Unlocking the True Potential of Decryption Failure Oracles: A Hybrid Adaptive-LDPC Attack on ML-KEM Using Imperfect Oracles

Qian Guo* *Lund University*
Denis Nabokov* *Lund University*
Thomas Johansson *Lund University*

Abstract

Side-channel attacks exploiting Plaintext-Checking (PC) and Decryption Failure (DF) oracles are a pressing threat to deployed post-quantum cryptography. These oracles can be instantiated from tangible leakage sources like timing, power, and microarchitectural behaviors, making them a practical concern for leading schemes based on lattices, codes, and isogenies. In this paper, we revisit chosen-ciphertext side-channel attacks that leverage the DF oracle on ML-KEM. While DF oracles are often considered inefficient compared to their binary PC counterparts in lattice-based schemes, we demonstrate that their full potential has been largely unrealized.

We introduce a novel attack framework that combines adaptive query generation with belief propagation for Low-Density Parity-Check (LDPC) codes. Our methodology crafts carefully balanced parity checks over multiple secret coefficients, maximizing the Shannon information extracted from each oracle query, even in the presence of significant noise. This approach dramatically reduces the number of queries required for a full key recovery, achieving near-optimal efficiency by approaching the theoretical Shannon information bound. For ML-KEM-768 with an oracle accuracy of 95%, our attack requires only 2 950 queries (a 1.35 ratio to the Shannon lower bound), establishing that a well-designed DF attack can surpass the efficiency of state-of-the-art binary PC attacks.

To validate the practical impact of our findings, we apply our framework to the recent GoFetch attack, showing significant gains in this real-world, microarchitectural side-channel scenario. Our method reduces the required measurement traces by over an order of magnitude and eliminates the need for computationally expensive post-processing, enabling a full key recovery on higher-security schemes previously considered intractable.

*Equal contribution, alphabetical order.

1 Introduction

The recent conclusion of the National Institute of Standards and Technology (NIST) post-quantum cryptography (PQC) standardization process has established CRYSTALS-Kyber as a primary global standard for public-key encryption and key establishment. Now formally standardized as the Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM) in FIPS 203 [25], its widespread deployment is imminent. This landmark decision has intensified research into its practical security, especially its resilience against side-channel attacks. Kyber's design follows a common paradigm in modern cryptography: a public-key encryption (PKE) scheme with chosen-plaintext attack (CPA) security is first constructed, which is then transformed into a key encapsulation mechanism (KEM) with chosen-ciphertext attack (CCA) security using a variant of the Fujisaki–Okamoto (FO) transform [10]. This transformation, while providing theoretical security, introduces new attack surfaces at the implementation level.

Among the various side-channel vectors, oracle-based attacks represent a particularly potent threat. This powerful methodology abstracts the physical leakage of a device into a conceptual black box, or "oracle", which reveals specific information about the internal state or outputs during cryptographic operations. By carefully crafting ciphertexts and feeding them to the decapsulation procedure, an attacker can use the oracle output to recover the secret key.

The concept of exploiting decryption outcomes in lattice-based cryptography can be traced back to early reaction attacks [17] and has evolved into a sophisticated taxonomy of oracles. The most frequently discussed are the decryption failure (DF) oracle and the plaintext-checking (PC) oracle. A general PC oracle [1, 41], given a chosen ciphertext $\tilde{c}$ and a message m, reveals whether the PKE decryption of $\tilde{c}$ equals m. A DF oracle [13] is similar to the general PC oracle, by viewing that the chosen ciphertext $\tilde{c}$ is typically (but not necessarily) modified from a valid ciphertext c corresponding to a plaintext m (the "reference plaintext"). The DF oracle, similar to the general PC oracle, tells whether the PKE decryption

result of $\tilde{c}$ is equivalent to the reference plaintext m. If the oracle returns 1, then it is a decryption success; otherwise, it is a decryption failure.

In the area of lattice-based cryptography, the term PC oracle by default refers to the binary PC oracle, a specialized variant introduced by D'Anvers et al. [5] and later utilized by Ravi et al. [34]. With this oracle, an attacker crafts ciphertexts such that the decrypted message m' depends on a single coefficient of the secret key, taking one of two distinct values—specifically, an all-zero vector versus a sparse vector with a single nonzero element. It is important to note that general PC oracles, DF oracles, and binary PC oracles extract at most 1 bit of information per oracle call. While more powerful variations exist, such as the multi-value PC oracle [15, 23, 31, 38, 40] and the full decryption oracle [16, 26, 33, 45], these advanced oracles require significantly more information from a single query, which limits their use-cases.

The abstraction of oracles offers a clean and effective separation between the physical discovery of a leakage and its mathematical exploitation. This allows researchers to concentrate on developing efficient key-recovery algorithms based on specific oracle types, while others focus on identifying physical leakages to instantiate those oracles, ultimately leading to generic attack methodologies applicable across lattice-based, code-based, and isogeny-based KEMs [41]. Furthermore, this approach facilitates the exploitation of a wide range of side-channel leakages. These leakages range from strong to extremely subtle and include timing discrepancies [2, 4, 5, 8, 12, 13, 28], power and electromagnetic emanations [7, 16, 26, 31, 33–35, 39–41, 45], and complex microarchitectural behaviors such as attacks exploiting dynamic frequency scaling [42], cache timing [22], contention on execution units due to variable-time instructions [36], data memory-dependent prefetchers [3], and contention on shared resources in trusted execution environments [11, 43].

Furthermore, in contrast to attacks targeting operations that directly manipulate secret values, such as the Number Theoretic Transform [29], oracle-based attacks are considerably more challenging to detect and mitigate. This difficulty arises because such attacks exploit information leakages that have an indirect and often obscured relationship with the secret key. Last, the implications of oracle-based attacks extend beyond the realm of traditional side-channel analysis, e.g., in key-mismatch and misuse attacks [1, 6, 9, 14, 30] against IND-CPA secure KEMs or in fault-injection attacks [24, 44]. This is because such attack vectors can be used to construct required PC-style oracles, effectively transforming the problem into one solvable with a PC oracle-based attack.

Motivation. Recent research into PC oracle-based side-channel attacks against lattice-based cryptographic schemes has largely followed two trends. The first focuses on optimizing query efficiency for *perfect* oracles. This line of work, initiated by Qin et al. [30], employs adaptive techniques, such as using Huffman coding on oracle outputs, to dramatically reduce the number of queries required for a key recovery. Subsequent work has refined this approach, achieving query counts that approach the theoretical Shannon limit [14]. The second trend confronts the more practical scenario of *inaccurate* oracles, which arise from environmental noise, measurement limitations, or active countermeasures. Here, methods have been proposed that combine adaptivity with error detection codes [39]. However, these approaches tend to lose efficiency as the oracle's error rate increases, as their reliance on techniques like majority voting requires a high number of traces to maintain accuracy. Non-adaptive approaches using Low-Density Parity-Check (LDPC) codes, such as the SCA-LDPC framework proposed in [16], offer strong error-correction capability. However, they lack the information gain from adaptive queries, and their evaluation on binary PC or DF oracles in the lattice-based setting has been limited.

Within this context, a prevailing belief has formed that binary PC oracles are fundamentally more powerful and efficient than DF oracles for lattice-based schemes (see e.g. [32]). This view, however, overlooks a crucial advantage of DF oracles: flexibility. This flexibility is twofold. First, certain physical leakages can naturally instantiate a DF oracle such as the timing leakage in [13]. Second, DF oracles are more general; they trigger when decryption fails to an *unspecified* message, whereas a binary PC oracle requires the decrypted message to match one of two precisely crafted, distinct values. This gives an attacker greater freedom in constructing chosen ciphertexts. Despite this, existing attacks based on DF oracles are often considered inefficient. For instance, a recent attack by Hermelink et al. [20] on ML-KEM-768 required 7 000 perfect DF oracle queries for a full key recovery—a query count 4.49 times the information-theoretic lower bound. In stark contrast, our method achieves a full key recovery with only 2 950 queries against a noisy oracle, demonstrating a dramatic leap in efficiency. The recent GoFetch attack [3] serves as a prime example of these challenges. While it successfully constructs a DF oracle against Kyber on Apple M-series CPUs, its practical efficiency is hampered by significant limitations. The attack can only partially recover secret key coefficients, necessitating a computationally expensive, multi-hour lattice reduction step for full key recovery. Furthermore, it requires a large number of traces to overcome measurement noise, highlighting the inefficiencies of current DF oracle exploitation techniques.

This paper challenges the perception of DF oracle inferiority. We investigate whether the DF oracle is truly less efficient than the binary PC oracle, or if its full potential has simply not been realized. This leads to the core problem we address: *How can we design a more query-efficient oracle-based attack, especially for inaccurate oracles, and how closely can we approach the information-theoretic lower bound?* Our work explores this question by developing a novel DF oracle-

based attack, demonstrating that a carefully constructed attack methodology allows DF oracles to achieve, and even surpass, the efficiency of their binary PC-based counterparts, particularly in noisy, realistic environments.

New Techniques. To answer the research questions posed, we introduce a novel attack methodology that combines the principles of adaptive querying with the powerful error-correction capabilities of LDPC decoding. Our technique is centered on two core concepts.

First, we design queries that create high-information parity checks involving multiple secret key coefficients. Instead of targeting individual coefficients, which often leads to unbalanced and less informative oracle outputs, our method constructs linear combinations of several secret coefficients. This approach is powerful for two reasons: 1) it directly generates the parity checks required by an LDPC decoder, and 2) it allows us to maximize the information gained from each query. By carefully selecting the coefficients and a corresponding threshold, we can balance the oracle's output distribution (i.e., make the probability of success or failure close to 0.5). This technique effectively functions as a randomness extractor, ensuring each oracle call yields close to the channel capacity, a maximum of Shannon entropy one can obtain with the noisy oracle.

Second, our attack is structured as a multi-phase, adaptive framework. The attack begins by issuing a series of these information-maximized queries. The noisy oracle responses are then processed by a belief propagation decoder, which computes a posterior probability distribution for each targeted secret coefficient, simultaneously aggregating information and correcting errors. The crucial step is the adaptive nature of the subsequent phase: based on these posterior probabilities, the attacker generates new, targeted queries designed specifically to resolve the highest remaining uncertainties. This adaptive loop—querying, decoding, and re-adapting—maximizes efficiency at each step. This hybrid approach overcomes the fragility of purely adaptive methods in noisy settings while improving the query count compared to purely non-adaptive schemes. The combination allows us to significantly reduce the number of traces for a full key recovery, demonstrating that DF oracles can be highly efficient.

Research Contributions. Our core contributions are:

- We introduce a novel hybrid attack framework that combines adaptive querying with belief propagation for Low-Density Parity-Check (LDPC) codes. Our method crafts ciphertexts that generate balanced, high-information parity checks over multiple secret key coefficients. This approach maximizes the Shannon information extracted from each query, proving highly effective even when interacting with imperfect or noisy oracles.
- We demonstrate that a well-designed Decryption Failure (DF) oracle attack can achieve, and in noisy environments surpass, the efficiency of state-of-the-art binary Plaintext-Checking (PC) oracle attacks. Our extensive simulations show a dramatic reduction in the required number of oracle queries. For instance, on ML-KEM-768 with an oracle accuracy of 95%, our attack requires only 2 950 queries for a full key recovery (a ratio of just 1.35 to the lower Shannon's bound). A detailed comparison with previous works is presented in Table 1.
- We validate the practical impact of our findings by applying our framework to the GoFetch attack scenario [3]. Our methodology achieves significant gains in this real-world setting, overcoming previous limitations. We reduce the required measurement traces by more than an order of magnitude, eliminate the need for computationally expensive post-processing, and enable the practical full key recovery on higher-security schemes, which was intractable with the original GoFetch method.

