Do Not Bound to a Single Position: Near-Optimal Multi-Positional Mismatch Attacks Against Kyber and Saber

Qian Guo[1] and Erik Mårtensson[2,1]

[1] Dept. of Electrical and Information Technology, Lund University, Sweden
{qian.guo,erik.martensson}@eit.lth.se
[2] Selmer Center, Department of Informatics, University of Bergen, Norway
erik.martensson@uib.no

Abstract. Misuse resilience is an important security criterion in the evaluation of the NIST Post-quantum cryptography standardization process. In this paper, we propose new key mismatch attacks against Kyber and Saber, NIST's selected scheme for encryption and one of the finalists in the third round of the NIST competition, respectively. Our novel idea is to recover partial information of multiple secret entries in each mismatch oracle call. These multi-positional attacks greatly reduce the expected number of oracle calls needed to fully recover the secret key. They also have significance in side-channel analysis.
From the perspective of lower bounds, our new attacks falsify the Huffman bounds proposed in [Qin et al. ASIACRYPT 2021], where a one-positional mismatch adversary is assumed. Our new attacks can be bounded by the Shannon lower bounds, i.e., the entropy of the distribution generating each secret coefficient times the number of secret entries. We call the new attacks "near-optimal" since their query complexities are close to the Shannon lower bounds.

Keywords: Lattice-based cryptography · Mismatch attacks · LWE · LWR · Kyber · Saber.

1 Introduction

Post-quantum cryptography (PQC) has become essential due to the rapid advances in building quantum computers. In post-quantum cryptography, researchers investigate new cryptographic primitives that resist quantum attacks (e.g. Shor's algorithm [41]) even when large-scale quantum computers exist. There are five main branches in post-quantum cryptography, lattice-based, code-based, multivariate-based, isogeny-based, and symmetric-based cryptography. Lattice-based cryptography [3,39] is arguably the most promising of them all.

In 2016, NIST (National Institute of Standards and Technology, U.S. Department of Commerce) started a standardization process [1], referred to as the NIST PQC project, to solicit new quantum-resistant public-key cryptographic standards. Recently the fourth round began and the Learning With Errors (LWE)-based KEM Kyber [40] was chosen for the Public Key Encryption (PKE)/Key

Encapsulation Mechanism (KEM) category. Other than Kyber the third round included two more lattice-based PKE/KEM finalists, the Learning With Rounding (LWR)-based KEM Saber [16], as well as NTRU [12]. The last finalist in the third round was the code-based KEM Classic McEliece [4].

NIST defines five security levels, from NIST-I to NIST-V, characterizing the required security level by the computational effort of key-search on a block cipher (or of collision search on a hash function). For instance, the NIST-I parameter set should resist any attack requiring computational resources less than those for key-search on AES128. In the call-for-proposal [2], NIST also listed additional desirable security properties, and among them, side-channel and keypair-reuse resiliences are two attractive properties requiring extensive further investigation.

The focal point of this paper is a specific type of keypair-reuse attacks called mismatch attacks [19], and we mainly target the LWE/LWR-based KEMs Kyber and Saber. In such a mismatch attack, one communicating party's public key is reused and the adversary will impersonate the other party and recover the secret key by repeatedly checking whether the two shared keys match. There is a long list of known mismatch attacks on the lattice-based KEMs, e.g. [36,7,6,35,24,37]. The main problem related to mismatch attacks in the NIST PQC project is evaluating the candidate's key reuse resilience, i.e., how many key reuses can be tolerated before the full secret key is recovered. Mismatch attacks are not only significant when evaluating key reuse resilience, but also in side-channel and fault analysis, because mismatch attacks are closely connected with certain types of chosen-ciphertext side-channel attacks [37,18,26,38,42] and a fault-injection attack [43]. The authors in [37] proposed a generic method of transforming the problem of finding optimal mismatch attacks to finding an optimal binary recovery tree (BRT), and obtained the optimal BRT and the corresponding lower bounds by Huffman coding.

However, the lower bounds in [37] are derived under the assumption that the adversary only can recover partial information of one secret entry in each mismatch oracle call. Thus, their bounds could be invalid for general adversaries who can recover partial secret information related to multiple positions. However, it is challenging *to design a better attacking strategy to constructively beat the lower bound proposed in [37].*

1.1 Related Work

Chosen-ciphertext attacks (CCA) on IND-CPA secure schemes can be traced back to Bleichenbacher's attacks on the RSA PKCS#1 [11]. In 1999, Hall, Goldberg, and Schneier [29] proposed the reaction attack model that checks if the decryption is successful or not. This model is a weaker model than the CCA model, and the reaction attacks of [29] can be used to recover messages for code-based schemes like McEliece [33] and private keys for early lattice-based schemes such as the Ajtai-Dwork [3] and the GGH [23] cryptosystems. Hoffstein and Silverman [31] extended these attacks to NTRU [30].

These attacks can be protected against by using CCA transforms, such as the famous Fujisaki-Okamoto (FO) transform [22]. Numerous works [27,20,15,28,17,25,10]

Table 1: Sample complexity of the new attack **v.s.** lower bounds.

	Shannon Bound(b_1)	Lower Bound (b_2) from [37]	Previous Best(n_{pb}) n_{pb} from [37]	n_{pb}/b_1	n_{pb}/b_2	New Attack(n_{na}) n_{na}	n_{na}/b_1	n_{na}/b_2
Kyber512	1195	1216	1312	1.098	1.079	1205	1.008	**0.991**
Kyber768	1560	1632	1776	1.138	1.088	1588	1.018	**0.973**
Kyber1024	2079	2176	2368	1.139	1.088	2118	1.019	**0.973**
LightSaber	1386	1412	1460	1.053	1.033	1410	1.017	**0.998**
Saber	1954	1986	2091	1.071	1.053	1985	1.015	**0.999**
FireSaber	2389	2432	2624	1.098	1.079	2411	1.009	**0.991**

in lattice-based and code-based cryptography demonstrate that the CCA protection can fail if the decryption error rate is not sufficiently small.

In 2016, Fluhrer [21] initiated key-reuse attacks against lattice-based encryption. Later, Ding, Fluhrer, and Saraswathy [19] extended the attacks to lattice-based key exchange and proposed a key mismatch attack. Similar attacks can be applied to many lattice-based KEMs and the query complexities are further improved in [8,7,36,35,24,32].

Regarding the lower bounds on the average number of queries needed to recover the full secret key in a mismatch attack, in EUROCRYPT 2019, Băetu et al. [6] proved that this number should be larger than the Shannon entropy of the secret distribution. This lower bound is referred to as the Shannon bound in this paper. In ASIACRYPT 2021, Qin et al. [37] proposed sharper lower bounds from Huffman coding theory for CPA-secure lattice-based KEMs and also presented improved constructive results with query complexities close to the proposed Huffman coding lower bounds.

1.2 Contributions

In this paper, we propose novel mismatch attacks against Kyber and Saber that beat the Huffman coding lower bounds proposed in [37], i.e., we falsify their lower bounds by providing better constructive results. Our techniques include novel attacking strategies that can recover partial information about multiple coefficients of the secret key in each query. We name the new attacks *multi-positional mismatch attacks*. The main contributions of the paper are the following.

1. We first study two-positional attacks on Kyber and Saber. We propose new methods to split the two-dimensional plane for two secret coefficients and decide from the mismatch oracle call which part the two coefficients belong to. We propose various splitting approaches and transform the problem of finding the most efficient splits into an optimization problem. For Kyber512, Kyber768, Kyber1024, and FireSaber we search for the best two-positional attacks manually. When the possible pairs of secret coefficients are too large, e.g. in the cases of LightSaber and Saber, we design a greedy algorithm to automatically solve the optimization problem. This greedy approach is

extended to attacking more than two positions at a time.
In Table 1 we present the new query complexities, which are compared with the Shannon bounds and the one-dimensional Huffman bounds proposed in [37]. Firstly, our new methods beat the lower bounds in [37] for all the parameter sets of Kyber and Saber. Secondly, the new attacks significantly improve the query complexities, e.g., reducing 107 queries for Kyber512 and 250 for Kyber1024, compared to the attacks in [37]. We call the attacks "near-optimal" since their query complexities are close to the Shannon bound. For instance, for Kyber512, the constructive result is only larger than the Shannon bound by 10 queries (or by a factor of 0.8 %).

2. We further employ the lattice estimator, a new version of the widely-used LWE estimator [5] to roughly estimate the query sample complexity when a certain amount (e.g., 2^{60}) of post-processing with lattice reduction is allowed. Our new multi-positional mismatch attacks also improve the query complexity in this scenario. Though it has been remarked in [37] that the derived lower bounds are invalid when post-processing is allowed, we present the first quantitative analysis of mismatch attacks with post-processing.
3. Last, our new multi-positional mismatch attacks can improve the efficiency of side-channel attacks on CCA-secure implementations of Kyber and Saber. The new attack strategy may also be applied to improve the fault-injection attack in [43].

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we present the necessary background including the CPA-secure versions of Kyber and Saber, the model of mismatch attacks, and the Huffman coding. In Section 3, we present the state-of-the-art mismatch attacks on Kyber and Saber proposed in [37]. We present our multi-positional attacks, that constructively beat the lower bounds from [37], in Section 4. This is followed by more discussions about using post-processing to further reduce the query complexity and the connection to side-channel and fault-injection analysis in Section 5. We conclude this paper and present possible future works in Section 6.

2 Background

In this section, we introduce CPA-secure instantiations of the KEMs Kyber and Saber. Note that in the official documents of Kyber and Saber, the CPA-secure versions are limited to ephemeral keys, but this constraint may be ignored in practice. We thus create these CPA-secure instantiations to assess their key reuse resilience. Our notations and terminology are similar to the ones in [37].

