyber

As described above, the CPA-secure Kyber is susceptible to pairwise-parallel key mismatch attacks. However, the NIST candidate KEMs always achieve CCA-security through the use of FO transformation. The decryption and re-encryption process in the FO transformation guarantees the validity of the ciphertext, returning failure when an invalid ciphertext is detected. Consequently, Alice consistently rejects these malicious chosen-ciphertexts, preventing Bob from extracting any meaningful information from Alice's responses.

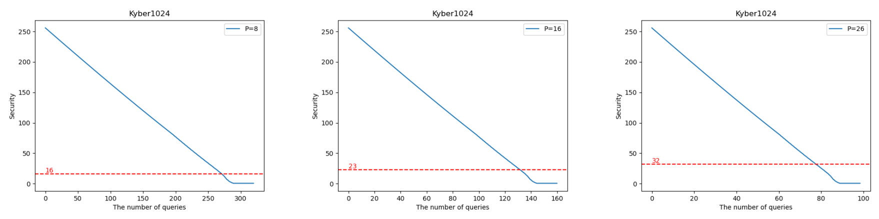


Fig. 7: The lattice reduction cost in $\log_2()$ vs. the number of queries. The horizontal line represent the lattice reduction cost same with the computational complexity under the parallel level P.

However, at CHES 2020, Ravi et al. [22] showed that chosen ciphertext attacks on CCA-secure NIST candidate KEMs can be executed by leveraging side-channel information. The adversary can construct the required PC oracle using side-channel information leakage from re-encryption. Recently, Rajendran et al. [23] and Tanaka et al. [26] proposed a parallel PC oracle-based SCA, which can recover P number of bits of information about the secret key in a single query/trace. Based on simple analysis, our *pairwise-parallel* attack method can be directly employed to further enhance the efficiency of these attacks.

We compare the number of queries between PC oracle-based SCA in [23, 26] and our proposed method. The results are presented in Table 6. Specifically, for Kyber768/1024, the parallel PC oracle-based SCA requires 6 queries to recover $2P$ number of coefficients. In contrast, our method achieves the same result with only 5 queries, resulting in a reduction of approximately 16.67%.

Table 6: The number of queries required for full secret key recovery in [26], [23] and ours.

	P	2	4	6	8	10	12	14	16
Kyber768	Tanaka et al. [26]	1152	576	384	288	234	198	165	144
	Ravi et al. [23]	970	533	373	283	232	197	171	144
	Our results	960	480	320	240	195	165	137	120
Kyber1024	Tanaka et al. [26]	1536	768	512	384	312	264	219	192
	Ravi et al. [23]	1294	711	498	378	310	263	228	192
	Our results	1280	640	427	320	260	220	183	160

6.3 Applicability to other Lattice-based KEMs

While we only present our pairwise-parallel key mismatch attacks on Kyber, we believe that our attacks can be adapted to other lattice-based KEMs such as Saber [4] and other LPR encryption schemes [12]. We briefly sketch the idea to perform pairwise-parallel key mismatch attacks on Saber and present the cuts diagram in appendix A.2.

Saber is based on the hardness of the MLWR, where $n = 256$ and the rank of the module $k = 3$. For the secret key in Saber, the coefficients are in a range of $[-4, 4]$. According to the binary search, recovering one coefficient at most requires 4 queries as $2^3 < 9 < 2^4$. Thus the number of queries required for full key recovery is $4 \cdot 256 \cdot 3 = 3000$. For secret key coefficient-pair $(s_A[i], s_A[i + 128])$, there are $9 \times 9 = 81$ possible values. In pairwise-parallel key mismatch attacks, recovering P number of coefficient-pairs at most requires 7 queries as $2^6 < 81 < 2^7$. Thus the number of queries required for full key recovery is $7 \cdot \lceil 256/2P \rfloor \cdot 3$. When $P = 26$, the number of required queries is 105, reducing by 96.5%. Similarly, we believe our attack can also be adapted to other LWE/LWR-based KEMs.

7 Conclusion

In this paper, we propose *pairwise-parallel key mismatch attacks* on CPA-secure Kyber, significantly enhancing the key mismatch attacks. Our attacks can recover partial information about P number of coefficient-pairs of the secret key in a parallel way per query. And we realize the required multi-value key mismatch oracle in a simple CPA-secure KEM-based key exchange scenario. Then we experimentally validated our attacks on the C reference implementations of Kyber. The results show we can recover the full secret key within 100 queries for Kyber1024, reducing the number of queries by 95% with computational complexity 2^{32}, when parallel level $P = 26$.