The source code is available at https://doi.org/10.5281/zenodo.17899501.

Outline. The remainder of this paper is organized as follows. Section 2 provides the necessary background on ML-KEM and defines the threat model. Section 3 details our novel key-recovery methodology. In Section 4, we apply our framework to the GoFetch attack scenario to demonstrate its practical effectiveness. We then discuss the broader implications of our findings in Section 5. Finally, Section 6 concludes the paper and outlines directions for future research.

2 Preliminaries

In this section, we provide the necessary background, covering the fundamental aspects of the ML-KEM scheme and outlining the threat model assumed in this work.

Notation. We use lowercase bold letters (e.g., $\mathbf{v}$) for polynomials or vectors, and uppercase bold letters (e.g., $\mathbf{A}$) for matrices. For a polynomial $\mathbf{v}$, its i-th coefficient is denoted by $\mathbf{v}[i]$. All arithmetic on polynomial coefficients is performed modulo q. Polynomials reside in the ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n+1)$, where $n = 256$. We represent values from $\mathbb{Z}_q$ as $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$ for odd q. For a vector $\mathbf{s} = (\mathbf{s}_0, \ldots, \mathbf{s}_{d-1})$ composed of d polynomials from $\mathcal{R}_q$, we abuse notation and write $\mathbf{s}[i]$ for $0 \leq i < d \cdot n$ to refer to $\mathbf{s}_{\lfloor i/n \rfloor}[i \bmod n]$.

We use H_2 for the Shannon binary entropy and Ber_ρ for the Bernoulli distribution with parameter ρ.

Table 1: Comparison of binary PC and DF Oracle-Based Key-Recovery Attacks on ML-KEM.

Work	Oracle Accuracy Level (ρ)	# Oracle Calls (Approx.)	Theoretical Lower Bound[b]	Ratio[c]
Scheme: ML-KEM-512				
Hermelink et al. [21] (2021)	Perfect (100%)	5 750	1 195	4.81
Hermelink et al. [18] (2023)	100%; 90%	5 500; 12 000	1 195; 2 250	4.6; 5.33
Ueno et al. [41] (2022)	96%	7 680	1 577	4.87
Shen et al. [39] (2023)[d]	95%	3 875	1 674	2.31
This Work	**95%**	**2 400**	**1 674**	**1.43**
Scheme: ML-KEM-768				
Hermelink et al. [21] (2021)	Perfect (100%)	6 750	1 560	4.33
This Work	**95%**	**2 950**	**2 186**	**1.35**

[b]Calculated as $H_2(\mathsf{sk})/(1-H_2(\rho))$, where $H_2(\mathsf{sk})$ is the Shannon entropy of the full secret key ($\approx 1\,195$ bits for ML-KEM-512, $\approx 1\,560$ bits for ML-KEM-768) and $1-H_2(\rho)$ is the maximum information per query with noise level ρ.
[c]Ratio of Oracle Calls to the Theoretical Lower Bound. A lower ratio indicates higher efficiency.
[d]This is a follow-up for [14, 30] that extends Huffman coding approach to the noisy oracles. While Huffman coding is the most efficient method for perfect oracles, the focus of this paper is on the noisy case.

2.1 ML-KEM (Kyber)

ML-KEM, previously named CRYSTALS-Kyber [37], is a key encapsulation mechanism (KEM) selected by the U.S. National Institute of Standards and Technology (NIST) as a primary standard for post-quantum cryptography [25]. The scheme's security is based on the hardness of the Module Learning with Errors (MLWE) problem.

To achieve security against chosen-ciphertext attacks (IND-CCA), ML-KEM employs a variant of the Fujisaki-Okamoto (FO) transform [10]. This transform converts an IND-CPA secure public-key encryption scheme (denoted ML-KEM.CPAPKE) into an IND-CCA secure KEM (denoted ML-KEM.CCAKEM).

The core procedures for key generation, encapsulation, and decapsulation are detailed in Algorithm 1. These descriptions are simplified for clarity, omitting implementation-level optimizations like the Number Theoretic Transform (NTT). The complete specifications are available in the official documentation [25].

In ML-KEM, a smart design includes compression and decompression functions that reduce ciphertext sizes by operating coefficient-wise on polynomials. They are defined as follows:

Definition 1 (Compression). *The function* $\mathsf{Comp}_q(x,d) : \mathbb{Z}_q \to \mathbb{Z}_{2^d}$ *is defined as:*

$$\mathsf{Comp}_q(x,d) = \left\lfloor \frac{2^d}{q} \cdot x \right\rceil \pmod{2^d}. \tag{1}$$

Definition 2 (Decompression). *The function* $\mathsf{Decomp}_q(x,d) : \mathbb{Z}_{2^d} \to \mathbb{Z}_q$ *is defined as:*

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

Table 2: Parameter sets for ML-KEM [37]

Scheme	n	d	q	η_1	η_2	(d_u, d_v)
ML-KEM-512	256	2	3329	3	2	(10, 4)
ML-KEM-768	256	3	3329	2	2	(10, 4)
ML-KEM-1024	256	4	3329	2	2	(11, 5)

The scheme also employs hash functions $\mathcal{J}$, $\mathcal{G}$, and $\mathcal{H}$. ML-KEM is specified for three security levels—ML-KEM-512, ML-KEM-768, and ML-KEM-1024—each with a unique parameter set, as detailed in Table 2. All variants use the prime modulus $q = 3329$. The module rank, d, is set to 2, 3, or 4 for the respective security levels. Secret and error polynomials are sampled from a centered binomial distribution, $\mathcal{B}_\eta$, parameterized by η_1 and η_2.

2.2 The Threat Model

In our threat model, we consider a side-channel adversary whose goal is to recover the secret key from an ML-KEM implementation. The adversary has the capability to perform a chosen-ciphertext attack: they can submit crafted ciphertexts to the decapsulation device and observe physical side-channel information (e.g., timing, power) emitted during the computation. We abstract this physical leakage into a conceptual tool known as an "oracle", which the adversary queries to gain information about the secret.

This oracle-based approach is a powerful and widely used methodology in side-channel analysis. It creates a clean separation between the physical discovery of a leakage and its mathematical exploitation. This allows for the development of generic key-recovery algorithms that are applicable across a

Algorithm 1 A Simplified View of the ML-KEM

ML-KEM.CCAKEM.KeyGen()

Output: sk', pk

$z \xleftarrow{\$} \{0,1\}^{256}$
$(\mathbf{A}, \mathbf{s}, \mathbf{e}) \xleftarrow{\$} \mathcal{R}_q^{d \times d} \times \mathcal{B}_{\eta_1}^d \times \mathcal{B}_{\eta_1}^d$
$\mathbf{b} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}$
$pk \leftarrow (\mathbf{b}, \rho)$, where ρ is the seed for $\mathbf{A}$
$sk \leftarrow \mathbf{s}$
$sk' \leftarrow (sk, pk, \mathcal{H}(pk), z)$
return (pk, sk')

ML-KEM.CCAKEM.Encaps(pk)

Input: $pk = (\mathbf{b}, \rho)$
Output: K, ct

$\mathbf{m} \xleftarrow{\$} \{0,1\}^{256}$
$(K, r) \leftarrow \mathcal{G}(\mathbf{m}, \mathcal{H}(pk))$
$\mathbf{A} \leftarrow \mathsf{Gen}(\rho)$
$\mathbf{r} \xleftarrow{\$} \mathcal{B}_{\eta_1}^d; \mathbf{e}_1 \xleftarrow{\$} \mathcal{B}_{\eta_2}^d; e_2 \xleftarrow{\$} \mathcal{B}_{\eta_2}$
$\mathbf{u} \leftarrow \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$
$v \leftarrow \mathbf{b}^T\mathbf{r} + e_2 + \mathsf{Decomp}_q(\mathbf{m}, 1)$
$c_1 \leftarrow \mathsf{Comp}_q(\mathbf{u}, d_u)$
$c_2 \leftarrow \mathsf{Comp}_q(v, d_v)$
$ct \leftarrow (c_1, c_2)$
return (K, ct)

ML-KEM.CCAKEM.Decaps(sk', ct)

Input: $sk' = (sk, pk, h, z)$, $ct = (c_1, c_2)$
Output: K'

$\mathbf{u}' \leftarrow \mathsf{Decomp}_q(c_1, d_u)$
$v' \leftarrow \mathsf{Decomp}_q(c_2, d_v)$
$\mathbf{m}' \leftarrow \mathsf{Comp}_q(v' - sk^T\mathbf{u}', 1)$
$(K', r') \leftarrow \mathcal{G}(\mathbf{m}', h)$
$\bar{K} \leftarrow \mathcal{J}(z, ct)$
$ct' \leftarrow ML-KEM.\mathsf{CPAPKE.Enc}(pk, \mathbf{m}', r')$
if $ct \neq ct'$ **then**
    $K' \leftarrow \bar{K}$
end if
return K'

wide range of cryptographic schemes and physical platforms.

The oracles central to attacks on PQC KEMs such as ML-KEM, including the ones discussed in this paper, are generally categorized as follows:

- **Binary Plaintext-Checking (PC) Oracle:** This is the most common oracle model for lattice-based schemes that allows for a given ciphertext $\tilde{c}$ to obtain information about the corresponding decrypted plaintext m'. An attacker crafts ciphertexts such that m' depends on a single coefficient of the secret key. This causes m' to take one of two distinct, predictable values, for instance, "the decrypted plaintext will be 0" (an all-zero vector) or "a unit vector $0^{i-1}\|1\|0^{l-i-1}$" (a single 1 at a specific position, with all other bits being 0). The side-channel leakage, therefore, acts as a simple binary distinguisher for a secret coefficient. More powerful, though less common, variants like the multi-value PC oracle also exist.

- **General PC and Decryption-Failure (DF) Oracle:** A more general primitive is an oracle that, given a ciphertext $\tilde{c}$ and a reference message m, reveals whether the decrypted plaintext m' is equal to m. In the context of FO-transformed KEMs, the General PC oracle and the DF oracle effectively collapse into the same side-channel observation [41] and can be treated as mathematically equivalent. In this work, we adopt the term *DF Oracle* for this primitive. We do so for two reasons: (1) to clearly distinguish our approach from the restrictive binary PC oracle described above, and (2) to align with the physical reality of our target scenario (GoFetch [3]), where the leakage manifests as a system-level failure (e.g., a pointer dereference fault) when decryption yields an unexpected value.

As both the binary PC and DF oracles described here provide at most one bit of information per query, we collectively refer to them as *one-bit-bound* oracles.

These abstract one-bit-bound oracle models are remarkably generic; key-recovery algorithms based on them are applicable not only to lattice-based schemes but also to code-based and isogeny-based KEMs. Furthermore, they are versatile and can be instantiated by exploiting a wide spectrum of side-channel leakages—from strong to extremely subtle—including timing discrepancies [2, 4, 5, 8, 12, 13, 28], power and electromagnetic emanations [7, 16, 26, 31, 33–35, 39–41, 45], and complex microarchitectural behaviors [3, 11, 22, 36, 42, 43]. While the oracle abstraction is general, its concrete instantiation is highly platform-dependent, relying on the specific leakage characteristics of the target's hardware and software. In this work, we assume the existence of such an oracle to simulate the performance of the key recovery; its instantiation within the GoFetch attack scenario will be detailed in Section 4.