- Let $\mathbf{H}(\cdot)$ be a hash function and $\mathbf{x}||\mathbf{y}$ be the concatenation of two strings $\mathbf{x}$ and $\mathbf{y}$.
- The symbol $\leftarrow_{\$}$ denotes sampling from a distribution.

Alice		**Bob**
1. Generate matrix $\mathbf{a} \in \mathcal{R}_q^{l \times l}$		
$\mathbf{s}_A, \mathbf{e}_A \leftarrow_\$ \mathbf{B}_\eta^l$		2. $\mathbf{m} \leftarrow_\$ \{0,1\}^{256}$
$\mathbf{P}_A \leftarrow \mathbf{a} \circ \mathbf{s}_A + \mathbf{e}_A$		Generate matrix $\mathbf{a} \in \mathcal{R}_q^{l \times l}$
Output: $(\mathbf{s}_A, \mathbf{P}_A)$	$\xrightarrow{\mathbf{P}_A}$	$\mathbf{s}_B \leftarrow_\$ \mathbf{B}_\eta^l, \mathbf{e}_B \leftarrow_\$ \mathbf{B}_{\eta'}^l, \mathbf{e}'_B \leftarrow_\$ \mathbf{B}_{\eta'}$
		$\mathbf{P}_B \leftarrow \mathbf{a} \circ \mathbf{s}_B + \mathbf{e}_B$
		$\mathbf{v}_B \leftarrow \mathbf{P}_A^{\mathsf{tr}} \circ \mathbf{s}_B + \mathbf{e}'_B + \mathbf{Decompress}_q(\mathbf{m}, 2)$
3. $\mathbf{u}_A \leftarrow \mathbf{Decompress}_q(\mathbf{c}_1, 2^{d_{\mathbf{P}_B}})$		$\mathbf{c}_1 \leftarrow \mathbf{Compress}_q(\mathbf{P}_B, 2^{d_{\mathbf{P}_B}})$
$\mathbf{v}_A \leftarrow \mathbf{Decompress}_q(\mathbf{c}_c, 2^{d_{\mathbf{v}_B}})$	$\xleftarrow{\mathbf{P}_B, \mathbf{c}_1, \mathbf{c}_2}$	$\mathbf{c}_2 \leftarrow \mathbf{Compress}_q(\mathbf{v}_B, 2^{d_{\mathbf{v}_B}})$
$\mathbf{m}' \leftarrow \mathbf{Compress}_q(\mathbf{v}_A - \mathbf{s}_A^{\mathsf{tr}} \circ \mathbf{u}_A, 2)$		$K_B \leftarrow \mathbf{H}(\mathbf{m} \| (\mathbf{P}_B, (\mathbf{c}_1, \mathbf{c}_2)))$
$K_A \leftarrow \mathbf{H}(\mathbf{m}' \| (\mathbf{P}_B, (\mathbf{c}_1, \mathbf{c}_2)))$		

Fig. 1: The CPA-secure version of Kyber

- Let $\mathbf{A}^{\mathsf{tr}}$ denote the transpose of the matrix $\mathbf{A}$.
- The distribution $\mathbf{B}_\eta$ is the central binomial distribution whose output is computed as $\sum_{i=1}^{\eta}(a_i - b_i)$, where a_i and b_i are independently and uniformly randomly sampled from $\{0,1\}$.

2.1 CPA-Secure Version of Kyber

Kyber [40] is the KEM proposal of CRYSTALS (Cryptographic Suite for Algebraic Lattices), based on the module learning with errors (MLWE) problem. In the fourth round NIST has selected Kyber as their scheme for PKE/KEM. Following the work of [37], we describe a potential instantiation of the CPA-secure Kyber KEM in Figure 1 by invoking the functions of Kyber.CPAPKE from [40].

Let $\mathcal{R}_q$ be the polynomial ring $\mathbb{Z}_q[x]/(x^n+1)$, where $q = 3329$ and $n = 256$, and let $\circ$ (+ or −) be the corresponding multiplication (addition or subtraction) in the ring. Let l denote the rank of the module, which is set to be $2, 3$, and 4, for the three different versions, Kyber512, Kyber768, and Kyber 1024. Alice and Bob generate a matrix $\mathbf{a}$ from a public seed by calling a pseudorandom function. As shown in Figure 1, round-3 Kyber employs two central binomial distributions $\mathbf{B}_\eta$ and $\mathbf{B}_{\eta'}$. The designers pick $(\eta, \eta') = (3, 2)$ for Kyber512, and $(\eta, \eta') = (2, 2)$ for Kyber768 and Kyber1024. The $\mathbf{Compress}_q(x, p)$ function transforms x from module q to module p by

$$\mathbf{Compress}_q(x, p) = \lceil x \cdot p/q \rfloor \mod^+ p,$$

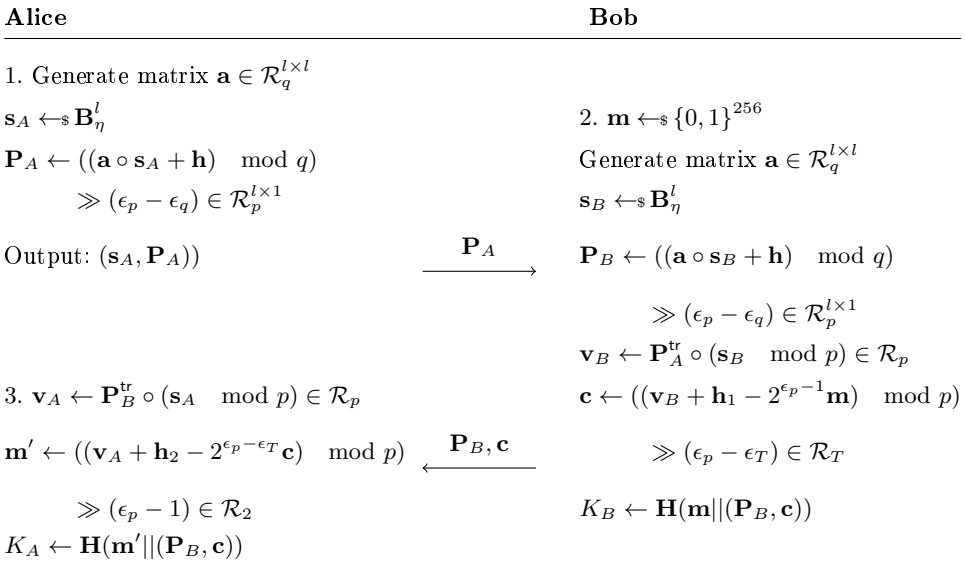

Fig. 2: The CPA-secure version of Saber

where $r' = r \mod^+ p$ represents the unique element r' in the range $-\frac{p}{2} < r' \leq \frac{p}{2}$ such that $r' \equiv r \pmod p$. We also define the inverse function

$$\mathbf{Decompress}_q(x, p) = \lceil x \cdot q/p \rfloor.$$

If the first input to the function $\mathbf{Compress}_q(x, p)$ (or $\mathbf{Decompress}_q(x, p)$) is a vector/polynomial, then we call the function coefficient-wise.

2.2 CPA-Secure Version of Saber

Saber [16] was a finalist candidate in the third round of the NIST PQ project, whose security is based on the hardness of the Module Learning with Rounding problem (MLWR). Similar to the CPA-secure version of the Kyber KEM, in Figure 2 we present a possible instantiation of a CPA-secure Saber KEM by invoking the functions of Saber.PKE from [16].

We use the same notations as in Section 2.1, e.g. $\mathcal{R}_q$ denotes the polynomial ring $\mathbb{Z}_q[x]/(x^n + 1)$, where $n = 256$ and the rank of the module l is set to be 2 for LightSaber, 3 for Saber, and 4 for FireSaber. The matrix $\mathbf{a}$ is also generated from a public seed. The secret coefficients are generated from the central binomial distribution $\mathbf{B}_\eta$, where η is 5 for LightSaber, 4 for Saber, and 3 for FireSaber, respectively. Saber chooses three positive integers q, p, and T as powers of 2, i.e., $q = 2^{\epsilon_q}$, $p = 2^{\epsilon_p}$, and $T = 2^{\epsilon_T}$, respectively. In round-3 Saber, $\epsilon_q = 13$ and $\epsilon_p = 10$. The exponent ϵ_T is set to be 3 for LightSaber, 4 for Saber, and 6 for FireSaber, respectively.

We denote the bitwise right shift operation by $\gg$ and apply it to polynomials and matrices by calling it coefficient-wise. Saber also introduces two constant polynomials $\mathbf{h}_1 \in \mathcal{R}_q$ and $\mathbf{h}_2 \in \mathcal{R}_q$ with all coefficients set to $2^{\epsilon_q-\epsilon_p-1}$ and $2^{\epsilon_p-2} - 2^{\epsilon_p-\epsilon_T-1} + 2^{\epsilon_q-\epsilon_p-1}$, respectively, and one constant vector $\mathbf{h} \in \mathcal{R}_q^{l\times 1}$ with each polynomial set to $\mathbf{h}_1$.

2.3 Mismatch Attack Model

In a key mismatch attack, we assume that Alice reuses her keypair $(\mathbf{s}_A, \mathbf{P}_A)$ and the adversary Eve impersonates Bob to recover Alice's secret key $\mathbf{s}_A$ by communicating with Alice. We can build an oracle to simulate the decapsulation of Alice with input including the pair $(\mathbf{P}_B, \mathbf{c})$ chosen by Eve and the corresponding shared key K_B. Here we denote $\mathbf{c}_1, \mathbf{c}_2$ by $\mathbf{c}$ for Kyber. The oracle denoted $\mathcal{O}$ calls Alice's decapsulation function and obtains the shared key K_A. It outputs 1 if the shared keys $K_A = K_B$ and 0 otherwise. The aim of a mismatch attack is to recover Alice's key by selecting the chosen pairs of the form $(\mathbf{P}_B, \mathbf{c})$ and iteratively querying the oracle $\mathcal{O}$.