Then we further reduce the number of queries required for full secret key recovery using the post-processing of lattice reduction. When $P = 26$, we only need 78 queries to recover the full secret key with a lattice reduction cost of 2^{32}. Besides, our pairwise-parallel attacks method can also be applied to enhance the efficiency of PC oracle-based SCA on CCA-secure KEMs. Specifically, compared with parallel PC oracle-based attacks on Kyber768/1024 in [23, 26], our attacks can reduce the number of queries/traces by 16.67%. Finally we discuss the applicability of our attacks on other lattice-based KEMs. We believe our attacks can be adapted to similar LWE/LWR-based KEMs such as Saber etc.

Key mismatch attacks provide valuable insights into the resilience of these KEMs against key reuse. These findings emphasize the critical importance of preventing key reuse for CPA-secure KEMs and the necessity of safeguarding against SCA in the case of CCA-secure KEMs.

Acknowledgements This work is supported by National Key R&D Program of China (No.2022YFB3103800), National Natural Science Foundation of China (No.U1936209, No.62002353, No.62202231, No.62202230), China Postdoctoral Science Foundation (No.2021M701726), Jiangsu Funding Program for Excellent Postdoctoral Talent (No.2022ZB270) and Yunnan Provincial Major Science and Technology Special Plan Projects (No.202103AA080015).

References

1. Băetu, C., Durak, F. B., Huguenin-Dumittan, L., Talayhan, A., Vaudenay, S.: Misuse attacks on post-quantum cryptosystems. In: Advances in Cryptology–EUROCRYPT 2019, pp. 747-776. Springer (2019)
2. Bauer, A., Gilbert, H., Renault, G., Rossi, M.: Assessment of the key-reuse resilience of NewHope. In: The Cryptographers' Track at the RSA Conference, pp. 272-292. Springer (2019)
3. Ding, J., Fluhrer, S., Rv, S.: Complete attack on RLWE key exchange with reused keys, without signal leakage. In: Information Security and Privacy, pp. 467-486. Springer (2018)
4. D'Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F., Mera, J.M.B., Beirendonck, M.V., Basso, A.: SABER. Tech. rep., National Institute of Standards and Technology (2020)
5. Dierks, T., Rescorla, E.: The transport layer security (TLS) protocol version 1.2 (No. rfc5246) (2008)
6. Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with side information: attacks and concrete security estimation. In Advances in Cryptology–CRYPTO 2020, Part II, pp. 329-358. Springer (2020)
7. D'Anvers, J. P., Tiepelt, M., Vercauteren, F., Verbauwhede, I.: Timing attacks on error correcting codes in post-quantum schemes. In: Proceedings of ACM Workshop on Theory of Implementation Security Workshop, pp. 2-9. (2019)
8. Fluhrer, S.: Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085 (2016)
9. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Advances in Cryptology-CRYPTO 1999, pp. 537-554. Springer (1999)
10. Guo, Q., Mårtensson, E.: Do not bound to a single position: Near-Optimal multi-positional mismatch attacks against Kyber and Saber. Cryptology ePrint Archive, Report 2022/983 (2022)
11. Kirkwood, D., Lackey, B. C., McVey, J., Motley, M., Solinas, J. A., Tuller, D.: Failure is not an option: Standardization issues for post-quantum key agreement. In: Workshop on Cybersecurity in a Post-Quantum World (2015)
12. Lyubashevsky, V., Peikert, C., Regev, O: On ideal lattices and learning with errors over rings. In: Advances in Cryptology–EUROCRYPT 2010, pp. 1–23. Springer (2010)
13. Moody, D., Alagic, G., Apon, D., Cooper, D., Dang, Q., Kelsey, J., Liu, Y., Miller, C., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D., Alperin-Sheriff, J.: Status report on the second round of the NIST Post-Quantum cryptography standardization process, NIST (2020)
14. Mi, R., Jiang, H., Zhang, Z.: Lattice reduction meets key-mismatch: New misuse attack on Lattice-based NIST candidate KEMs. Cryptology ePrint Archive, Report 2022/1064 (2022)
15. Okada, S., Wang, Y., Takagi, T.: Improving key mismatch attack on NewHope with fewer queries. In: Information Security and Privacy, pp. 505-524. Springer (2020)
16. Alkim, E., Avanzi, R., Bos, J., Ducas, L., De La Piedra, A., Pöppelmann, P. S. T., Stebila, D.: NewHope, Submission to round 1 of NIST Post Quantum cryptography competition (2017)