3 New Key-Recovery Approach for ML-KEM

This section outlines our new key-recovery strategy for ML-KEM while drastically reducing the number of oracle queries.

3.1 Information-Theoretic Foundations

As established in Section 2.2, a chosen oracle returns a single binary value. Because of noise, imperfect encodings, and conditional mutual information, the amount of Shannon information we obtain per query is generally *strictly less than one bit*. Fewer bits per query translate directly into more queries before the secret is recovered, so we are motivated to maximize the information carried by each oracle response.

Setting. Let S denote the random variable we wish to learn; for concreteness, you may think of $S = \mathbf{s}[i] \in \{-2, -1, 0, 1, 2\}$[1] when we deal with ML-KEM-768. Because

[1] Later in the attack, we group together several coefficients, but setting with one coefficient should be sufficient for the present discussion.

S is finite, we can list all its possible values in a table together with their probabilities. A side-channel adversary crafts a ciphertext that leads to the computation of a predicate

$$X \;=\; f(S) \;\in\; \{0,1\},$$

that is detectable via side-channels. This function we call an *encoding* of S. For instance, one may take $f(S) = [\mathbf{s}[i] > 0]$, yielding the truth vector $(0,0,0,1,1)$ for $\mathbf{s}[i] \in \{-2,-1,0,1,2\}$.

Noisy Oracle. In practice, the oracle is noisy: instead of X we observe $Y = X \oplus E$, where $E \leftarrow\!\$\, \mathsf{Ber}_{1-\rho}$ is independent of S and the oracle accuracy $\rho > \frac{1}{2}$. Upon seeing Y we gain the mutual information

$$I(S;Y) \;=\; H(S) - H(S \mid Y).$$

Maximizing $I(S;Y)$ therefore minimizes the expected number of queries. The next lemma shows that information is maximized when the observable output Y is *balanced*.

Lemma 1 (Balanced leakage maximizes information)**.** *Fix the oracle accuracy $\rho \in (\frac{1}{2}, 1]$ and let $f\colon \mathrm{supp}(S) \to \{0,1\}$ be an arbitrary predicate. Write $q := \Pr[f(S) = 1]$ and $r := \Pr[Y = 1] = (1-\rho) + (2\rho - 1)q$. Then*

$$I(S;Y) \;=\; H_2(r) - H_2(1-\rho) = H_2(r) - H_2(\rho),$$

where H_2 is the binary entropy function. The quantity $I(S;Y)$ achieves its maximum $1 - H_2(\rho)$ when and only when $r = \frac{1}{2}$.

Proof. Because $X = f(S)$ is deterministic, $H_2(Y \mid S) = H_2(Y \mid X)$ and hence

$$I(S;Y) = H_2(Y) - H_2(Y \mid S) = H_2(Y) - H_2(Y \mid X) = I(X;Y).$$

Maximizing $I(X;Y)$ therefore *also* maximizes $I(S;Y)$, and it is enough to work with the simpler binary random variable X.

For a binary-symmetric channel with crossover probability $1-\rho$, we have $I(X;Y) = H_2(Y) - H_2(Y \mid X) = H_2(r) - H_2(1-\rho)$, where $r = (1-\rho) + (2\rho-1)q$. The second term is constant, and $H_2(r)$ is strictly concave with a unique maximum at $r = \frac{1}{2}$. Hence $I(X;Y)$, and therefore $I(S;Y)$, is maximized when $r = \frac{1}{2}$. □

Implications for Attack Design. The preceding lemma says that every oracle call can obtain at most $1 - H_2(\rho)$ bits of information; this upper bound is achieved *only* when the observable bit Y is perfectly balanced. Note here that if $\Pr[X = 1]$ is close to 0.5, then $\Pr[Y = 1]$ is even closer to 0.5, thus, finding a good balanced encoding guarantees that Y is well-spread. So, for now we focus on X for clarity, but our attack searches for encodings maximizing how balanced Y is, i.e., it depends on the accuracy of the oracle ρ.

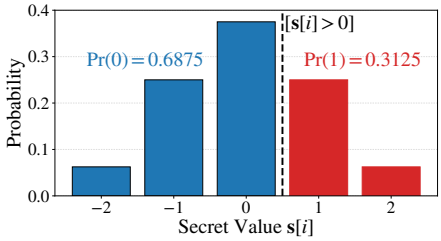


Figure 1: An example of a simple highly unbalanced encoding $[\mathbf{s}[i] > 0]$ on a single secret coefficient for ML-KEM-768

One of our goals is to select such a balanced encoding. Unfortunately, one can not choose any binary string for an encoding: the set of possible encodings is dictated by the oracle itself, for example, in a typical binary PC oracle attack, the message is forced to be all zero except for a single bit. This leads to encodings of the form $\mathbf{s}[i] > t$ or $\mathbf{s}[i] < t$ for $t \in \{-2,1,0,1,2\}$ that are generally poorly balanced. See example in Figure 1.

In the following we show a new construction that allows much more balanced encodings which leads to a better information per oracle call extraction and reduction of the number of traces.

3.2 Creating Basic Chosen Ciphertexts

Our main idea is to make the oracle output depend not on a single coefficient of the secret key, but on several of them simultaneously. Specifically, we define the target random variable S as an ω-dimensional vector composed of $\omega > 1$ chosen coefficients from the secret key $\mathbf{s}$. The random variable S can thus take on up to $|\mathcal{B}_{\eta_1}|^{\omega}$ distinct vector values. This vastly larger support space alleviates the problem of probability being concentrated on a single value, which is crucial for creating well-balanced encodings.

This approach extends the work of [16], which proposed targeting *the sum* of several secret coefficients. While targeting the sum also improves the balancing problem, it leads to an unavoidable loss of information through aliasing. For instance, the secret vectors $(0,-1)$ and $(-1,0)$ are indistinguishable as their sum is -1 in both cases. Our method is more general as it targets the joint distribution of the vector directly, allowing us to potentially distinguish every possible outcome. This increased granularity, however, comes at the price of a more computationally intensive key-recovery phase.

Let us briefly revisit the decapsulation procedure. The secret key $\mathbf{s}$ consists of d polynomials, $\mathbf{s} = (\mathbf{s}_0, \ldots, \mathbf{s}_{d-1})$, where $\mathbf{s}_i \in \mathcal{R}_q$. Similarly, for a ciphertext $(\mathbf{u}, \mathbf{v})$, we have $\mathbf{u} = (\mathbf{u}_0, \ldots, \mathbf{u}_{d-1})$ for $\mathbf{u}_i, \mathbf{v} \in \mathcal{R}_q$. The message is computed as $\mathbf{m}' \leftarrow \mathsf{Comp}_q(\mathbf{v}' - \mathbf{s}^T\mathbf{u}', 1)$, where $\mathbf{u}'$ and $\mathbf{v}'$ are obtained via application of the Decomp_q function. The message bit $\mathbf{m}[j]$ is set to 0 if $(\mathbf{v}' - \mathbf{s}^T\mathbf{u}')[j]$ is closer to 0 (modulo q) and

to 1 otherwise. The switch happens at $\pm(\lfloor \frac{q}{4} \rfloor + 1)$. Thus, we always choose $\mathbf{u}$ in a way so that for each coefficient c of $\mathbf{s}^T\mathbf{u}'$ we have $|c| \leq \lfloor \frac{q}{4} \rfloor$. This guarantees that the decrypted message would be exactly $\mathbf{m}$ except for positions with additional noise in $\mathbf{v}$.

More concretely, let $\mathbf{v}' = \mathsf{Decomp}_q(\mathbf{m}, 1) + \delta$, then the input to Comp_q at position $0 \leq j < n$ is

$$(\mathbf{v}' - \mathbf{s}^T\mathbf{u}')[j] = \mathbf{m}[j] + \underbrace{\delta[j]}_{\text{additional noise}} - \underbrace{(\mathbf{s}^T\mathbf{u}')[j]}_{\text{secret dependent}}.$$

A decryption failure for positive $\delta[j]$ happens if

$$\delta[j] - (\mathbf{s}^T\mathbf{u}')[j] > \left\lfloor \frac{q}{4} \right\rfloor,$$

or, rearranging,

$$(\mathbf{s}^T\mathbf{u}')[j] < \delta[j] - \left\lfloor \frac{q}{4} \right\rfloor.$$

Similarly, for negative $\delta[j]$ we have $(\mathbf{s}^T\mathbf{u}')[j] > \delta[j] + \lfloor \frac{q}{4} \rfloor$. Due to the choice of q and construction of Decomp_q function, the value $\delta[j] \pm \lfloor \frac{q}{4} \rfloor$ is a multiple of $\left\lfloor \frac{q}{2^{d_v}} \right\rfloor$. Thus, by carefully choosing the added noise at position j, we can trigger a decryption failure if

$$\begin{aligned} (\mathbf{s}^T\mathbf{u}')[j] &> k \cdot \left\lfloor \frac{q}{2^{d_v}} \right\rfloor, \text{ or} \\ (\mathbf{s}^T\mathbf{u}')[j] &< -k \cdot \left\lfloor \frac{q}{2^{d_v}} \right\rfloor, \end{aligned} \tag{3}$$

where $k \in \{0,1,2,3\}$. In other words, the decryption failure at position j happens when one of the inequalities from 3 is satisfied. Which inequality we look at depends on the sign of the added noise; both can not be satisfied at the same time.

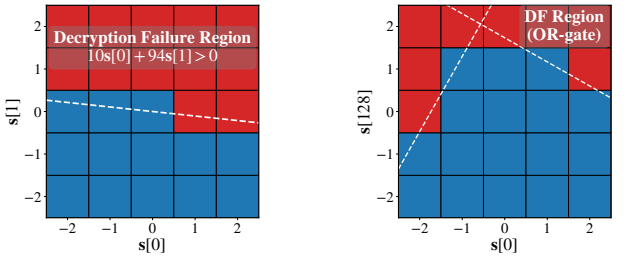
The first type of encoding we work on is created when exactly one value of δ is nonzero at position j and $\mathbf{u}$ has ω nonzero values at arbitrary positions. Note that setting $\mathbf{u}_0[i_1] = z$, results in $(\mathbf{s}^T\mathbf{u}')[j]$ having a value

$$\begin{cases} -z \cdot s_0[j - i_1], \text{ if } i_1 \leq j, \\ z \cdot s_0[n - i_1 + j], \text{ otherwise} \end{cases}$$

due to multiplication in $\mathcal{R}_q$. This notation can easily become cumbersome; thus, instead of describing how to choose $\mathbf{u}$, we focus on the inequality determining if decryption failure is triggered. Then this encoding type could be described by a single inequality (or a predicate)

$$z_0\mathbf{s}[i_0] + \ldots + z_{\omega-1}\mathbf{s}[i_{\omega-1}] \lessgtr T, \tag{4}$$

where indices $(i_0, \ldots, i_{\omega-1})$ could be chosen freely, T is the multiple of $\left\lfloor \frac{q}{2^{d_v}} \right\rfloor$, the comparison sign (greater or less) depends on the sign of T and values $(z_0, \ldots, z_{\omega-1})$ are from image of $\mathsf{Decomp}_q(\cdot, d_u)$ with additional restriction



(a) Encoding over linear combination of ω secret coefficients

(b) Combination of several inequalities

Figure 2: A visualization of encoding types that are used in the key-recovery process for $\omega = 2$. (a) Basic encoding that is represented by a single inequality, geometrically is determined by a hyperplane, separating ω-dimensional space into two regions, (b) using DF oracle, we can combine multiple inequalities/hyperplanes, implementing logical OR over these inequalities to achieve more balanced encodings.