2.4 Huffman Coding

Huffman coding refers to an algorithm that finds an efficient binary code used for lossless data compression. Given a symbol-by-symbol encoding of strings with independent and identically distributed symbols from a known distribution, Huffman coding creates an optimal code. Huffman coding works by iteratively building a binary tree from the bottom up by merging the two least probably symbols into a new symbol. Basic (one-dimensional) Huffman coding can be generalized to n dimensions and improved by considering each possible n-tuple of symbols from an alphabet as a new symbol and applying Huffman coding to these n-tuples. We refer the reader to a book on information theory (e.g., [13]) for more details.

3 One-Positional Mismatch Attacks

In a mismatch attack, Eve impersonates Bob and wants to recover Alice's secret key $\mathbf{s}_A$. As we can see in Figure 1 and 2, given the pair $(\mathbf{P}_B, \mathbf{c})$, the keys K_A and K_B match if and only if Alice's computed $\mathbf{m}'$ matches Eve's chosen $\mathbf{m}$. Thus, for each query of the oracle Eve sets her parameters $\mathbf{m}$ and $(\mathbf{P}_B, \mathbf{c})$ such that the output of the oracle teaches her something about $\mathbf{s}_A$.

Let us explain in some detail how Eve's attack works when retrieving one position at a time and let us first focus on position 0. Let s_i denote the value of $\mathbf{s}_A$ on index i, when the subscript A is implied. Eve creates the message $\mathbf{m} = [1, 0, \ldots, 0]$. She sets her parameters $(\mathbf{P}_B, \mathbf{c})$ such that Alice's received message is 0 by design on all positions except for the position 0, whose value depends on the secret value s_0. By observing the output of the oracle, Eve learns some information about s_0. Repeating the process, Eve learns the exact value of s_0 and then continues the process to learn the other s_i values.

3.1 Kyber

Let us now describe the attack in some more detail, for Kyber1024[3]. Eve lets $\mathbf{P}_B = [\lceil \frac{q}{32} \rfloor, 0, \ldots, 0]$. She calculates $\mathbf{c}_1 = \mathbf{Compress}_q(\mathbf{P}_B, 2^{d_{\mathbf{P}_B}})$ and sets $\mathbf{c}_2 = [h, 0, \ldots, 0]$, where h is a parameter designed to extract different information about the secret, depending on the context. Alice calculates $\mathbf{u}_A = \mathbf{Decompress}_q(\mathbf{c}_1, 2^{d_{\mathbf{P}_B}}) = \mathbf{P}_B$ and $\mathbf{v}_A = \mathbf{Decompress}_q(\mathbf{c}_2, 2^{d_{\mathbf{v}_B}}) = [\lceil \frac{q}{32} h \rfloor, 0, \ldots, 0]$. Next, Alice calculates

$$\begin{aligned}
\mathbf{m}'[0] =& \mathbf{Compress}_q((\mathbf{v}_A - \mathbf{s}_A^{\mathsf{tr}}\mathbf{u}_A)[0], 2) && (1)\\
=& \mathbf{Compress}_q(\mathbf{v}_A[0] - (\mathbf{s}_A^{\mathsf{tr}}\mathbf{u}_A)[0], 2) && (2)\\
=& \left\lceil \frac{2}{q} \left(\left\lceil \frac{q}{32} h \right\rfloor - \mathbf{s}_A[0] \left\lceil \frac{q}{32} \right\rfloor \right) \right\rfloor \mod 2. && (3)
\end{aligned}$$

The value of $\mathbf{m}'[0]$ depends on h and s_0 according to Table 2. Each query gives us partial information about s_0. Notice that we are not able to split the values of s_0 into all possible two subsets of values. All the possible combinations split the values into two adjacent intervals. Attempts at making any other type of split fail. Modifying the mismatch attack to allow for other splits with respect to s_0 leads to a situation where $\mathbf{m}'[i]$ is not always equal to 0 for $i \neq 0$.

Let us show why Alice's received message is equal to 0 by construction on all positions with non-zero index. Since $\mathbf{v}_A[i] = 0$, for index $i \neq 0$, Alice computes

$$\begin{aligned}
\mathbf{m}'[i] =& \mathbf{Compress}_q((\mathbf{v}_A - \mathbf{s}_A^{\mathsf{tr}}\mathbf{u}_A)[i], 2) && (4)\\
=& \mathbf{Compress}_q(\mathbf{v}_A[i] - (\mathbf{s}_A^{\mathsf{tr}}\mathbf{u}_A)[i], 2) && (5)\\
=& \left\lceil \frac{2}{q} \left(-\mathbf{s}_A[i] \left\lceil \frac{q}{32} \right\rfloor \right) \right\rfloor \mod 2. && (6)
\end{aligned}$$

The expression within the outer rounding function is bounded in absolute value by $2/3329 \cdot 2 \cdot 105 = 0.126\ldots < 1/2$. Hence the value is always equal to 0 when rounded to the nearest integer and thus $\mathbf{m}'[i] = 0$, for $i \neq 0$.

To modify the mismatch attack to recover s_i, where $1 \leq i \leq n$, we instead let $\mathbf{P}_B$ be equal to 0 on all positions, except for $\mathbf{P}_B[n-i] = -\lceil \frac{q}{32} \rfloor$.

As Kyber is based on module-LWE, $\mathbf{P}_b$ and $\mathbf{s}$ consist of l blocks, where each block has size n. The multiplication of them consists of a scalar product of two vectors with l polynomials each. Thus, to retrieve positions n to $2n-1$, we just shift the index of $\mathbf{P}_B$ by n positions. That is, we let all positions in $\mathbf{P}_B$ be equal to 0, except that $\mathbf{P}_B[n] = \lceil \frac{q}{32} \rfloor$, to retrieve s_n. To retrieve the value s_{n+i}, for $0 < i < n$, we let all positions of $\mathbf{P}_B$ be equal to 0, except that $\mathbf{P}_B[2n-i] = -\lceil \frac{q}{32} \rfloor$. To retrieve another block we just continue shifting another n positions and so on.

In Figure 3 we illustrate the mismatch attacks from [37] on the different versions of Kyber.

[3] To attack other versions of Kyber you just alter the attack parameters slightly.

Table 2: $\mathbf{m}'[0]$ as a function of s_0 for different values of h for Kyber1024.

	s_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
22	0	1	1	1	1
23	0	0	1	1	1
24	0	0	0	1	1
25	0	0	0	0	1

3.2 Saber

A mismatch attack on Saber works similarly to a mismatch attack on Kyber. Let $\mathbf{c}$ be equal to 0 on all positions, except that $\mathbf{c}[0] = h$. Let H denote $2^{\epsilon_p-2} - 2^{\epsilon_p-\epsilon_T-1}+2^{\epsilon_q-\epsilon_p-1}$. Finally, let $\mathbf{P}_B$ be equal to 0 on all positions except $\mathbf{P}_B[0] = k$. For the first index $i = 0$ we get

$$\mathbf{m}'[0] = ((k(s_i \mod p) + H - 2^{\epsilon_p-\epsilon_T}h) \mod p) \gg (\epsilon_p - 1). \tag{7}$$

For the indices $i \neq 0$ we get

$$\mathbf{m}'[i] = ((k(s_i \mod p) + H) \mod p) \gg (\epsilon_p - 1). \tag{8}$$

If we make k small enough, then we make sure that $\mathbf{m}'[i] = 0$ for all possible values of s_i. Table 3 shows parameters achieving different splits of $\mathbf{m}'[0]$, while making sure that $\mathbf{m}'[i] = 0$, for all $i \neq 0$, for FireSaber[4].

Note that the parameters differ a bit from the ones in [37]. We pick the minimal values of k that achieve each possible split. This allows our multi-positional attacks in Section 4 to use as many positions at the same time as possible.

To retrieve positions s_i, for $1 \leq i \leq n-1$, we make the adjustment that $\mathbf{c}$ is equal to 0 on all positions, except that $\mathbf{c}[n-i] = -k$, where k refers to a value used to achieve a specific split to retrieve the value s_0 for an implied value of h.

Just like Kyber, Saber is based on module-LWE and the mismatch attack retrieves the secret values in blocks of size n. Just like for Kyber, we shift the non-zero indices of $\mathbf{P}_B$ by n positions to retrieve each new block of secret values.

In Figure 3 we illustrate the mismatch attacks from [37] on the different versions of Saber.

3.3 The Lower Bounds from [37]

An obvious lower bound on the attack is the Shannon bound. In a pure mismatch attack, the number of queries cannot be lower than the entropy of the secret.

[4] To attack other versions of Saber you just alter the attack parameters slightly.

Table 3: $\mathbf{m}'[0]$ as a function of s_0 for different values of h and k for FireSaber.

		s_0						
h	k	-3	-2	-1	0	1	2	3
15	5	1	0	0	0	0	0	0
15	7	1	1	0	0	0	0	0
15	13	1	1	1	0	0	0	0
16	4	1	1	1	1	0	0	0
16	2	1	1	1	1	1	0	0
17	7	1	1	1	1	1	1	0
47	5	0	1	1	1	1	1	1
47	7	0	0	1	1	1	1	1
47	13	0	0	0	1	1	1	1
48	4	0	0	0	0	1	1	1
48	2	0	0	0	0	0	1	1
49	7	0	0	0	0	0	0	1

As each position in the secret vector is independent of the other positions, the entropy is equal to the number of positions times the entropy of each position, leading to the Shannon bounds of Table 4.

Regarding their mismatch attacks the authors of [37] write "For simplicity, we assume the adversary recovers Alice's secret key s_A one coefficient block by one coefficient block." For Kyber and Saber, this means recovering one coefficient at a time. Under this restriction, the authors show that Huffman coding is optimal and leads to another lower bound for a pure mismatch attack. In our concrete setting, since not every possible splitting of the secret values into two subsets is possible using our available types of queries, there is no guarantee that we can reach the performance of the Huffman code in practice.