17. Peikert, C.: Lattice cryptography for the internet. In: Post-Quantum Cryptography, pp. 197-219. Springer (2014)
18. Qin, Y., Cheng, C., Ding, J.: A complete and optimized key mismatch attack on NIST candidate NewHope. In: 24th European Symposium on Research in Computer Security, pp. 504-520. Springer (2019)
19. Qin, Y., Cheng, C., Ding, J.: An efficient key mismatch attack on the NIST second round candidate Kyber. Cryptology ePrint Archive, Report 2019/1343 (2019)
20. Qin, Y., Cheng, C., Zhang, X., Pan, Y., Hu, L., Ding, J.: A systematic approach and analysis of key mismatch attacks on lattice-based NIST candidate KEMs. In: Advances in Cryptology–ASIACRYPT 2021, pp. 92-121. Springer (2021)
21. Rescorla, E.: The transport layer security (TLS) protocol version 1.3 (No. rfc8446) (2018)
22. Ravi, P., Roy, S. S., Chattopadhyay, A., Bhasin, S.: Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(3), pp. 307-335. (2020)
23. Rajendran, G., Ravi, P., D'Anvers, J. P., Bhasin, S., Chattopadhyay, A.: Pushing the limits of generic side-channel attacks on LWE-based KEMs-parallel PC oracle attacks on Kyber KEM and beyond. IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 418-446. (2023)
24. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Seiler, G., Stehlé, D.: CRYSTALS-KYBER. Tech. rep., National Institute of Standards and Technology (2022)
25. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of 35th Annual Symposium on Foundations of Computer Science 1994, pp. 124–134. IEEE (1994)
26. Tanaka, Y., Ueno, R., Xagawa, K., Ito, A., Takahashi, J., Homma, N: Multiple-valued plaintext-checking side-channel attacks on post-quantum KEMs. Cryptology ePrint Archive, Report 2022/940 (2022)
27. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/em analysis on post-quantum kems. IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 296-322. (2022)

A Pairwise-Parallel Key Mismatch Attacks

A.1 Pairwise-Parallel Key Mismatch Attacks on Kyber768/1024

In this section, we give a feasible parameters selection scheme, so that the P number of coefficient-pairs can be determined within 5 cuts for Kyber768/1024.

At the first step, we choose a triangular cut by setting $b_1 = 2, b_2 = -1$. Thus $\mathbf{m}'[i] = \left\lceil \frac{2}{q} \left(\left\lceil \frac{q}{32} h_i \right\rfloor - (2\mathbf{s}_A^T[i] + \mathbf{s}_A^T[128+i]) \left\lceil \frac{q}{32} \right\rfloor \right) \right\rfloor mod\ 2$. When $h_i = 8$, $\mathbf{m}'[i] = 1$ only if $(2\mathbf{s}_A^T[i] + \mathbf{s}_A^T[128+i]) \leq -1$, then we cut the two-dimensional grid into two regions, containing 11 and 14 values respectively as Fig. 8.

At the second step, we choose a horizontal cut by setting $b_1 = 0, b_2 = -1$. Thus $\mathbf{m}'[i] = \left\lceil \frac{2}{q} \left(\left\lceil \frac{q}{32} h_i \right\rfloor - \mathbf{s}_A^T[128+i] \left\lceil \frac{q}{32} \right\rfloor \right) \right\rfloor mod\ 2$. For the region $\mathbf{m}'[i] = 1$ in the first cut, we set $h_i = 8$, and for another region, $h_i = 9$ as Fig. 9.

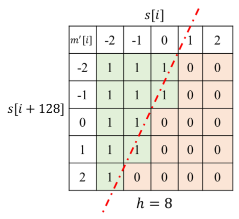


Fig. 8: Step1. $b_1 = 2, b_2 = -1$.

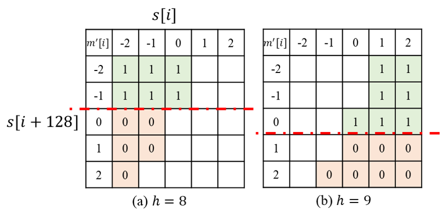


Fig. 9: Step2. $b_1 = 0, b_2 = -1$.

At the following three steps, we set different parameters b_1, b_2, h_i according to the previous parameters selection and the results of $\mathbf{m}'[i]$.

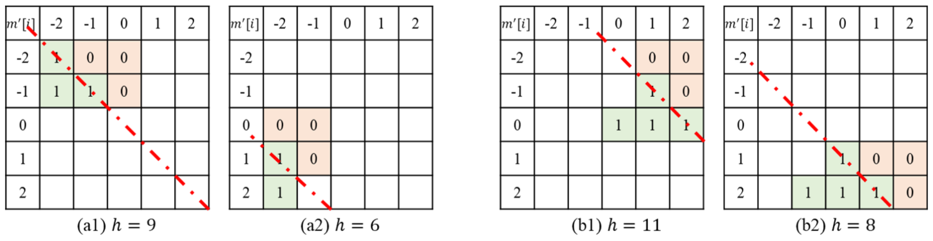


Fig. 10: Step3. $b_1 = 1, b_2 = 1$.