$|\eta_1 \cdot \sum_{i=0}^{\omega-1} z_i| < \lfloor \frac{q}{4} \rfloor$ to avoid triggering decryption failure for positions different from j. So, in words, we can create a ciphertext that, with the help of the oracle, should reveal information about the linear combination of ω secret coefficients, in particular, where this linear combination is greater or less than some threshold value of our choice. See a low-dimensional example in Figure 2a.

The random variable S is then $S = (\mathbf{s}[i_0], \ldots, \mathbf{s}[i_{\omega-1}])$. The complexity of our key-recovery process depends on the support size of S, which is $|\mathcal{B}_{\eta_1}|^\omega$. Generally, a larger ω allows for more balanced encodings at the price of exponentially slower key recovery. We choose $\omega = 4$ for our attack as it provides a good trade-off, and an additional reason is revealed in Section 3.3.

3.3 Utilizing DF Benefits: Complex Encoding

Even though the truth table for S is quite large (i.e., $7^4 = 2401$ for ML-KEM-512 and $5^4 = 625$ for ML-KEM-768 and ML-KEM-1024), the encoding we can obtain from Section 3.2 is still highly structural as it is a hyperplane separating vectors in two classes and, for example, if the inequality $(\mathbf{s}^T\mathbf{u}')[j] > 0$ is satisfied for some $\mathbf{s}[i] = 1$ where $\mathbf{s}[i]$ is part of S, then it would also be satisfied for $\mathbf{s}[i] = 2$. This is rather unfortunate since the more structure we have, the less likely to have a well-balanced encoding.

We propose a new idea that allows us to increase the number of possible encodings that are less structured, which allows a more efficient (in number of oracle calls) key-recovery process. The idea is based on the benefits that the DF oracle provides, specifically, that it can detect if the decrypted message differs from the intended message at *any* position. In Section 3.2, similarly to a typical binary PC oracle attack, we

choose only one nonzero position in δ. But now we allow $\mathbf{m}'$ to be different from $\mathbf{m}$ at *a few* positions.

As was shown previously, after fixing $\mathbf{u}$ to have exactly ω nonzero positions and setting $\delta[j]$ to be nonzero, we get an inequality over ω secret coefficients. Therefore, we can choose $\delta[j_1]$ and $\delta[j_2]$ to be nonzero, then the decryption failure would occur if the inequality at position j_1 *or* the inequality at position j_2 is satisfied. However, a naive choice of j_1 and j_2 would result in an encoding over two disjoint sets of secret coefficients, increasing the dimension of S to 2ω and making key recovery much more computationally heavy.

Luckily, the negacyclic ring structure of Kyber provides a powerful solution. By carefully selecting the nonzero positions in both $\mathbf{u}'$ and the noise δ to be related by cyclic shifts, we can force all the resulting inequalities to depend on the exact same set of ω secret coefficients. We now give an example that is easily generalizable. Set

$$\begin{aligned}\mathbf{u}'_0 &= z_0 + z_1 x^{64} + z_2 x^{128} + z_3 x^{192} \\ \delta &= \delta_0 + \delta_1 x^{64} + \delta_2 x^{128} + \delta_3 x^{192},\end{aligned}$$

while other polynomials of $\mathbf{u}$ are $\mathbf{0}$. A careful inspection of the computation of $\mathbf{v}' - \mathbf{s}^T\mathbf{u}'$ reveals the 4 inequalities:

$$\begin{bmatrix} z_0\mathbf{s}_0[0] + z_1\mathbf{s}_0[64] + z_2\mathbf{s}_0[128] + z_3\mathbf{s}_0[192] \lessgtr T_0 \\ -z_3\mathbf{s}_0[0] + z_0\mathbf{s}_0[64] + z_1\mathbf{s}_0[128] + z_2\mathbf{s}_0[192] \lessgtr T_1 \\ -z_2\mathbf{s}_0[0] - z_3\mathbf{s}_0[64] + z_0\mathbf{s}_0[128] + z_1\mathbf{s}_0[192] \lessgtr T_2 \\ -z_1\mathbf{s}_0[0] - z_2\mathbf{s}_0[64] - z_3\mathbf{s}_0[128] + z_0\mathbf{s}_0[192] \lessgtr T_3 \end{bmatrix}$$

where threshold T_i depends on $\delta[64i]$. Instead of a single inequality over ω variables, the oracle now depends on ω inequalities over the same variables. The coefficients of the linear combination could be chosen freely for the first inequality, later we apply a negacyclic shift. Notably, despite some limitations on the linear combination coefficients, we still can choose T_i freely and independently which greatly increases the flexibility of constructing an encoding and, consequently, how balanced it is. The visualization of this approach for $\omega = 2$ is shown in Figure 2b.

This approach implicitly uses the fact that ω has to be a divisor of n. And since $n = 256$, ω has to be a power of 2, $\omega = 4$ is a reasonable choice as $\omega \geq 8$ gets too expensive. This is because the runtime of our key-recovery scales with $|\mathcal{B}_{\eta_1}|^{\omega}$.

The idea is generalizable in a way that we can choose which block of secret key to target, then we set $\mathbf{u}_i$ to have nonzero values at positions $(j_1, j_1+64, j_1+128, j_1+192)$, while for $\mathbf{v}$ at positions $(j_2, j_2+64, j_2+128, j_2+192)$ for $0 \leq j_1, j_2 < n$. This leads to 4 inequalities over 4 secret coefficients from $\mathbf{s}_i$. We call this type of encoding *full rotation encoding*. See more details and examples in Appendix A.

3.4 Key-Recovery Strategy

In Sections 3.2 and 3.3 it was shown how to obtain a single bit from an oracle that maximizes the amount of information we learn. In order to recover the whole secret key, many more bits are required. We can define a lower bound on the number of required oracle calls using channel capacity and entropy of the key as $H_2(\mathbf{s})/(1-H_2(\rho))$. In this Section, we discuss the strategy of combining several calls to maximize information gain.

3.4.1 Multibit Encoding

The first straightforward approach is to keep the subset of secret coefficients fixed, then several different encodings are used. Thus, each vector of the random variable S is encoded with several bits. The estimation of the received information in this case essentially follows Section 3.1. Calling oracle k times on S gives us $\mathbf{y}$, which is a binary string of length k, so that we obtain multibit encoding of S. If $\mathbf{y}$ is uniformly distributed, then we obtain the maximum possible amount of information $k(1-H_2(\rho))$, thus, the goal of making $\mathbf{y}$ balanced stays the same, but now instead of a single bit we try to balance k bits.

The common approach is to use something akin to Hamming adaptive encoding, where the first bit is chosen to be as balanced as possible, while the choice of the subsequent bit encoding depends on the previous bits, again trying to make a balanced bit. This adaptive approach, however, struggles when the accuracy of the oracle ρ drops since incorrect early bit negatively affects subsequent choices. Therefore, as our approach is designed to work with inaccurate oracles, we try to limit adaptivity for now.

We create multibit encodings for full rotation checks that were described in Section 3.3 as they provide the most flexibility and create the most balanced encodings. Ideally, we just put together k balanced encodings and obtain close to $k(1-H_2(\rho))$ amount of information. Unfortunately, the encodings are not independent in general. Even if each bit is perfectly balanced on its own, the bitstring $\mathbf{y}$ is not uniform. This leads to an interesting scenario: the more bits we add to a multibit encoding of S, the less amount of information each subsequent bit provides, as we are slowly exhausting nearly independent encodings. We stress again that each bit encoding is fixed in advance before the attack starts.

We search for this multibit encoding first by randomly sampling 10 000 balanced full rotation 1-bit encodings, then we greedily try to combine them with a goal to get as uniform $\mathbf{y}$ as possible. In our experiments, we found 7 bits to be a good cutoff. For $\rho = 0.95$, the theory suggests the upper bound of information to be 4.99522 bits, while we can get (on average) 4.66449 bits for ML-KEM-512 and 4.67581 bits for ML-KEM-768, i.e., we lose about 0.3 bits of information per 7 calls.

We split all secret coefficients into $\frac{n}{\omega}d$ sets of ω coefficients. Then, for each set, we call the oracle 7 times. But it is not

enough to recover the whole secret. Adding more bits to multi-bit encoding gives diminishing returns, a different approach is needed to maximize further extracted information.

3.4.2 Batches: New Adaptive Approach

Since the multibit full rotation encodings were applied to independent subsets of $\mathbf{s}$, the total information gathered in this initial phase is simply the sum of the mutual information from each subset. However, to resolve the remaining uncertainty, we must introduce new encodings that link these previously independent groups, as illustrated in Figure 3. A significant challenge arises here: estimating the information gain from a new linking query is nontrivial, as it is statistically dependent on 16 previous secret coefficients. A direct computation of the overall mutual information quickly becomes intractable.

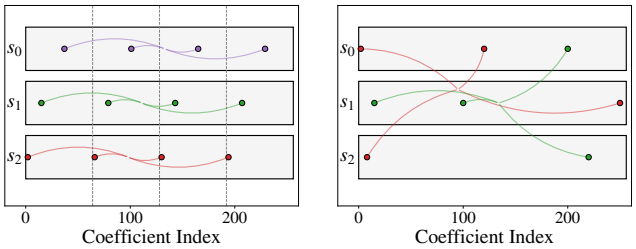
To overcome this, we propose an adaptive, iterative strategy. After the initial phase of full rotation encodings, we compute a posterior probability mass function (PMF) for each secret coefficient based on the oracle outputs. This belief state then guides the selection of new queries. For any given set of four coefficients, we can efficiently search for a nearly optimal balanced encoding. We achieve this by checking a pre-selected set of approximately 12 000 candidate encodings and, based on the current posterior PMFs, quickly selecting the one that yields the most balanced oracle response, thereby maximizing the expected information gain. To manage this process, we again partition all secret coefficients into $\frac{n}{\omega}d$ disjoint subsets and apply one such adaptively chosen encoding to each. We define this full set of $\frac{n}{\omega}d$ new queries as a *batch*.

To integrate the information from this new batch and update our belief state, we employ a belief propagation (LDPC) decoder, following the SCA-LDPC framework [16]. The decoder processes the responses from *all* oracle queries made so far—both the initial full rotation encodings and all subsequent batches. This not only combines the information but also provides powerful error-correction. The output is a new, more refined set of marginal PMFs for each secret coefficient. The attack proceeds by iteratively adding new batches of adaptively generated queries and running the LDPC decoder, progressively reducing the uncertainty until the secret key is recovered.

3.4.3 Hybrid Adaptive Algorithm Using LDPC

The complete hybrid attack strategy, combining the initial information gathering with the adaptive batch-based approach, is outlined in Algorithm 2. The process begins by querying the oracle with a precomputed set of 7-bit full rotation encodings to establish a baseline of information. Based on these initial responses, we compute a preliminary posterior probability mass function (PMF) for each secret coefficient using a maximum-likelihood approach.

The attack then enters an iterative loop, where in each round, or *batch*, we partition the secret coefficients and apply new, adaptively chosen single-bit basic encodings to link them. The responses from each batch are collected and then processed, along with all previous data, by an LDPC decoder. This step updates the PMFs for all coefficients, refining our belief state and leveraging the error-correcting properties of the decoder. This loop continues for a fixed number of batches.