Table 4: Lower limits for key mismatch attacks on Kyber and Saber.

Scheme	s Range	Unknowns	Entropy Per Position	Shannon Bound	Huffman Bound
Kyber512	$[-3, 3]$	512	2.3334	1195	1216
Kyber768	$[-2, 2]$	768	2.0306	1560	1632
Kyber1024	$[-2, 2]$	1024	2.0306	2079	2176
LightSaber	$[-5, 5]$	512	2.7064	1386	1412
Saber	$[-4, 4]$	768	2.5442	1954	1986
FireSaber	$[-3, 3]$	1024	2.3334	2389	2432

3.4 The Practical Mismatch Attacks from [37]

Figure 3 illustrates and summarizes the practical mismatch attacks on Kyber and Saber from [37]. The blue/red/green/yellow/brown/orange lines correspond to the 1st/2nd/3rd/4th/5th/6th splits respectively.

All the attacks on both Kyber and Saber follow the same strategy. Start with a query that splits the possibilities for the secret value s_i into two halves as equally probable as possible. Then no matter on which half of the secret values we end up, each of the remaining queries decides whether the s_i is equal to the remaining most probable value or any of the other values[5].

There are two interesting ways of viewing these mismatch attacks, which we generalize to higher dimensions in Section 4.4.

Partly, the attacks correspond to first making as even of a split as possible, and then for the remaining part of the attack make splits that correspond to Huffman coding[6]. In Section 4.4 we apply a similar strategy where we start out with a couple of roughly even splits of the secret values and then apply steps that are identical to/close to Huffman coding. This approach is applied manually for attacks on all versions of Kyber and on FireSaber.

Another perspective on these mismatch attacks is that they are all greedy attacks. Each split for each attack is the split that divides up the remaining secret values into two as equally probable halves as possible. This approach is generalized in Section 4.4 for our attacks on Saber and LightSaber, systems where creating attacks manually by hand is tedious due to the s_i values taking a wider range of values.

3.5 On the Performance of the Mismatch Attacks From [37]

In Table 5 we list the expected number of queries used for the practical mismatch attacks from [37]. Notice that there are small deviations between our table and Table 6 of [37]. This is due to us avoiding premature rounding. Notice that for all deviations, our corrected values are closer to the experimental results of [37].

For Kyber and Saber we can argue that their approach is optimal given their restrictions.

1. By testing all parameter settings we find for both Kyber and Saber that the only possible splits divide the secret values into two adjacent intervals.
2. For the initial split it should be optimal to split the secret values as evenly as possible, that is, split into one interval of positive values, one interval of negative values, and put the value 0 in whichever of the two intervals.
3. For all versions of both Kyber and Saber, their approach after the initial split is identical to Huffman coding, which is optimal, given the restrictions.

[5] The fact that all the mismatch attacks on Kyber and Saber follow this approach is due to the distribution of the s_i values. For another distribution, such as the uniform distribution, this would of course not be a sensible strategy.

[6] For all the attacks, after the initial split, the remaining splits correspond to Huffman coding for their respective half of the secret values. The possibility of this depends on the distribution of the s_i values. This is not possible for all secret distributions.

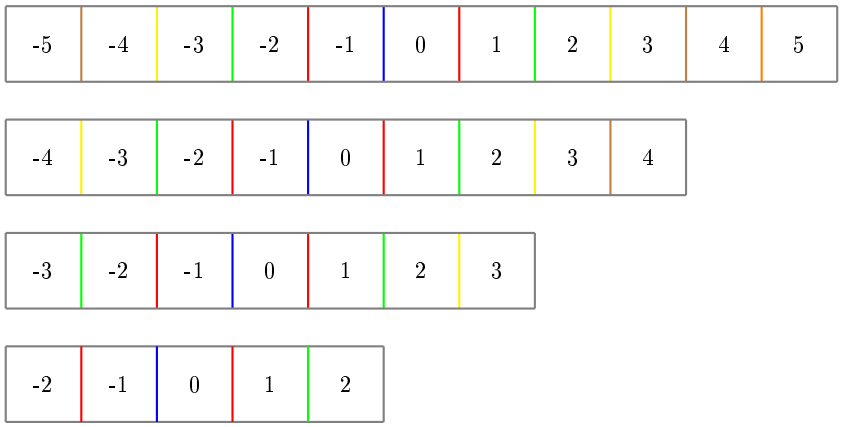

Fig. 3: Illustrations of the mismatch attacks on all versions of Kyber and Saber from [37]. Starting from the bottom of the figure and moving upwards the illustrations cover Kyber768/Kyber1024, Kyber512/FireSaber, Saber, and LightSaber, respectively.

Table 5: Practical mismatch attacks compared to the Huffman bounds.

Scheme	s Range	Unknowns	Queries Per Position	Total Queries	Huffman Bound
Kyber512	$[-3, 3]$	512	2.5625	1312	1216
Kyber768	$[-2, 2]$	768	2.3125	1776	1632
Kyber1024	$[-2, 2]$	1024	2.3125	2368	2176
LightSaber	$[-5, 5]$	512	2.8515..	1460	1412
Saber	$[-4, 4]$	768	2.7226..	2091	1986
FireSaber	$[-3, 3]$	1024	2.5625	2624	2432

4 Multi-Positional Mismatch Attacks

The main idea of this paper is to remove the constraint of recovering only one coefficient at a time. Let us begin by explaining how our idea works with two positions at a time and first explain it for Kyber.

4.1 Two-Positional Mismatch Attacks on Kyber

To start with we will show how to obtain s_0 and s_{128} (for the setting where $\mathbf{s}_A$ consists of blocks of size 256, which covers all versions of Kyber and Saber) and then later explain how to generalize this approach to obtain the other values of $\mathbf{s}_A$. Let us focus on Kyber1024[7]. Eve lets $\mathbf{m}$ have the value 0 on all positions, except that $\mathbf{m}[0] = 1$ and/or $\mathbf{m}[128] = 1$. Let $\mathbf{P}_B$ be 0 on all positions except that $\mathbf{P}_B[0] = b_1 \cdot \lceil \frac{q}{32} \rfloor$ and $\mathbf{P}_B[128] = b_2 \lceil \frac{q}{32} \rfloor$, where $b_1, b_2 \in \{-1, 0, 1\}$. Also, let $\mathbf{c}_2$ be 0 on all positions, except that $\mathbf{c}_2[0] = h_1 \lceil \frac{q}{32} \rfloor$ and $\mathbf{c}_2[128] = h_2 \lceil \frac{q}{32} \rfloor$. Next, we calculate $\mathbf{m}'[0]$ and $\mathbf{m}'[128]$. We get

$$\mathbf{m}'[0] = \mathbf{Compress}_q(\mathbf{v}_A[0] - (\mathbf{s}_A^{\mathsf{tr}}\mathbf{u}_A)[0], 2) \tag{9}$$

$$= \left\lceil \frac{2}{q} \left(\left\lceil \frac{q}{32} h_1 \right\rfloor - \left(\mathbf{s}_A[0] b_1 \left\lceil \frac{q}{32} \right\rfloor - \mathbf{s}_A[128] b_2 \left\lceil \frac{q}{32} \right\rfloor \right) \right) \right\rfloor \mod 2, \tag{10}$$

and

$$\mathbf{m}'[128] = \mathbf{Compress}_q(\mathbf{v}_A[128] - (\mathbf{s}_A^{\mathsf{tr}}\mathbf{u}_A)[128], 2) \tag{11}$$

$$= \left\lceil \frac{2}{q} \left(\left\lceil \frac{q}{32} h_2 \right\rfloor - \left(\mathbf{s}_A[0] b_2 \left\lceil \frac{q}{32} \right\rfloor + \mathbf{s}_A[128] b_1 \left\lceil \frac{q}{32} \right\rfloor \right) \right) \right\rfloor \mod 2. \tag{12}$$

For an integer i, with $1 \leq i \leq 127$ we get

$$\mathbf{m}'[i] = \mathbf{Compress}_q(-(\mathbf{s}_A^{\mathsf{tr}}\mathbf{u}_A)[i], 2) \tag{13}$$

$$= \left\lceil \frac{2}{q} \left(- \left(\mathbf{s}_A[i] b_1 \left\lceil \frac{q}{32} \right\rfloor - \mathbf{s}_A[128+i] b_2 \left\lceil \frac{q}{32} \right\rfloor \right) \right) \right\rfloor \mod 2, \tag{14}$$

and

$$\mathbf{m}'[128+i] = \mathbf{Compress}_q(-(\mathbf{s}_A^{\mathsf{tr}}\mathbf{u}_A)[128+i], 2) \tag{15}$$

$$= \left\lceil \frac{2}{q} \left(- \left(\mathbf{s}_A[i] b_2 \left\lceil \frac{q}{32} \right\rfloor + \mathbf{s}_A[128+i] b_1 \left\lceil \frac{q}{32} \right\rfloor \right) \right) \right\rfloor \mod 2. \tag{16}$$

For both of the latter two positions the expression within the outer rounding function is bounded in absolute value by $2/3329 \cdot 2 \cdot 2 \cdot 105 = 0.252\ldots < 1/2$. Hence these values are always equal to 0 when rounded to the nearest integer and thus $\mathbf{m}'[i] = 0$, for $i \neq 0, 128$.

[7] To attack other versions of Kyber you just alter the parameters of the attack slightly.