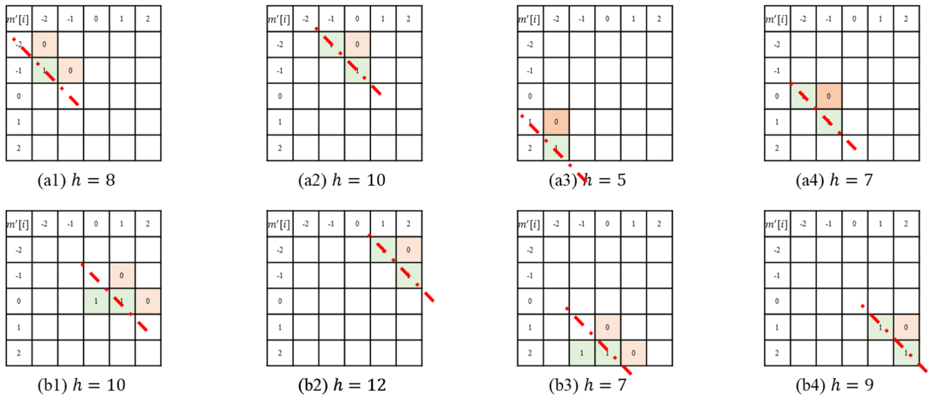


Fig. 11: Step4. $b_1 = 1, b_2 = 1$.

After five rounds of cuts, the value of $(\mathbf{s}_A[i], \mathbf{s}_A[128 + i])$ can be uniquely determined.

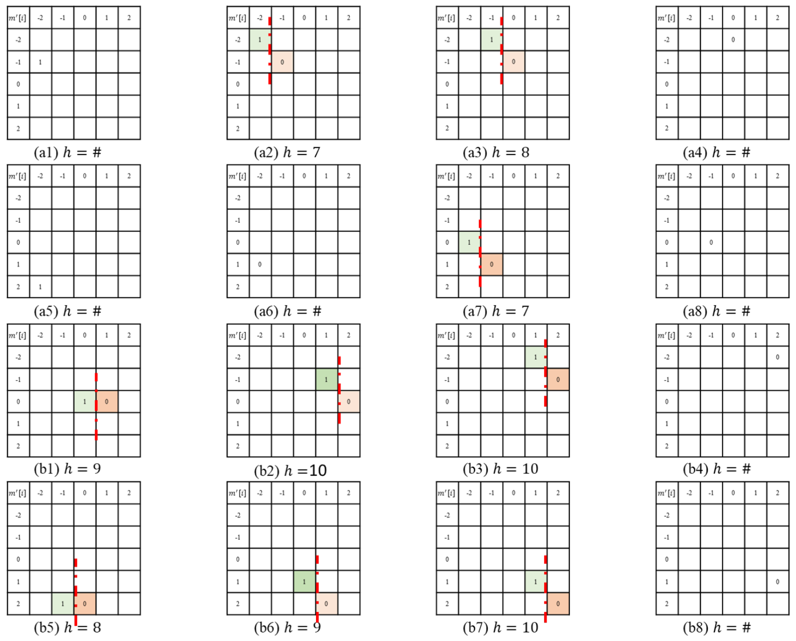


Fig. 12: Step5. $b_1 = 1, b_2 = 0$.

A.2 Pairwise-Parallel Key Mismatch Attacks on Kyber512/Saber

In this section, we give feasible parameters selection schemes for Kyber512 ($\eta = 3$) and Saber ($\eta = 4$), so that we can recover P number of coefficient-pairs within 6 cuts and 7 cuts respectively.

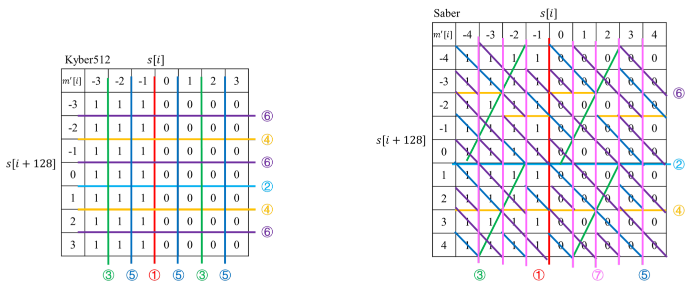


Fig. 13: Pairwise-parallel key mismatch attack on Kyber512 and Saber.