(a) Full Rotation Encodings (b) Basic "Linking" Encodings

Figure 3: Illustration of the two-phase query strategy. (a) In the initial phase, full rotation encodings create strong, independent checks over structured, disjoint groups of coefficients (e.g., within $\mathbf{s}_0$, $\mathbf{s}_1$, etc.). (b) In the batch phase, basic encodings create flexible "linking" checks that connect arbitrarily chosen coefficients across the different groups.

To evaluate the effectiveness of our strategy, we ran simulations for both ML-KEM-512 and ML-KEM-768 using an oracle with $\rho = 0.95$. The results, averaged over 100 keys, are presented in Figure 4. We measure the attack's progress using the *Hamming distance*, defined as the number of incorrect coefficients in the recovered by LDPC secret key compared to the correct one. As shown in the Figure, the average Hamming distance steadily decreases as more information is gathered through successive batches of oracle queries. A key finding is that a perfect recovery (Hamming distance of 0) does not always happen. Often, the attack terminates with a small number of errors in the recovered key. This remaining uncertainty can be efficiently resolved with light post-processing. Specifically, we have implemented a meet-in-the-middle approach over two lists of size about 2^{20}. For the implementation written in C on a single thread, we can recover the key for a Hamming distance of 4 or less in less than a second.

3.5 Investigating Extremely High-Noise Oracle

For completeness, we estimate the performance of our approach when the oracle is extremely noisy. To reuse the analysis and experiments already presented in this paper, we construct a higher-accuracy oracle by repeating independent calls to the original (noisier) oracle and combining the outcomes. In other words, we reduce the noisy-oracle setting to the higher-accuracy setting treated elsewhere in the main body.

Concretely, let the basic (Bernoulli) oracle have accuracy ρ. After n repeated calls we observe $\mathbf{y}$ containing k zeros. The

Algorithm 2 Hybrid Adaptive Key-Recovery Strategy

Input: Oracle noise level ρ, Maximum number of batches B_{max}, Entropy threshold B_{entr}
Output: Estimated secret key $\hat{\mathbf{s}}$

$C_{FR} \leftarrow$ Precomputed 7-bit full rotation encoding
$D_{FR} \leftarrow \emptyset$ ▷ *Storage for full rotation oracle responses*
$D_{batch} \leftarrow \emptyset$ ▷ *Storage for batch oracle responses*
for all polynomials $\mathbf{s}_k$ in $\mathbf{s}$ **do**
  for all $j \in \{0,\ldots,63\}$ **do**
    $I \leftarrow \{j, j+64, j+128, j+192\}$ ▷ *Select coefficient indices*
    $\mathbf{y} \leftarrow$ empty array of size 7
    for $i \in \{0,\ldots,6\}$ **do**
      $\mathsf{ct}_i \leftarrow \text{BuildFRCt}(C_{FR}, I, i, k)$ ▷ *Build ciphertext for i-th bit for* $\mathbf{s_k}$
      $\mathbf{y}[k] \leftarrow \text{QueryOracle}(\mathsf{ct}_i)$
    end for
    Add $(I, C_{FR}, \mathbf{y})$ to D_{FR}
  end for
end for
$\mathsf{pmfs} \leftarrow \text{ComputeInitialPMFs}(D_{FR})$ ▷ *Initial belief via Max-Likelihood*
for $b \in \{1,\ldots,B_{max}\}$ **do**
  $P \leftarrow \text{RandomPartition}(\{0,\ldots,n\cdot d-1\}, \text{size}=\omega)$
  for all subset of indices $I \in P$ **do**
    if $H_2(\mathsf{pmfs}[I]) < B_{entr}$ **then continue**; ▷ *Skip if the chosen coefficients have low total entropy*
    $\mathsf{enc} \leftarrow \text{FindBalancedEncoding}(I, \mathsf{pmfs})$ ▷ *Adaptively find best encoding*
    $\mathsf{ct} \leftarrow \text{BuildBatchCt}(\mathsf{enc}, I)$
    $y \leftarrow \text{QueryOracle}(\mathsf{ct})$ ▷ *Get 1 noisy bit*
    Add (I, c, y) to D_{batch}
  end for
  $\mathsf{pmfs} \leftarrow \text{LDPC-Decode}(D_{FR} \cup D_{batch})$ ▷ *Update beliefs*
end for
$\hat{\mathbf{s}} \leftarrow \text{EstimateKeyFromPMFs}(\mathsf{pmfs})$ ▷ *Make final decision*
return $\hat{\mathbf{s}}$

likelihood of these observations assuming the true value is 0 is

$$\Pr(\mathbf{y} \mid 0) = \rho^k (1-\rho)^{n-k},$$

while assuming the true value is 1 it is

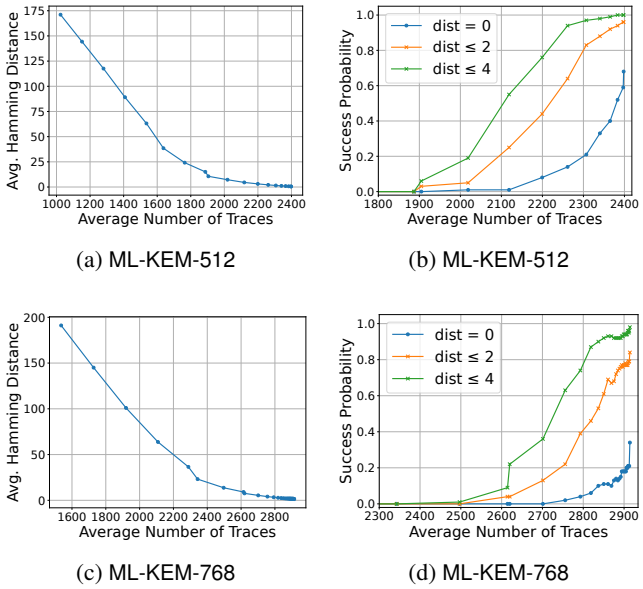
$$\Pr(\mathbf{y} \mid 1) = (1-\rho)^k \rho^{n-k}.$$

The likelihood ratio (and hence the posterior odds, up to the prior) is therefore

$$\frac{\Pr(\mathbf{y} \mid 0)}{\Pr(\mathbf{y} \mid 1)} = \left(\frac{\rho}{1-\rho}\right)^{2k-n}.$$

We work in the log domain to avoid numerical underflow. We repeat oracle calls until the posterior probability for one hypothesis exceeds a prescribed threshold; equivalently, we stop when the log-likelihood ratio crosses the threshold

$$\log \frac{\rho'}{1-\rho'},$$



(a) ML-KEM-512 (b) ML-KEM-512

(c) ML-KEM-768 (d) ML-KEM-768

Figure 4: Attack results for ML-KEM-512 (top row) and ML-KEM-768 (bottom row), averaged over 100 keys. The *left column* shows the decrease in the number of incorrect secret coefficients as more oracle calls are made. The *right column* shows the final key-recovery success rate with no post-processing, light, and medium post-processing. Each data point is obtained after running the full LDPC decoder on the accumulated information from each batch of oracle queries.

which corresponds to a target effective accuracy ρ'. This procedure is essentially majority voting with soft information; the resulting posterior is passed into the decoder as soft information.

For ML-KEM-768, we take the noisy oracle accuracy $\rho = 0.515$, reported for the masked-hardware scenario in [41]. Lifting the oracle to $\rho' = 0.95$, we can achieve on average 4 291 390 traces which has a ratio of 1.79 to the theoretical lower bound.

After we submitted this paper in August 2025 we became aware of a related paper in the ePrint by Hermelink, Mårtensson, and Tran [19], which also touches upon very noisy oracles and reports results for the noisy binary PC oracle. They report a mean of 19 319 143 traces for ML-KEM-768 for $\rho = 0.51$, which has a ratio of 3.57 to the lower bound. Our approach with lifting to $\rho' = 0.95$ achieves on average 9 690 509 traces with a ratio of 1.79, demonstrating the practical benefit of our attack.

We deliberately fixed $\rho' = 0.95$ because the parameters and encodings were already optimized for that target. We believe, however, that further gains are possible: one could target a smaller accuracy $\rho' < 0.95$ and optimize all the parameters for it. We leave the analysis of the optimal choice of ρ' as an open problem.

4 Application to GoFetch Attack on Kyber

4.1 Adversary Model and Attack Phases

The GoFetch attack [3] is a microarchitectural side-channel attack capable of breaking constant-time cryptographic implementations. It exploits a hardware feature known as the Data Memory-Dependent Prefetcher (DMP), which is present in Apple M-series and Intel's 13th generation CPUs.

Unlike traditional prefetchers that predict memory access patterns based on address history, the DMP inspects the content of data being loaded into the cache. If any data value resembles a memory pointer, the DMP attempts to "dereference" it by prefetching data from the supposed memory address. This behavior creates a powerful side channel: an attacker can craft a cryptographic input such that an intermediate value, dependent on a secret key, appears as a valid pointer if and only if the secret holds a specific value. By monitoring cache activity to detect whether the DMP prefetches from the target of this "pointer", the attacker can learn information about the secret key.

In our attack, this mechanism is used to construct a decryption failure (DF) oracle. The attacker crafts a ciphertext that, upon decryption, produces a message containing either a target pointer or a slightly modified version with one flipped bit. The secret-dependent bit flip is designed such that the DMP is triggered on the target pointer in one case but not the other. By observing the DMP's activity, the attacker can distinguish between these two outcomes, effectively creating a DF oracle to extract the secret key. Our method employs a ciphertext construction that allows targeting all secret key coefficients. This enables a full key recovery without the need for the computationally intensive lattice reduction step required by the original GoFetch approach and leverages a more robust statistical model for interpreting noisy measurements.

Adversary Model. The adversary executes unprivileged code on the same machine as the victim, but the attacker and victim processes operate in separate address spaces and do not share memory. This assumption, consistent with the GoFetch paper, precludes the use of common Flush+Reload cache attacks. Instead, the attack leverages a separate thread to perform high-resolution timing measurements for a Prime+Probe [27] attack. Following the GoFetch methodology, our attack is divided into two primary stages: a **calibration phase** and a **key-recovery phase**. While we assume the availability of high-resolution timers typical of native execution environments, we consider the further restriction of browser-based sandboxes—where such timers may be jittered or unavailable—to be outside our current scope and a subject for future work.

Phase 1: Calibration. The calibration phase prepares the environment for the attack and profiles the target system's timing characteristics. This phase consists of three steps. First, an **Eviction Set Search** is performed to find groups of memory addresses that map to the same cache set, which is necessary for the Prime+Probe technique. This initial step does not require interaction with the victim process. Second, the attacker builds **Compound Eviction Sets**, as introduced by GoFetch. These are pairs of L2 eviction sets chosen so that one evicts the victim buffer where the crafted pointer may reside during computation, while the other targets the cache set of the pointer's eventual dereference. Finding these sets is essential for isolating the DMP's activity and requires sending a large volume of ciphertexts to the victim. These two steps are essentially unchanged from GoFetch, as it is the necessary precondition to exploit the DMP. The final step is **Profiling**, where we model the distributions of cache hits and misses as Gaussian. We collect 1 000 timing measurements to establish parameters (μ_0, σ_0) for cache hits (Class 0) and (μ_1, σ_1) for cache misses (Class 1, DMP activation), as illustrated in Figure 5. This profiling step may be optional on similar hardware, but we include it explicitly as our results differ from those in [3, Figure 9], likely due to hardware differences between our MacBook Air (Apple M1) and their Mac Mini (Apple M1), both running macOS 13.5.