To retrieve the positions s_i and s_{128+i}, where $1 \leq i \leq 127$, we can for example make the following adjustments. Let $\mathbf{m}$ be equal to 0 on all positions except that $\mathbf{m}[i] = 1$ and/or $\mathbf{m}[128+i] = 1$. Also, let $\mathbf{c}_2$ be 0 on all positions, except that $\mathbf{c}_2[i] = h_1\lceil \frac{q}{32} \rfloor$ and $\mathbf{c}_2[128+i] = h_2\lceil \frac{q}{32} \rfloor$.

Next, we will interpret our two-positional approach. We organize the possible pairs of values s_0, s_{128} in a two-dimensional grid. For each cell we write the value of $\mathbf{m}'[0]$, $\mathbf{m}'[128]$ or $\mathbf{m}'[0]\&\mathbf{m}'[0]$, depending on whether only $\mathbf{m}[0] = 1$, only $\mathbf{m}[128] = 1$ or if $\mathbf{m}[0] = \mathbf{m}[128] = 1$. Here of course the value of the bits depend in general on s_0, s_{128} and the chosen parameters b_1, b_2, h_1, h_2.

Planar Cuts The most obvious split that can be achieved is the one where we cut with respect to only one of the variables. These planar cuts are achieved by letting $\mathbf{m}[0] = 1$, $\mathbf{m}[128] = 0$ and $h_2 = 0$. Two examples of such splits are shown in Figure 4. To achieve a vertical split we let $b_2 = 0$ and $b_1 = 1$. In Figure 4a, specifically we let $h_1 = 9$. To achieve a horizontal split we let $b_1 = 0$, $b_2 = -1$. In Figure 4b specifically we let $h_1 = 24$. Horizontal and vertical splits are already the ones achieved in the mismatch attacks from [37], explained in detail in Section 3. In and of themselves, these two types of splits do not add anything to the mismatch attacks compared to previous work, but they are useful in combination with the other methods from this section.

	s_0				
m'_0	-2	-1	0	1	2
-2	1	1	1	0	0
-1	1	1	1	0	0
s_{128} 0	1	1	1	0	0
1	1	1	1	0	0
2	1	1	1	0	0

(a) A vertical split.

	s_0				
m'_0	-2	-1	0	1	2
-2	0	0	0	0	0
-1	0	0	0	0	0
s_{128} 0	0	0	0	0	0
1	1	1	1	1	1
2	1	1	1	1	1

(b) A horizontal split.

Fig. 4: Two examples of planar splits of the secret values.

Rectangular Cuts We can also simultaneously cut horizontally and vertically. This allows us to cut out any rectangle of values, where at least one corner of the rectangle is at one of the corners of the grid. To do this we let $\mathbf{m}[0] = \mathbf{m}[128] = 1$. We let $b_1 = 1$ and $b_2 = 0$. In Figure 5 we let $h_1 = 24$ and $h_2 = 9$. The figure

shows m'_0, m'_{128} and $m' = m'_0 \& m'_{128}$ respectively. In other words, the vertical cut, the horizontal cut, and the resulting rectangular cut respectively.

m'_0 \ s_0	-2	-1	0	1	2
-2	0	0	0	1	1
-1	0	0	0	1	1
s_{128} 0	0	0	0	1	1
1	0	0	0	1	1
2	0	0	0	1	1

(a) The vertical cut

m'_{128} \ s_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

(b) The horizontal cut

m' \ s_0	-2	-1	0	1	2
-2	0	0	0	1	1
-1	0	0	0	1	1
0	0	0	0	1	1
1	0	0	0	0	0
2	0	0	0	0	0

(c) The rectangular result

Fig. 5: The cuts with respect to m'_0, m'_{128} and the rectangular cut as their intersection.

Triangular Cuts Our next type of split is a triangular cut, originating from any of the 4 corners of the grid. Here we let $\mathbf{m}[0] = 1$, $\mathbf{m}[0] = 1$ and $h_2 = 0$. See Figure 6 for two examples of this type of cut. In Figure 6a we let $h_1 = 10$, $b_1 = 1$ and $b_2 = -1$. In Figure 6b we let $b_1 = b_2 = 1$ and $h_1 = 24$.

Intersections of Two Triangular Cuts Finally, by letting $\mathbf{m}[0] = \mathbf{m}[0] = 1$ and $b_1, b_2 \neq 0$, we create the intersection of two perpendicular triangular cuts. Notice that we are not able to create the intersection of all possible pairs of triangular cuts. The sign change and flip of b_1 and b_2 in (9) compared to (11) means that the two triangular cuts cannot originate from the same corner or from opposite corners. See Figure 7 for an example of this type of cut. Here we let $b_1 = b_2 = 1$, $h_1 = 24$ and $h_2 = 10$.

4.2 Two-Positional Mismatch Attacks on Saber

Two-positional mismatch attacks on Saber work similarly to the corresponding attacks on Kyber. We will briefly cover how to modify the parameters to make the attacks work to retrieve the values s_0 and s_{128}. The modifications used to recover the rest of the positions are analogous to the modifications covered in Section 4.1. Let $\mathbf{P}_B$ be equal to 0 on all positions, except that $\mathbf{P}_B[0] = k_1$ and $\mathbf{P}_B[128] = k_2$. Let $\mathbf{c}$ be equal to 0 on all positions, except that $\mathbf{c}[0] = h_1$ and $\mathbf{c}[128] = h_2$. For the index 0, we get

$$\mathbf{m}'[0] = ((k_1(s_0 \mod p) - k_2(s_{128} \mod p) + H - 2^{\epsilon_p - \epsilon_T} h_1) \mod p) \gg (\epsilon_p - 1). \tag{17}$$

m'_0 \ s_0	-2	-1	0	1	2
-2	0	1	1	1	1
-1	0	0	1	1	1
s_{128} 0	0	0	0	1	1
1	0	0	0	0	1
2	0	0	0	0	0

(a) A triangular cut of the secret values, originating from the upper right corner.

m'_0 \ s_0	-2	-1	0	1	2
-2	1	1	1	1	1
-1	1	1	1	1	1
s_{128} 0	1	1	1	1	0
1	1	1	1	0	0
2	1	1	0	0	0

(b) A triangular cut of the secret values, originating from the upper left corner.

Fig. 6: Two examples of triangular cuts.

m'_0 \ s_0	-2	-1	0	1	2
-2	1	1	1	1	1
-1	1	1	1	1	1
s_{128} 0	1	1	1	1	0
1	1	1	1	0	0
2	1	1	0	0	0

(a) First triangular cut

m'_{128} \ s_0	-2	-1	0	1	2
-2	0	1	1	1	1
-1	0	0	1	1	1
0	0	0	0	1	1
1	0	0	0	0	1
2	0	0	0	0	0

(b) Second triangular cut

m' \ s_0	-2	-1	0	1	2
-2	0	1	1	1	1
-1	0	0	1	1	1
0	0	0	0	1	0
1	0	0	0	0	0
2	0	0	0	0	0

(c) The intersection

Fig. 7: The cuts with respect to m'_0, m'_{128} and their intersection.

For the index 128 we get

$$\mathbf{m}'[128] = ((k_1(s_{128} \mod p) + k_2(s_0 \mod p) + H - 2^{\epsilon_p - \epsilon_T} h_1) \mod p) \gg (\epsilon_p - 1). \tag{18}$$

Now, for an integer i, where $1 \leq i \leq 127$ we get

$$\mathbf{m}'[i] = ((k_1(s_i \mod p) - k_2(s_{128+i} \mod p) + H) \mod p) \gg (\epsilon_p - 1), \tag{19}$$

and

$$\mathbf{m}'[128 + i] = ((k_2(s_i \mod p) + k_1(s_{128+i} \mod p) + H) \mod p) \gg (\epsilon_p - 1). \tag{20}$$

For small values of k_1 and k_2 the expressions in (19) and (20) are equal to 0 for all secrets $\mathbf{s}_A$. Combining k_1 and k_2 with suitable values of h_1, h_2 we split the values in (17) and (18) correctly as a function of the values of s_0 and s_{128}.

Just like in the one-dimensional setting, all the splits that we have introduced for Kyber can also be done for Saber. Thus, when designing our mismatch attacks in Section 4.4 we only need to consider the distribution of $\mathbf{s}_A$.

4.3 Hyperrectangular Cuts

It is possible to generalize the idea of [37] in other ways. Instead of making planar cuts in one dimension at a time, we can make planar cuts with respect to an arbitrary number of positions at a time. Let us explain the idea for Kyber. Let $I \subset \{0, 1, \ldots, n-1\}$ be the set of indices that we want to make planar splits with respect to. Let $\mathbf{m}[i] = 1$, for $i \in I$, and $\mathbf{m}[i] = 0$, for $i \notin I$. Let $\mathbf{P}_B$ be equal to 0 on all positions except that $\mathbf{P}_B[0] = \lceil \frac{q}{32} \rfloor$. Let $\mathbf{c}_2[i] = 0$, for $i \notin I$ and $\mathbf{c}_2[i] = h_i$, for $i \in I$. Here h_i are the parameters deciding the precise planar cut with respect to each dimension. For $i \in I$ we now get

$$\begin{aligned} \mathbf{m}'[i] =& \mathbf{Compress}_q(\mathbf{v}_A[i] - (\mathbf{s}_A^{\mathsf{tr}} \mathbf{u}_A)[i], 2) && (21) \\ =& \left\lceil \frac{2}{q} \left(\left\lceil \frac{q}{32} h_i \right\rfloor - \mathbf{s}_A[i] \left\lceil \frac{q}{32} \right\rfloor \right) \right\rfloor \mod 2. && (22) \end{aligned}$$

For other indices we get

$$\begin{aligned} \mathbf{m}'[i] =& \mathbf{Compress}_q(\mathbf{v}_A[i] - (\mathbf{s}_A^{\mathsf{tr}} \mathbf{u}_A)[i], 2) && (23) \\ =& \left\lceil \frac{2}{q} \left(-\mathbf{s}_A[i] \left\lceil \frac{q}{32} \right\rfloor \right) \right\rfloor \mod 2, && (24) \end{aligned}$$

which simplifies to 0 as explained above. The idea works similarly for the Saber schemes, except the values of $\mathbf{m}'[i]$ are evaluated according to (7) for $i \in I$ (with the index 0 replaced by i) and (8) for $i \notin I$.

It is of course possible to create a lot of other cuts in higher dimensions. We discuss these cuts and their potential in Section 5.1.

4.4 The Optimization Problem

Now we have introduced a set of cuts in the multi-positional setting and are ready to apply them to some schemes. In two dimensions, the optimization problem we now want to solve is, given our available planar, rectangular, triangular, and intersecting triangular splits, how do we come up with a splitting strategy, given the distribution of $\mathbf{s}_A$, that minimizes the expected number of splits?

First, we will cover our attacks on all versions of Kyber and on FireSaber in detail. Then we will show how we devised a greedy algorithm to develop an attack on Saber and LightSaber.

Kyber1024 and Kyber768 In Kyber768 and Kyber1024 the s_i values are centered binomially distributed with $\eta = 2$. The probabilities of the possible value pairs (s_0, s_{128}) are according to Figure 8.

The figure also shows the first four queries of the mismatch attack. For example, on the first query, represented by the blue lines, we learn whether the secret pair is among the nine positions in the lower left part or whether it is among the other sixteen positions. Depending on which is the case, the second query we make corresponds to either of the red splits and so on. The figure shows the first four queries.

Figure 9 then shows the next three queries. The positions that are filled in black are the positions that are found in less than or equal to four queries. Within seven queries the secret pair is guaranteed to be found. Notice for both figures that we do not always specify exactly which split we use. Given that the values of the secret pair are limited to a certain area, we only specify how the split works within that area. Given the lack of restrictions on the split's behavior outside of this area, most splits are not uniquely determined.

The overall strategy for choosing which splits to make is the following. Start by making a few splits that roughly divide up the possible secret pairs into equally probable blocks. Then make queries that perform identically/close to Huffman coding. This roughly generalizes the strategy of the mismatch attack on Kyber768/Kyber1024 in one dimension, as discussed in Section 3.4.

The number of queries needed to find all the different secret value pairs is found in Figure 10. Using these figures together, we calculate the expected number of queries to recover two positions as $1059/256 \approx 4.1367$. This corresponds to roughly 2.0684 queries per position. We compare this query complexity against other algorithms and some boundaries in Section 4.5.

Kyber512 and FireSaber In Kyber512 and FireSaber the values s_i are centered binomially distributed with $\eta = 3$. Thus, the probabilities of the possible value pairs (s_0, s_{128}) are according to Figure 11. This figure also illustrates the first four queries of the mismatch attack. The remaining six queries are covered in Figure 12. Just like for Kyber1024/Kyber768, the strategy is to start by making a couple of roughly even splits followed by using (close to) Huffman coding in latter queries. Also for these schemes, the strategy roughly mimics and

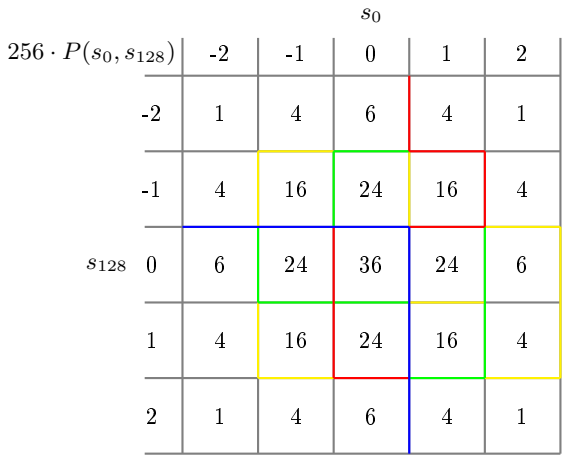

Fig. 8: The first part of the two-positional mismatch attack against Kyber768/Kyber1024. All the probabilities are multiplied by 256 for readability. The blue, red, green and yellow splits correspond to the first, second, third and fourth queries respectively.

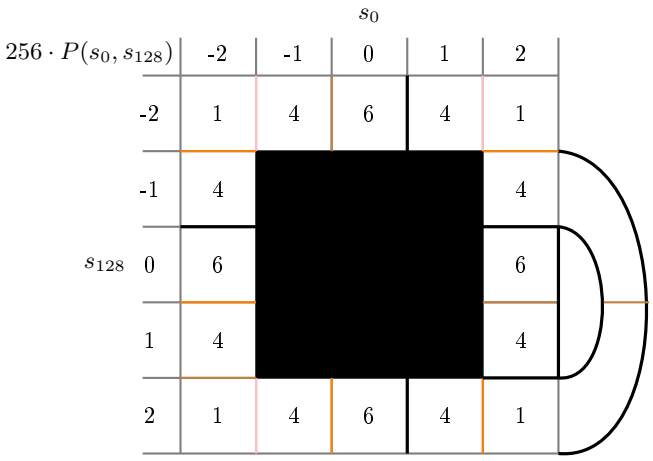

Fig. 9: The second part of the two-positional mismatch attack against Kyber768/Kyber1024. All the probabilities are multiplied by 256 for readability. The brown, orange and pink splits correspond to the fifth, sixth and seventh queries respectively. The positions that are filled in black correspond to the value pairs that are found after less than or equal to four queries. The curved path between the lower right and the upper right areas signals that these positions correspond to the same block after four queries.

		s_0				
		-2	-1	0	1	2
	-2	7	7	5	7	7
	-1	6	4	4	4	6
s_{128}	0	6	3	3	4	5
	1	6	4	3	4	5
	2	7	7	6	6	6

Fig. 10: The number of queries needed to find the secret pair (s_0, s_{128}) for the different values of this pair for the two-positional mismatch attack on Kyber768/Kyber1024.

generalizes the strategy of the mismatch attack on Kyber512/FireSaber in one dimension, as discussed in Section 3.4.

The number of queries needed to find all the different secret value pairs is found in Figure 13. Using these figures together, we calculate the expected number of queries to recover two positions as $19285/4096 \approx 4.70825$. This corresponds to roughly 2.3541 queries per position. We compare this performance against other algorithms and some boundaries in Section 4.5.

The Automatic Greedy Approach For Saber and LightSaber the number of possible pairs of secret values is too large to make manual optimization practical. To get a decent attack against these two schemes we make an automatic search for a solution instead. In two dimensions we apply a greedy attack, where the algorithm in each step chooses the query that splits the remaining positions as evenly as possible. It turns out that the algorithm performs better when only using planar and rectangular splits, compared to when adding triangular/triangular intersection splits. Using the latter types of splits makes the algorithm choose a worse local optimum.

We generalize this attack to three dimensions, where we allow for any types of cuboid splits (the hyperrectangular splits in three dimensions). This corresponds to letting $|I| \leq 3$ in Section 4.3. The performance of the attack in two and three dimensions is shown in Table 6.

All the mismatch attacks in [37] on Kyber and Saber are hyperrectangular, greedy attacks in one dimension, making this type of greedy approach a pretty natural generalization. Perhaps not surprisingly, the greedy hyperrectangular attack (mostly) performs better in three dimensions than in two, and better in two

$4096 \cdot P(s_0, s_{128})$	s_0 -3	-2	-1	0	1	2	3
3	1	6	15	20	15	6	1
2	6	36	90	120	90	36	6
1	15	90	225	300	225	90	15
s_{128} 0	20	120	300	400	300	120	20
-1	15	90	225	300	225	90	15
-2	6	36	90	120	90	36	6
-3	1	6	15	20	15	6	1

Fig. 11: The first part of the two-positional mismatch attack against Kyber512/FireSaber. All the probabilities are multiplied by 4096 for readability. The blue, red, green and yellow splits correspond to the first, second, third and fourth queries respectively. The green diagonal split decides whether the secret is $(0, 0)$ or any of the values in the upper left block. The yellow diagonal split decides whether the secret is $(0, -1)$ or $(-1, 0)$.

Table 6: The performance of the greedy mismatch attacks in two and three dimensions on all versions of Kyber and Saber, measured in expected number of queries per secret position.

	Kyber512	Kyber768	Kyber1024	LightSaber	Saber	FireSaber
Two Dimensions	2.4561	2.0820	2.0820	2.7643	2.6256	2.4561
Three Dimensions	2.3771	2.0837	2.0837	2.7540	2.5839	2.3771

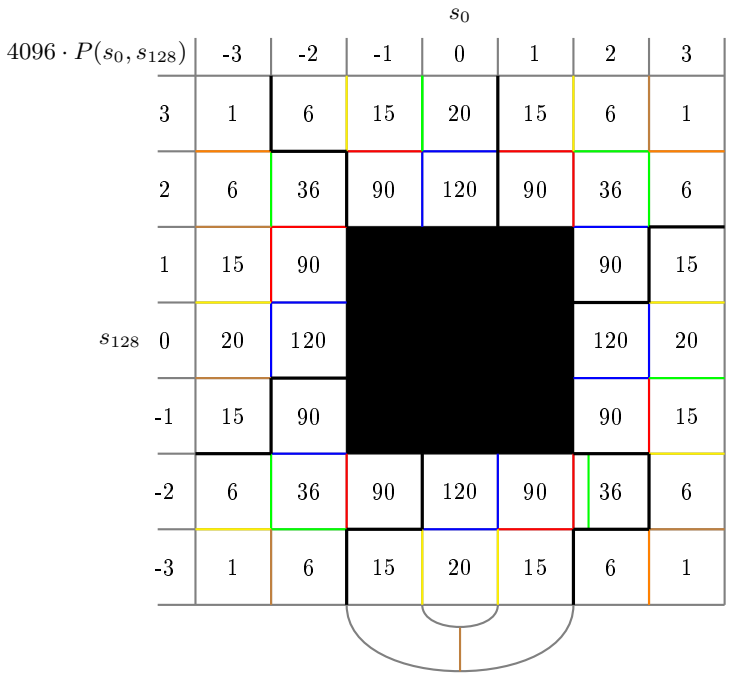

Fig. 12: The two-positional mismatch attack against Kyber512/FireSaber. The queries after the first four queries. The blue, red, green, yellow, brown and orange splits correspond to the fifth, sixth, seventh, eighth, ninth and tenth queries respectively. All the probabilities are multiplied by 4096 for readability. The green split inside the box to the lower right decides whether the secret pair is $(2, -2)$ or any of the other remaining values of the area. The brown split that crosses the curved lines decides whether the secret is $(-1, -3)$ or $(1, -3)$.

s_{128} \ s_0	-3	-2	-1	0	1	2	3
3	10	8	8	7	8	10	10
2	10	7	6	5	6	7	9
1	9	6	4	4	4	5	8
0	9	5	4	3	4	5	8
-1	9	5	4	4	4	6	8
-2	8	7	6	5	6	7	9
-3	9	9	9	8	9	10	10

Fig. 13: The number of queries needed to find the secret pair (s_0, s_{128}) for the different values of this pair for the two-positional mismatch attack on Kyber512/FireSaber.

than in one. In three dimensions the attack performs slightly better than/slightly worse than the Huffman bound in one dimension, see Table 7 and Section 4.5 for a comparison of this approach to other algorithms and limits.