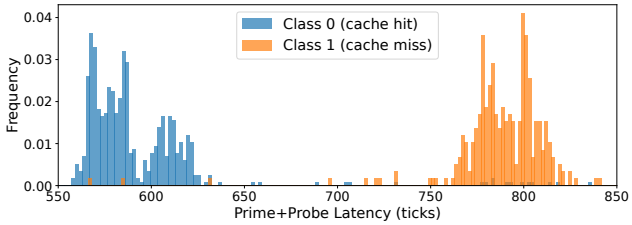


Figure 5: Timing difference between cache hit and cache miss during the attack (1 000 measurements)

The rationale for this profiling is to frame the oracle's response as a hypothesis testing problem. During the attack, for a new set of k timing measurements $\mathbf{t} = \{t_1, \ldots, t_k\}$, we evaluate two competing hypotheses: Hypothesis 0 (H_0), that $\mathbf{t}$ was sampled from the cache-hit distribution $\mathcal{N}(\mu_0, \sigma_0^2)$, and Hypothesis 1 ($H_1$), that it was sampled from the cache-miss distribution $\mathcal{N}(\mu_1, \sigma_1^2)$. Our goal is to determine the posterior probability of each hypothesis, for which we calculate $P(H_0|\mathbf{t})$ using Bayes' theorem with balanced priors

$$\Pr(H_0|\mathbf{t}) = \frac{\Pr(\mathbf{t}|H_0)\Pr(H_0)}{\Pr(\mathbf{t})}, \quad \Pr(H_0) = \Pr(H_1) = 0.5.$$

During the attack, we filter outliers (e.g., timings greater than 1 500 ticks) and accumulate measurements until either $\Pr(H_0|\mathbf{t})$ or $\Pr(H_1|\mathbf{t})$ exceeds a confidence threshold of 0.85. This method provides a more stable oracle despite measurement noise. While the posterior probability provides valuable soft information, we surprisingly found that key recovery was more effective when treating the oracle as a binary output

(i.e., outputting 0 if $\Pr(H_0|\mathbf{t}) \geq 0.5$). This observation may warrant further investigation.

Phase 2: Key Recovery. The key-recovery phase, which follows the strategy detailed in Section 3.4, is orchestrated by three cooperating processes. The first process implements the logic of the attack, deciding which ciphertext to test next based on the oracle's previous responses. The second process is responsible for interacting with the victim; it takes a ciphertext from the first process, sends it to the victim, executing the Prime+Probe measurement loop. This process uses a dedicated thread to create a high-resolution timer for its measurements and reports back the final conditional probability after several measurements. The victim process in a loop sends its public key, receives a ciphertext, then runs a decapsulation. While the first two processes could be merged into a single one, we separated the main logic into a distinct Python process for convenience during implementation.

4.2 Executing the Attack

To instantiate the decryption failure oracle, we craft ciphertexts that cause the decrypted message to contain a specific pointer, ptr, which is learned during the calibration phase. The core of the technique is to create a predicate, dependent on the secret key, that determines whether the decrypted message contains the exact ptr or a slightly modified version, ptr'. By observing whether the DMP dereferences ptr, we can learn the outcome of the predicate. As established in [3], the DMP is insensitive to bit flips in the upper 8 and lower 7 bits of a 64-bit pointer, a property that was a limitation but turns out as a helpful tool in our case.

Ciphertext for Basic Encodings. To create the encodings described in Section 3.2, we first define $\mathsf{ptr}' = \mathsf{ptr} \oplus (1 \ll i)$, where the bit at position i is within the DMP's sensitive region ($7 < i < 56$) and $\mathsf{ptr}[i] = 1$[2]. We then construct the ciphertext component $\mathbf{v}'$ such that the decrypted message is an array of pointers, starting with ptr' followed by three random pointers that reside within the same 4 GB memory region. The DMP will prefetch ptr if and only if the decryption result is modified back from ptr' to ptr, which happens when the chosen secret-dependent predicate is satisfied.

Crucially, for our attack, the value of $\mathbf{v}$ remains fixed across all oracle queries. To target different coefficients of the secret key $\mathbf{s}$, we modify the nonzero positions within the ciphertext component $\mathbf{u}$. This approach allows us to probe any coefficient of the secret key, thereby overcoming the DMP sensitivity limitations that constrained the original GoFetch attack and enabling a full key recovery without subsequent lattice reduction.

[2]There is an off-by-one error comparing an inequality over $\mathsf{ptr}'[i] = 1$ to $\mathsf{ptr}'[i] = 0$, thus, we fix $\mathsf{ptr}'[i] = 0$ to follow Section 3.

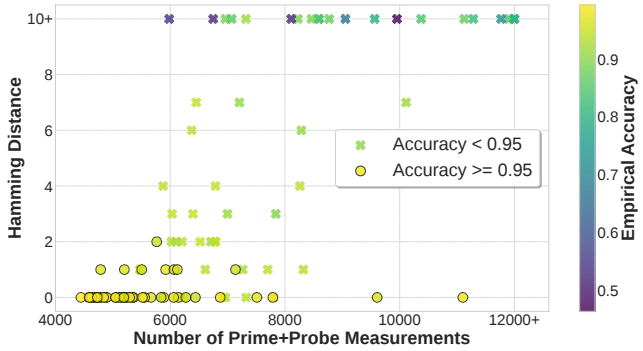


Figure 6: Distribution of final Hamming distance versus the total number of Prime+Probe measurements for 100 ML-KEM-512 key-recovery attempts. Each point's color indicates the empirical oracle accuracy for that run. Attack parameters were based on $\rho = 0.95$; thus, circles nicely follow the theory

Ciphertext for Full Rotation Encodings. Constructing the full rotation encodings from Section 3.3 is less straightforward, as the goal is to detect if the decrypted message contains ptr at *any* of four positions. A naive idea is to construct $\mathbf{v}'$ to correspond to a message of four identical pointers, $[\mathsf{ptr}', \mathsf{ptr}', \mathsf{ptr}', \mathsf{ptr}']$. The rationale is that if the secret causes any of these to become ptr, the DMP will prefetch it, and probing will result in a cache miss. Unexpectedly, this approach does not work. Our experiments show that the oracle is only sensitive to the first two pointers; for example, if the decrypted message is $[\mathsf{ptr}', \mathsf{ptr}', \mathsf{ptr}, \mathsf{ptr}']$, no prefetch occurs. This implies that the DMP may stop scanning an array of pointers after encountering the same repeating value at the beginning. This behavior, which reveals additional logic within the DMP, might be explained by a mechanism like the history filter mentioned in [3], designed to prevent redundant operations on repetitive data.

We bypass this limitation by constructing $\mathbf{v}'$ such that the intended decrypted message is an array of slightly different pointers: $[\mathsf{ptr}' \oplus 1, \mathsf{ptr}' \oplus 2, \mathsf{ptr}' \oplus 4, \mathsf{ptr}' \oplus 8]$. The DMP ignores the lower 7 bits of a pointer, so these values all target the same L2 cache line. However, despite the prefetch targeting the same cache line, the pointers are technically different, which avoids the problem of the DMP halting on repetitive values. This allows us to create a reliable oracle that triggers if any of the four secret-dependent predicates are satisfied. We also note an unexplained empirical observation: this technique only works reliably when the flipped bit i in the construction of ptr' satisfies $i \geq 14$. We leave a full investigation of this behavior to future work.

Results of the Attack. We ran the attack on 100 random secret keys to assess its performance. The initial profiling was performed only once, and the resulting Gaussian parameters were used for all subsequent tests. The results for ML-KEM-

Table 3: Aggregated results for 73 successful ML-KEM-512 key recoveries

Measurement repetitions	Empirical accuracy	Traces	Hamming distance
2.69	0.9614	6047.84	0.73

512 are shown in Figure 6, where the most accurate oracle cases are centered in the lower-left corner. Choosing a different confidence threshold can increase the chance of having a more accurate oracle. A full key recovery takes approximately 25 minutes, with nearly half of this time dedicated to the calibration phase. As expected with a timing-based side channel, the results show considerable variance, with some runs exhibiting significant noise. In 73 of the 100 trials, the recovered secret key had a Hamming distance of 4 or less from the true key, which we classify as a success. The aggregated statistics for these successful runs are presented in Table 3. The number of traces reported includes only those made during the key-recovery phase.

As seen in Figure 6, the empirical accuracy of the oracle exhibits significant variability. Our experiments reveal that the attack occasionally suffers from bursts of consecutive incorrect oracle responses. This phenomenon is likely attributed to system-level noise, such as operating system interrupts, context switches, or background processes. These events can inadvertently evict cache lines during the Prime+Probe measurement window, causing false cache misses that the attacker misinterprets as DMP activity. Despite these occasional bursts of noise, we find that increasing measurement redundancy to guarantee a success rate close to 1 is suboptimal. In our current setup, it is cheaper (in terms of number of traces) to accept a non-zero failure rate and simply rerun the key recovery phase when necessary.

4.3 Comparison with the GoFetch Attack

Our proposed attack demonstrates a significant improvement in efficiency and capability when compared to the GoFetch attack presented in [3]. The advantages are evident in several key areas.

Most critically, our method drastically reduces the number of traces required for key recovery. This is due to a fundamental difference in how the two attacks handle measurement noise. The GoFetch attack on Kyber-512 relies on repetition to create a *highly accurate* oracle; it uses 32 traces for each ciphertext query, effectively performing a majority vote to overcome noise. This approach leads to a total trace count of 100 352:

$$\begin{array}{rl} & 392 \text{ coefficients} \\ \times & 8 \text{ ciphertexts/coefficient} \\ \times & \mathbf{32} \text{ traces/ciphertext} \\ \hline = & 100\,352 \text{ traces} \end{array}$$

This extensive data collection is coupled with a computationally intensive post-processing step, which the authors report requires an average of 7.5 hours on a high-performance server (an Intel Xeon Platinum 8352Y with 128 logical cores and 1.48 TB of RAM). In stark contrast, our methodology is designed to work directly with noisy oracle responses. By integrating powerful error-correction techniques, we eliminate the need for extensive repetition, requiring only a small number of traces per query. This results in a tenfold reduction in the total online trace acquisition. Although our experimental setup exhibits a slightly lower noise floor (as seen by comparing Figure 5 with [3, Figure 9]), this dramatic improvement is overwhelmingly attributable to our novel attack methodology, not the hardware platform.

Finally, our attack overcomes a major limitation of the GoFetch methodology that prevents it from scaling to higher-security schemes. The GoFetch attack is fundamentally constrained by the physical behavior of the Data Memory-dependent Prefetcher (DMP). As detailed in their analysis, the DMP ignores the leading 8 and trailing 7 bits of any 64-bit value it interprets as a pointer. This means that for every block of 64 secret coefficients, a portion of the secret remains unrecoverable through their leakage mechanism. For ML-KEM-512, this results in 392 of 512 recoverable coefficients, leaving 120 unknown. While this smaller number of unknowns can be solved with a computationally intensive lattice reduction, the problem becomes intractable for ML-KEM-768 and ML-KEM-1024. For these larger parameter sets (768 and 1024 coefficients, respectively), the number of unrecoverable coefficients increases proportionally, making the search space for the final lattice reduction step too large to be practical. Our method does not suffer from this constraint and can recover all secret coefficients, enabling a full key recovery even for these schemes with a higher security level.

5 Discussions

We now discuss the key properties of our attack methodology that highlight its practical significance, as well as the implications for potential countermeasures.