Notice that this is essentially the most obvious type of automatic attack. There is room for all sorts of improvements here, which we discuss in Section 5.1.

4.5 Comparisons

In Table 7 we compare our mismatch attacks against the previous state-of-the-art, previous lower bounds and new lower bounds. Our Result 1 refers to manual optimization in two dimensions. Our Result 2 refers to the best greedy attack from Section 4.4 for the different schemes[8]. The values in bold are the new state-of-the-art values. When performing two-positional attacks, the lower limit is Huffman coding in two dimensions. When performing three-positional attacks, the lower limit is Huffman coding in three dimensions. The performance of Huffman coding in one, two and three dimensions respectively is called Huffman

[8] Notice that for the greedy attacks in three dimensions the performance is marginally worse than the number of positions times the performance from Table 6. The secret is retrieved in blocks of 256 positions at a time. After finding $3 \cdot 84 = 252$ positions using the three-positional attack, the remaining 4 positions of the block have to be retrieved using two-positional steps, which are slightly worse for most schemes.

Bound 1, 2 and 3 respectively. Finally, the Shannon Bound refers to the entropy of the secret, which is the theoretically best performance that you can achieve with a pure mismatch attack.

Table 7: Our results compared to various bounds and previous practical attacks

	Kyber512	Kyber768	Kyber1024	LightSaber	Saber	FireSaber
Previous Best [37]	1312	1776	2368	1460	2091	2624
Huffman Bound 1	1216	1632	2176	1412	1986	2432
Our Result 1	**1205.3**	**1588.5**	**2118**	-	-	**2410.6**
Our Result 2	1217.7	1599	2132	**1410.2**	**1984.9**	2435.4
Huffman Bound 2	1202.1	1575	2100	1395.9	1970.0	2404.3
Huffman Bound 3	1199.9	1569.8	2093.0	1391.7	1962.3	2399.7
Shannon Bound	1195	1560	2079	1386	1954	2389

Unlike the one-dimensional situation analyzed in Section 3.5, we do not claim that our strategy is optimal given our restrictions. Both developing an optimal strategy, and showing that a strategy is optimal, are much harder in our setting.

However, we can say that our results are fairly close to optimal. For all the schemes our attacks are much closer in performance to the Shannon bound than to [37]. We have thus made most of the possible improvements that can be made.

For all versions of Kyber and for FireSaber, we are very close to the Huffman bound in two dimensions, showing that there is very little room for improvement. Our greedy attacks against Saber and LightSaber are obviously not optimal in our given context, but can still only be improved a little bit, as they both perform fairly close to the Huffman bound in three dimensions.

It should also be noted that due to not all imaginable splits being accessible in practice, even the optimal strategy performs worse than the Huffman bounds for the respective dimension, making our results even closer to optimal.

In Section 5.1 we discuss how improving our attacks probably makes it possible to reach the Shannon bound. In Section 5.2 we discuss how we can beat the Shannon bound by using post-processing.

5 Discussions

5.1 Room for Further Improvements

For the two-positional splits of Kyber, by letting $b_1 = 2$, $b_2 = 1$, vice versa or changing signs of one of the variables, we can create yet other cuts[9]. This creates even more possible splits in two dimensions.

[9] Notice that we cannot increase the values of b_1 and b_2 too much, because then the other indices of (13) are not guaranteed to be equal to 0.

The hyperrectangular cuts from Section 4.3 only generalize the planar and rectangular splits. Of course, a lot of other splits are also possible in higher dimensions. Even limited to only hyperrectangular splits in higher dimensions our fairly simple greedy attack can of course be improved with smarter approaches and/or by increasing the number of dimensions.

Thus, we conjecture that getting (arbitrarily close) to the Shannon bound is possible, because of the numerous potential improvements. Meanwhile, we emphasize that our current results are already close to the bound.

5.2 Post-Processing with Lattice Reduction

Previous literature on mismatch attacks focuses on the number of queries needed to fully recover the secret $\mathbf{s}$. However, the adversary also has access to LWE samples. Not taking advantage of this information in an attack is clearly sub-optimal. While [37] states "Secondly, what we talk about is recovering the full key, but obviously the recovery of the partial key also leaks information about the key, significantly decreasing the bit-security.", they did not study this hybrid approach in detail.

In principle, the optimal hybrid mismatch attack with post-processing chooses, given a limited number of queries, among all possible mismatch strategies, the strategy that minimizes the complexity of the post-processing. A paper that studied solving the LWE and NTRU problem with hints about the secret more generally is [14]. In our setting it is most likely optimal to find the exact position value of as many positions as possible and, if the final few queries are not enough to completely know the final positions being queried, use the partial information we have on these positions.

We plot the relationship between the allowed number of queries using the best mismatch attack from Table 7 and the post-processing complexity in Figure 14, for all versions of Kyber and Saber. We use the $\mathsf{primal\_bdd}(\cdot)$ function in the new Lattice-Estimator[10] [5] to estimate the cost of solving the corresponding LWE instance. Without post-processing, the right-most vertical line shows the query complexity of the current best two-positional attack and the second (dotted) vertical line to the right shows the Shannon lower bound. We can see that the two lines are fairly close, so our new attack is near-optimal in the information-theoretical sense. We also see that the query complexity can be highly reduced by using post-processing. For instance, the query complexity is reduced by a factor 2 for Kyber512, when the allowed post-processing cost is 2^{80}.

5.3 Relation to Side-Channel and Fault-Injection attacks

Similar to the one-positional mismatch attacks proposed in [37], our multi-positional version can improve the query (or trace) complexity of chosen-ciphertext side-channel attacks on CCA-secure implementations of Kyber and Saber. The

[10] `https://github.com/malb/lattice-estimator`.

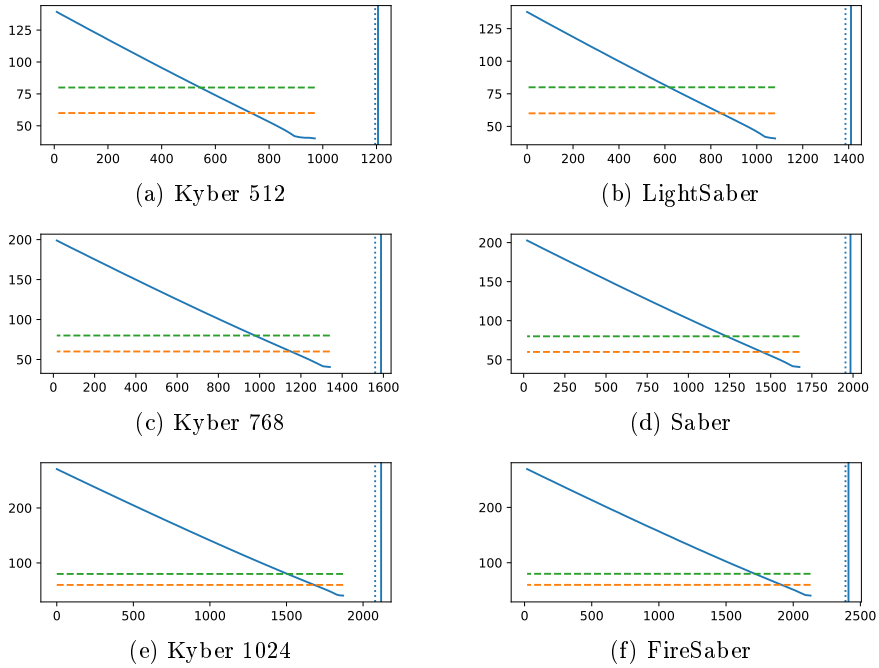

Fig. 14: The post-processing complexity in $\log_2(\cdot)$ **vs.** the number of queries. The two horizontal lines represent the post-processing complexity of 2^{80} (upper) and 2^{60} (lower), respectively. The most right vertical line shows the current best two-positional attack without post-processing and the second right (dotted) vertical line shows the Shannon lower bound for the attacks without post-processing.

method of generating the chosen ciphertexts is the same as the approach of selecting ciphertexts described in Section 4. The new attacks may also be applied to improve the fault-injection attack proposed in [43].

6 Conclusions and Future Work

In this paper, we have proposed novel multi-positional key mismatch attacks on Kyber and Saber that significantly improve the state-of-the-art mismatch attacks of [37]. Our new attacks even beat the lower bounds proposed in [37], and they are near-optimal since their query complexities are close to the Shannon lower bounds. The new attacks can be applied to improve the efficiency of chosen-ciphertext side-channel attacks against Kyber and Saber and may also have significance in fault-injection attacks.