5.1 Attack Properties

Near-Optimal Efficiency. Our attack is "near-optimal" in its query efficiency, approaching the theoretical Shannon lower bound for key recovery. By achieving a query-to-bound ratio as low as 1.35, our method demonstrates high efficiency with little room for further improvement. This result demonstrates that the information-theoretic bound is, in this case, a tight bound, which allows for an accurate evaluation of the security margins of cryptographic systems.

Extension to Multi-Value PC Oracles. Our attack's core methodology could be extended beyond one-bit-bound ora-

cles to more powerful multi-value PC variants. Finding optimally balanced queries in this larger information space is a nontrivial optimization problem that we leave for future work.

Applicability for ML-KEM-1024. While our presentation focuses on ML-KEM-512 and ML-KEM-768, the attack framework extends directly to ML-KEM-1024. We prioritized the lower-security variants as they are more commonly targeted in side-channel analysis, particularly in the context of resource-constrained and embedded devices. The attack on ML-KEM-1024 would follow the same methodology as for ML-KEM-768. Although the secret key space for ML-KEM-1024 is larger by a factor of $4/3$, we anticipate that the required number of traces would increase by a smaller margin. This expected gain in efficiency stems from ML-KEM-1024's larger compression parameters (d_u, d_v). The reduced compression allows for more fine-grained control when constructing the inequalities that form our oracle queries, enabling the creation of more balanced encodings. As each query would yield more information on average, the total number of traces needed for a full key recovery would be proportionally lower.

5.2 Countermeasures

Implications for One-Bit-Bound Oracle Attacks. Our work improves the mathematical exploitation of one-bit-bound oracles. Consequently, traditional software hardening approaches like masking and blinding, which aim to increase the noise and reduce the signal of side-channel leakage, remain conceptually valid. However, their practical security must be re-evaluated. By demonstrating a key recovery that requires significantly fewer traces, we show that an amount of noise previously considered sufficient to deter an attacker may no longer be adequate. Implementations may require more robust hardening (e.g., higher-order masking) to provide the same level of security, which in turn impacts performance. The trade-off between security and efficiency for these countermeasures must be reconsidered in light of our more potent attack methodology. For instance, first-order masked software has been shown to yield high oracle accuracy (e.g., $\rho \approx 0.96$ [41]), and our analysis in Section 3.5 indicates that even highly noisy hardware-masked scenarios ($\rho \approx 0.515$) are potentially vulnerable.

Implications for the GoFetch Scenario. In the specific context of the GoFetch attack, our method improves the efficiency of exploiting the DF oracle, but it does not alter the fundamental leakage source: the Data Memory-Dependent Prefetcher (DMP). Therefore, the countermeasures proposed in the original GoFetch paper remain fully effective. Disabling the DMP via an ISA extension or restricting cryptographic computations to cores where the DMP is not active would completely mitigate this specific attack vector by eliminating the oracle at its source. Our work simply makes the leakage that is available far more dangerous.

6 Conclusion and Future Works

In this paper, we have proposed a new methodology for decryption failure (DF) oracle attacks on ML-KEM that challenges their perceived inefficiency. Our approach integrates adaptive query generation with the error-correction power of LDPC codes, introducing a hybrid framework. By constructing queries that yield balanced, high-entropy parity checks over multiple secret coefficients, we maximize the information gained from each oracle call. This technique significantly reduces the number of traces required for key recovery, demonstrating that a carefully constructed DF oracle attack can be more potent than its binary PC counterparts, especially in noisy environments. We validate the real-world applicability of our attack by applying it to the GoFetch side-channel vulnerability, achieving a substantial reduction in the required number of traces for successful key recovery.

We selected ML-KEM as our primary target due to its standardization as FIPS 203 and its anticipated long-term importance among lattice-based KEMs. This methodology can be broadened to encompass other lattice-based proposals, potentially guiding the selection of PQC standards beyond those established by NIST. Future work should include extending our current approach to address multi-value plaintext checking oracle attacks [31, 40] on ML-KEM, aiming for performance optimization that approaches information-theoretic lower bounds—particularly in protected platforms with less accurate oracles. Further research should focus on identifying additional leakage sources from diverse physical platforms to instantiate the required DF oracle and apply our new attack framework; high-order masked platforms are particularly interesting cases. Additionally, the broader implications of our new DMP reverse-engineering results warrant further exploration. The halting scan behavior we identified, along with its bypass technique, could be leveraged to enhance attacks on other cryptographic implementations. Finally, given that more powerful attacks can render even protected implementations vulnerable with fewer traces needed for key recovery, designing more efficient countermeasures offering higher-order protection at a reduced performance cost is an important area of investigation. While our current adversary model focuses on an unprivileged user on a multi-user system with access to high-resolution timers, exploring this attack within a browser sandbox remains a compelling direction for future work.

Acknowledgement

This work was supported by the Swedish Research Council (grant numbers 2021-04602 and 2023-03654), the Swedish Civil Contingencies Agency (grant number 2020-11632), the

Crafoord Foundation, and the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation. The computations and simulations were partly enabled by resources provided by LUNARC.

Ethical Considerations

Our primary contribution is a general mathematical framework for dramatically improving oracle-based side-channel attacks, validating this framework through a specific instantiation against the GoFetch vulnerability on Apple hardware. In conducting this work, we navigated the ethical challenge of balancing the public benefit of identifying this heightened risk against the potential harm of publishing a more potent attack methodology. Below, we discuss the impact on stakeholders, our responsible disclosure process, and the broader implications for the security community.

Stakeholder Analysis and Impact. We considered the potential harm to several parties. First, for hardware vendors (in this case, Apple), our attack significantly reduces the resources required to exploit the Data Memory-Dependent Prefetcher (DMP) vulnerability, making the threat more immediate and accessible than previously understood. Specifically, larger security parameters are no longer secure. Second, for cryptographic designers (such as the NIST and CRYSTALS/ML-KEM teams), our findings demonstrate that any side-channel leakage instantiating a 1-bit oracle is far more dangerous than previously believed, capable of being exploited with near-optimal efficiency. Finally, users of affected systems face a higher risk of key theft.

However, we believe the long-term benefits to these stakeholders outweigh the risks. By demonstrating the severity of these improved attacks, we motivate the development and adoption of robust mitigations that will ultimately protect end-users and strengthen cryptographic libraries against future oracle-based threats. As a temporary measure, disabling DMP prevents this particular attack.

Responsible Disclosure. Given the increased severity of our new technique compared to the original GoFetch disclosure, we adhered to a strict responsible disclosure process.

We notified Apple and the CRYSTALS (Kyber/ML-KEM) team of our findings, providing them with the details of our improved attack efficiency. This ensures that software and hardware vendors have the necessary information to assess the urgency of deploying mitigations.

Our practical validation was exclusively performed on Apple silicon. While similar DMP mechanisms exist in other architectures (e.g., Intel), we focused our disclosure on the validated target. As the specifics of our attack rely on the M-series implementation and Intel is already aware of general DMP attack techniques, we did not extend the disclosure to Intel, focusing instead on the verified vulnerability.

Broader Implications and Dual Use. We acknowledge that our primary contribution has dual-use potential. While it empowers security researchers to identify and test vulnerabilities with greater efficiency, it could theoretically be used by adversaries to optimize attacks on other leakage sources (power, EM, or timing). However, we contend that suppressing this methodology would be more harmful to the security community. By publishing the mathematical underpinnings of these attacks, we enable defenders to accurately model the risk of "minor" leakages and design countermeasures that render 1-bit oracles imperfect or impossible to exploit in practical settings.

Open Science

To facilitate the verification of our results and to encourage further research, we make our simulation code, attack implementation, and the data used to generate the figures in this paper publicly available in an open repository https://doi.org/10.5281/zenodo.17899501.

References

[1] Ciprian Băetu, F. Betül Durak, Loïs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay. Misuse attacks on post-quantum cryptosystems. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 747–776. Springer, Cham, May 2019.

[2] Daniel J. Bernstein, Karthikeyan Bhargavan, Shivam Bhasin, Anupam Chattopadhyay, Tee Kiah Chia, Matthias J. Kannwischer, Franziskus Kiefer, Thales B. Paiva, Prasanna Ravi, and Goutam Tamvada. Kyberslash: Exploiting secret-dependent division timings in kyber implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(2):209–234, 2025.

[3] Boru Chen, Yingchen Wang, Pradyumna Shome, Christopher W. Fletcher, David Kohlbrenner, Riccardo Paccagnella, and Daniel Genkin. Gofetch: Breaking constant-time cryptographic implementations using data memory-dependent prefetchers. In *USENIX Security*, 2024.

[4] CVE-2024-37880. The Kyber reference implementation before 9b8d306, when compiled by LLVM Clang through 18.x with some common optimization options, has a timing side channel that allows attackers to recover an ML-KEM 512 secret key in minutes. https://www.cvedetails.com/cve/CVE-2024-37880/, 2024. Accessed: 2025-07-11.

[5] Jan-Pieter D'Anvers, Marcel Tiepelt, Frederik Vercauteren, and Ingrid Verbauwhede. Timing attacks on error correcting codes in post-quantum schemes. Cryptology ePrint Archive, Report 2019/292, 2019.

[6] Jintai Ding, Scott R. Fluhrer, and Saraswathy RV. Complete attack on RLWE key exchange with reused keys, without signal leakage. In Willy Susilo and Guomin Yang, editors, *ACISP 18*, volume 10946 of *LNCS*, pages 467–486. Springer, Cham, July 2018.

[7] Haiyue Dong and Qian Guo. OT-PCA: New key-recovery plaintext-checking oracle based side-channel attacks on HQC with offline templates. In *IACR TCHES*, 2025.

[8] Edward Eaton, Matthieu Lequesne, Alex Parent, and Nicolas Sendrier. QC-MDPC: A timing attack and a CCA2 KEM. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 47–76. Springer, Cham, 2018.

[9] Scott Fluhrer. Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085, 2016.

[10] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Berlin, Heidelberg, August 1999.

[11] Stefan Gast, Hannes Weissteiner, Robin Leander Schröder, and Daniel Gruss. CounterSEVeillance: Performance-counter attacks on AMD SEV-SNP. In *Network and Distributed System Security Symposium 2025: NDSS 2025*, 2025.

[12] Qian Guo, Clemens Hlauschek, Thomas Johansson, Norman Lahr, Alexander Nilsson, and Robin Leander Schröder. Don't reject this: Key-recovery timing attacks due to rejection-sampling in HQC and BIKE. *IACR TCHES*, 2022(3):223–263, 2022.

[13] Qian Guo, Thomas Johansson, and Alexander Nilsson. A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 359–386. Springer, Cham, August 2020.

[14] Qian Guo and Erik Mårtensson. Do not bound to a single position: Near-optimal multi-positional mismatch attacks against kyber and saber. In Thomas Johansson and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023*, pages 291–320. Springer, Cham, August 2023.

[15] Qian Guo, Erik Mårtensson, and Adrian Åström. The perils of limited key reuse: Adaptive and parallel mismatch attacks with post-processing against kyber. *CiC*, 1(3):21, 2024.

[16] Qian Guo, Denis Nabokov, Alexander Nilsson, and Thomas Johansson. SCA-LDPC: A code-based framework for key-recovery side-channel attacks on post-quantum encryption schemes. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part IV*, volume 14441 of *LNCS*, pages 203–236. Springer, Singapore, December 2023.