The new idea of targeting multiple secret coefficients simultaneously can be generalized to other lattice-based KEMs. This translation is probably straightfor-

ward for other LWE/LWR-based KEMs such as FrodoKEM [34], but might also be applicable to KEMs based on NTRU. e.g., NTRU [12] and NTRU prime [9].

We have also estimated how much the mismatch attacks can be improved by using post-processing, showing that we can beat the Shannon bounds with this approach when having access to moderate computational resources.

As we conjecture in our paper, by improving the greedy algorithm and increasing the dimensions, it is likely possible to come arbitrarily close to the Shannon bound for all versions of Kyber and Saber. One interesting topic is to apply more advanced automatic tools, such as mixed integer linear programming (MILP) or constraint programming, to search for better mismatch attacks. On the other hand, our results are near-optimal, so the room for improvement is small in practice.

References

1. NIST Post-Quantum Cryptography Standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization, accessed: 2018-09-24
2. Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf, accessed: 2021-02-18
3. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing. pp. 284–293. STOC '97, Association for Computing Machinery, New York, NY, USA (1997), https://doi.org/10.1145/258533.258604
4. Albrecht, M.R., Bernstein, D.J., Chou, T., Cid, C., Gilcher, J., Lange, T., Maram, V., von Maurich, I., Misoczki, R., Niederhagen, R., Paterson, K.G., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., Tjhai, C.J., Tomlinson, M., Wang, W.: Classic McEliece. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
5. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology 9(3), 169–203 (2015)
6. Băetu, C., Durak, F.B., Huguenin-Dumittan, L., Talayhan, A., Vaudenay, S.: Misuse attacks on post-quantum cryptosystems. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019, Part II. Lecture Notes in Computer Science, vol. 11477, pp. 747–776. Springer, Heidelberg, Germany, Darmstadt, Germany (May 19–23, 2019)
7. Bauer, A., Gilbert, H., Renault, G., Rossi, M.: Assessment of the key-reuse resilience of NewHope. In: Matsui, M. (ed.) Topics in Cryptology – CT-RSA 2019. Lecture Notes in Computer Science, vol. 11405, pp. 272–292. Springer, Heidelberg, Germany, San Francisco, CA, USA (Mar 4–8, 2019)
8. Bernstein, D.J., Bruinderink, L.G., Lange, T., Panny, L.: HILA5 Pindakaas: On the CCA security of lattice-based encryption with error correction. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 18: 10th International Conference on Cryptology in Africa. Lecture Notes in Computer Science, vol. 10831, pp. 203–216. Springer, Heidelberg, Germany, Marrakesh, Morocco (May 7–9, 2018)

9. Bernstein, D.J., Brumley, B.B., Chen, M.S., Chuengsatiansup, C., Lange, T., Marotzke, A., Peng, B.Y., Tuveri, N., van Vredendaal, C., Yang, B.Y.: NTRU Prime. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
10. Bindel, N., Schanck, J.M.: Decryption failure is more likely after success. In: Ding, J., Tillich, J.P. (eds.) Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020. pp. 206–225. Springer, Heidelberg, Germany, Paris, France (Apr 15–17 2020)
11. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) Advances in Cryptology – CRYPTO'98. Lecture Notes in Computer Science, vol. 1462, pp. 1–12. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 23–27, 1998)
12. Chen, C., Danba, O., Hoffstein, J., Hulsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P., Whyte, W., Zhang, Z., Saito, T., Yamakawa, T., Xagawa, K.: NTRU. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
13. Cover, T.M., Thomas, J.A.: Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing). Wiley-Interscience, USA (2006)
14. Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with side information: Attacks and concrete security estimation. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 329–358. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020)
15. D'Anvers, J.P., Guo, Q., Johansson, T., Nilsson, A., Vercauteren, F., Verbauwhede, I.: Decryption failure attacks on IND-CCA secure lattice-based schemes. In: Lin, D., Sako, K. (eds.) PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II. Lecture Notes in Computer Science, vol. 11443, pp. 565–598. Springer, Heidelberg, Germany, Beijing, China (Apr 14–17, 2019)
16. 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), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
17. D'Anvers, J.P., Rossi, M., Virdia, F.: (One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020, Part III. Lecture Notes in Computer Science, vol. 12107, pp. 3–33. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020)
18. D'Anvers, J.P., Tiepelt, M., Vercauteren, F., Verbauwhede, I.: Timing attacks on error correcting codes in post-quantum schemes. Cryptology ePrint Archive, Report 2019/292 (2019), https://eprint.iacr.org/2019/292
19. Ding, J., Fluhrer, S.R., RV, S.: Complete attack on RLWE key exchange with reused keys, without signal leakage. In: Susilo, W., Yang, G. (eds.) ACISP 18: 23rd Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, vol. 10946, pp. 467–486. Springer, Heidelberg, Germany, Wollongong, NSW, Australia (Jul 11–13, 2018)
20. Fabsic, T., Hromada, V., Stankovski, P., Zajac, P., Guo, Q., Johansson, T.: A reaction attack on the QC-LDPC McEliece cryptosystem. In: Lange, T., Takagi, T.

(eds.) Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017. pp. 51–68. Springer, Heidelberg, Germany, Utrecht, The Netherlands (Jun 26–28 2017)
21. Fluhrer, S.: Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085 (2016), https://eprint.iacr.org/2016/085
22. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 537–554. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999)
23. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) Advances in Cryptology – CRYPTO'97. Lecture Notes in Computer Science, vol. 1294, pp. 112–131. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 1997)
24. Greuet, A., Montoya, S., Renault, G.: Attack on LAC key exchange in misuse situation. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) CANS 20: 19th International Conference on Cryptology and Network Security. Lecture Notes in Computer Science, vol. 12579, pp. 549–569. Springer, Heidelberg, Germany, Vienna, Austria (Dec 14–16, 2020)
25. Guo, Q., Johansson, T.: A new decryption failure attack against HQC. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020, Part I. Lecture Notes in Computer Science, vol. 12491, pp. 353–382. Springer, Heidelberg, Germany, Daejeon, South Korea (Dec 7–11, 2020)
26. Guo, Q., Johansson, T., Nilsson, A.: A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 359–386. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020)
27. Guo, Q., Johansson, T., Stankovski, P.: A key recovery attack on MDPC with CCA security using decoding errors. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 789–815. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016)
28. Guo, Q., Johansson, T., Yang, J.: A novel CCA attack using decryption errors against LAC. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019, Part I. Lecture Notes in Computer Science, vol. 11921, pp. 82–111. Springer, Heidelberg, Germany, Kobe, Japan (Dec 8–12, 2019)
29. Hall, C., Goldberg, I., Schneier, B.: Reaction attacks against several public-key cryptosystems. In: Varadharajan, V., Mu, Y. (eds.) ICICS 99: 2nd International Conference on Information and Communication Security. Lecture Notes in Computer Science, vol. 1726, pp. 2–12. Springer, Heidelberg, Germany, Sydney, Australia (Nov 9–11, 1999)
30. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Third Algorithmic Number Theory Symposium (ANTS). Lecture Notes in Computer Science, vol. 1423, pp. 267–288. Springer, Heidelberg, Germany (Jun 1998)
31. Hoffstein, J., Silverman, J.H.: Protecting ntru against chosen ciphertext and reaction attacks. NTRU Cryptosystems Technical Report 16 (2000)

32. Huguenin-Dumittan, L., Vaudenay, S.: Classical misuse attacks on NIST round 2 PQC - the power of rank-based schemes. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) ACNS 20: 18th International Conference on Applied Cryptography and Network Security, Part I. Lecture Notes in Computer Science, vol. 12146, pp. 208–227. Springer, Heidelberg, Germany, Rome, Italy (Oct 19–22, 2020)
33. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. The deep space network progress report 42-44, Jet Propulsion Laboratory, California Institute of Technology (Jan/Feb 1978), https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF
34. Naehrig, M., Alkim, E., Bos, J., Ducas, L., Easterbrook, K., LaMacchia, B., Longa, P., Mironov, I., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: FrodoKEM. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
35. Okada, S., Wang, Y., Takagi, T.: Improving key mismatch attack on NewHope with fewer queries. In: Liu, J.K., Cui, H. (eds.) ACISP 20: 25th Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, vol. 12248, pp. 505–524. Springer, Heidelberg, Germany, Perth, WA, Australia (Nov 30 – Dec 2, 2020)
36. Qin, Y., Cheng, C., Ding, J.: A complete and optimized key mismatch attack on NIST candidate NewHope. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019: 24th European Symposium on Research in Computer Security, Part II. Lecture Notes in Computer Science, vol. 11736, pp. 504–520. Springer, Heidelberg, Germany, Luxembourg (Sep 23–27, 2019)
37. 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: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 13093, pp. 92–121. Springer (2021), https://doi.org/10.1007/978-3-030-92068-5_4
38. 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), 307–335 (2020), https://tches.iacr.org/index.php/TCHES/article/view/8592
39. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th Annual ACM Symposium on Theory of Computing. pp. 84–93. ACM Press, Baltimore, MA, USA (May 22–24, 2005)
40. 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 (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
41. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science. pp. 124–134. IEEE Computer Society Press, Santa Fe, NM, USA (Nov 20–22, 1994)
42. 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 Trans. Cryptogr. Hardw. Embed. Syst. 2022(1), 296–322 (2022), https://doi.org/10.46586/tches.v2022.i1.296-322

43. Xagawa, K., Ito, A., Ueno, R., Takahashi, J., Homma, N.: Fault-injection attacks against nist's post-quantum cryptography round 3 KEM candidates. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13091, pp. 33–61. Springer (2021), https://doi.org/10.1007/978-3-030-92075-3_2