[17] Chris Hall, Ian Goldberg, and Bruce Schneier. Reaction attacks against several public-key cryptosystems. In Vijay Varadharajan and Yi Mu, editors, *ICICS 99*, volume 1726 of *LNCS*, pages 2–12. Springer, Berlin, Heidelberg, November 1999.

[18] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. Belief propagation meets lattice reduction: Security estimates for error-tolerant key recovery from decryption errors. *IACR TCHES*, 2023(4):287–317, 2023.

[19] Julius Hermelink, Erik Mårtensson, and Maggie Tran. Noise-tolerant plaintext-checking oracle attacks–a soft-analytic approach applied to ml-kem. *Cryptology ePrint Archive*, 2025.

[20] Julius Hermelink, Kai-Chun Ning, Richard Petri, and Emanuele Strieder. The insecurity of masked comparisons: SCAs on ML-KEM's FO-transform. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 2430–2444. ACM Press, October 2024.

[21] Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. Fault-enabled chosen-ciphertext attacks on kyber. In *International Conference on Cryptology in India*, pages 311–334. Springer, 2021.

[22] Senyang Huang, Rui Qi Sim, Chitchanok Chuengsatiansup, Qian Guo, and Thomas Johansson. Cache-timing attack against HQC. *IACR TCHES*, 2023(3):136–163, 2023.

[23] Jinnuo Li, Chi Cheng, Muyan Shen, Peng Chen, Qian Guo, Dongsheng Liu, Liji Wu, and Jian Weng. Grafted trees bear better fruit: An improved multiple-valued plaintext-checking side-channel attack against kyber. In *Design, Automation & Test in Europe Conference, DATE 2025, Lyon, France, March 31 - April 2, 2025*, pages 1–7. IEEE, 2025.

[24] Puja Mondal, Suparna Kundu, Sarani Bhattacharya, Angshuman Karmakar, and Ingrid Verbauwhede. A practical key-recovery attack on LWE-based key-encapsulation mechanism schemes using rowhammer. In Christina Pöpper and Lejla Batina, editors, *ACNS 2024, Part III*, volume 14585 of *LNCS*, pages 271–300. Springer, Cham, March 2024.

[25] National Institute of Standards and Technology. Module-Lattice-Based Key-Encapsulation Mechanism Standard. Technical Report FIPS PUB 203, U.S. Department of Commerce, August 2024.

[26] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A side-channel attack on a masked IND-CCA secure Saber KEM implementation. *IACR TCHES*, 2021(4):676–707, 2021.

[27] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: the case of aes. In *Cryptographers' track at the RSA conference*, pages 1–20. Springer, 2006.

[28] Thales Bandiera Paiva and Routo Terada. A timing attack on the HQC encryption scheme. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 551–573. Springer, Cham, August 2019.

[29] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 513–533. Springer, Cham, September 2017.

[30] Yue Qin, Chi Cheng, Xiaohan Zhang, Yanbin Pan, Lei Hu, and Jintai Ding. A systematic approach and analysis of key mismatch attacks on lattice-based NIST candidate KEMs. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 92–121. Springer, Cham, December 2021.

[31] Gokulnath Rajendran, Prasanna Ravi, Jan-Pieter D'Anvers, Shivam Bhasin, and Anupam Chattopadhyay. Pushing the limits of generic side-channel attacks on LWE-based KEMs - parallel PC oracle attacks on Kyber KEM and beyond. *IACR TCHES*, 2023(2):418–446, 2023.

[32] Prasanna Ravi, Anupam Chattopadhyay, Jan-Pieter D'Anvers, and Anubhab Baksi. Side-channel and fault-injection attacks over lattice-based post-quantum schemes (kyber, dilithium): Survey and new results. *ACM Trans. Embed. Comput. Syst.*, 23(2):35:1–35:54, 2024.

[33] Prasanna Ravi, Martianus Frederic Ezerman, Shivam Bhasin, Anupam Chattopadhyay, and Sujoy Sinha Roy. Will you cross the threshold for me? Generic side-channel assisted chosen-ciphertext attacks on NTRU-based KEMs. *IACR TCHES*, 2022(1):722–761, 2022.

[34] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs. *IACR TCHES*, 2020(3):307–335, 2020.

[35] Thomas Schamberger, Lukas Holzbaur, Julian Renner, Antonia Wachter-Zeh, and Georg Sigl. A power side-channel attack on the reed-muller reed-solomon version of the HQC cryptosystem. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022*, pages 327–352. Springer, Cham, September 2022.

[36] Robin Leander Schröder, Stefan Gast, and Qian Guo. Divide and surrender: Exploiting variable division instruction timing in HQC key recovery attacks. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024.

[37] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[38] Mingyao Shao, Yuejun Liu, and Yongbin Zhou. Pairwise and parallel: Enhancing the key mismatch attacks on kyber and beyond. In Jianying Zhou, Tony Q. S. Quek, Debin Gao, and Alvaro A. Cárdenas, editors, *ASIACCS 24*. ACM Press, July 2024.

[39] Muyan Shen, Chi Cheng, Xiaohan Zhang, Qian Guo, and Tao Jiang. Find the bad apples: An efficient method for perfect key recovery under imperfect SCA oracles - A case study of Kyber. *IACR TCHES*, 2023(1):89–112, 2023.

[40] Yutaro Tanaka, Rei Ueno, Keita Xagawa, Akira Ito, Junko Takahashi, and Naofumi Homma. Multiple-valued plaintext-checking side-channel attacks on post-quantum KEMs. *IACR TCHES*, 2023(3):473–503, 2023.

[41] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. Curse of re-encryption: A generic power/EM analysis on post-quantum KEMs. *IACR TCHES*, 2022(1):296–322, 2022.

[42] Yingchen Wang, Riccardo Paccagnella, Elizabeth Tang He, Hovav Shacham, Christopher W. Fletcher, and David Kohlbrenner. Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 679–697. USENIX Association, August 2022.

[43] Hannes Weissteiner, Fabian Rauscher, Robin Leander Schröder, Jonas Juffinger, Stefan Gast, Jan Wichelmann, Thomas Eisenbarth, Daniel Gruss, SIT Fraunhofer, and Vienna Fraunhofer Austria. Teecorrelate: An information-preserving defense against performance-counter attacks on tees. In *USENIX Security 2025*. 2025.

[44] Keita Xagawa, Akira Ito, Rei Ueno, Junko Takahashi, and Naofumi Homma. Fault-injection attacks against NIST's post-quantum cryptography round 3 KEM candidates. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 33–61. Springer, Cham, December 2021.

[45] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, and David Oswald. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of Kyber. Cryptology ePrint Archive, Report 2020/912, 2020.

A Extended Analysis of Encoding Strategies

To provide better intuition for the "Full Rotation" encoding chosen in Section 3.3, we analyze three different main strategies for constructing oracle queries. We utilize a simplified "toy model" to make the index arithmetic transparent.

Toy Model and Math Recap. We consider a polynomial ring with degree $n = 16$. We impose a computational constraint $\omega \leq 4$, meaning the set of secret key coefficients influencing a *single* oracle query must have cardinality at most 4.

Recall from Section 3.2 that the decryption failure oracle checks inequalities based on the value $(-\mathbf{s}^T\mathbf{u})[i]$. By selecting specific nonzero positions in the noise vector $\mathbf{v}$ (which correspond to check indices i), we determine which linear combinations are validated.

Strategy 1: Arbitrary disjoint encoding. The simplest approach is to target an arbitrary set of ω secret coefficients. We construct $\mathbf{u}$ with ω nonzero terms and set $\mathbf{v}$ to have exactly one nonzero coefficient at index i. We can target any subset of the key; however, in general, it is difficult to have a balanced encoding.

Strategy 2: The sliding window method. We choose $\mathbf{u}$ with nonzero coefficients separated by a fixed distance d (modulo n). We then check multiple positions in $\mathbf{v}$, also separated by distance d. This causes the linear combination to "slide" over the secret key, creating a chain of overlapping dependencies.

Example: Let $\mathbf{u} = z_0x^{15} + z_1x^{13}$ (distance 2). We check positions $i \in \{0,2,4\}$.

- $i = 0$: $(-\mathbf{s}^T\mathbf{u})[i] = s_1z_0 + s_3z_1$.
- $i = 2$: $(-\mathbf{s}^T\mathbf{u})[i] = s_3z_0 + s_5z_1$.
- $i = 4$: $(-\mathbf{s}^T\mathbf{u})[i] = s_5z_0 + s_7z_1$.

The total set of involved variables is $\{s_1, s_3, s_5, s_7\}$, which has size 4 (satisfying $\omega \leq 4$); see visual representation in Table 4.

Table 4: Sliding Window Dependencies ($n = 16, \omega = 4$). Shaded cells indicate involved coefficients.

	s_0	$\mathbf{s_1}$	s_2	$\mathbf{s_3}$	s_4	$\mathbf{s_5}$	s_6	$\mathbf{s_7}$	...
$i = 0$	·	z_0	·	z_1	·	·	·	·	...
$i = 2$	·	·	·	z_0	·	z_1	·	·	...
$i = 4$	·	·	·	·	·	z_0	·	z_1	...

Analysis: With sliding window we can use $1 < k < \omega$ inequalities, each over $\omega - k + 1$ variables. The flexibility is quite high, as the distance could be chosen arbitrarily. Despite the overall encoding depending on ω variables, each inequality has fewer than ω variables, which reduces the granularity and the probability of having a balanced encoding. Additionally, some variables participate in more inequalities than others, as seen in Table 4, s_1 is present only in the first inequality, while s_3 is present in two of them. This strategy is not used in our paper, but we believe it might be of use in a different scenario.

Strategy 3: Full rotation encoding. This approach, described in Section 3.3, constructs a "closed loop". Each out of ω inequalities has the same set of ω variables.

Example: Let $\mathbf{u} = z_0x^{15} + z_1x^{11} + z_2x^7 + z_3x^3$. We select non-zero positions $i \in \{1,5,9,13\}$ in $\mathbf{v}$. Carefully calculating $(-\mathbf{s}^T\mathbf{u})[i]$ leads to Table 5.

Table 5: Full Rotation Dependencies. Note that every check column is fully populated.

	$\mathbf{s_2}$	$\mathbf{s_6}$	$\mathbf{s_{10}}$	$\mathbf{s_{14}}$
$i = 1$	z_0	z_1	z_2	z_3
$i = 5$	$-z_3$	z_0	z_1	z_2
$i = 9$	$-z_2$	$-z_3$	z_0	z_1
$i = 13$	$-z_1$	$-z_2$	$-z_3$	z_0

Analysis: The full rotation encoding allows us to consider ω inequalities over ω variables, which favorably compares to the sliding window approach; thus, each encoding is more balanced, and the key recovery is more efficient. The downside is the lack of flexibility in the choice of secret variables. In our approach, we resolve this in the attack's second phase by adding random "linking" queries.

Constraints and Extensions. Two important implementation details distinguish these methods:

1. **Divisibility:** The full rotation approach relies on symmetric spacing, requiring ω to be a divisor of n. Since $n = 256$, ω must effectively be a power of 2. Strategies 1 and 2 do not have this restriction and can support any integer ω.

2. **Cross-Polynomial Linking:** In ML-KEM, the secret key is a vector of polynomials. The full rotation method is rigid and operates within a single polynomial ring. However, strategies 1 and 2 can be easily adapted to construct $\mathbf{u}$ with nonzero coefficients in different polynomial slots. We use the first strategy in the batch phase to create links between the disjoint sets of variables recovered by the full rotation phase.