How Strong is the FO-Calypse, Really? Instantiating Plaintext-Checking Oracles against Masked Software Implementations of ML-KEM

Brieuc Balon[1], Gaëtan Cassiers[1,2], Thibaud Schoenauen[1] and François-Xavier Standaert[1]

[1] Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium
[2] CryptoExperts, Paris, France

Abstract. Side-channel attacks exploiting Plaintext-Checking Oracles (PCOs) instantiated thanks to the leakage of the re-encryption step taking place during decapsulation are a well-known weakness of ML-KEM. An already wide literature investigated how to efficiently exploit such oracles, leading to easy (full) key recoveries. Somewhat surprisingly, the investigation of how to best instantiate PCOs against ML-KEM's most leaking operations is less investigated, in particular when it comes to quantitative evaluations against concrete masked implementations. In this paper, we first remedy this lack by systematically instantiating PCOs against three open source masked software implementations of the Keccak function used in ML-KEM, based on different masking techniques and programming styles. We evaluate the accuracy of PCOs for increasing number of shares using state-of-the-art profiled attacks against ARM Cortex-M4 implementations, and succeed obtaining high accuracy for up to 7 shares by leveraging the leakage of approximately 50 ML-KEM executions only. Doing so, we confirm the "computing more implies leaking more" adage and conclude that enforcing high security levels on such platforms will not be affordable. Next, we consolidate recent solutions for exploiting PCOs. For this purpose, we start by introducing a simple, concrete and re-usable model for PCOs targeting masked implementations of Keccak. We follow by clarifying that approaches based on hard decisions are suboptimal compared to soft (probabilistic) ones. We finally open a study of how to best exploit the adversary's computational power in a security evaluation. We show that (even naive) lattice based attacks are a promising approach for this purpose, leaving the design of a generic estimator that could efficiently leverage physical (side-channel) information as an interesting research direction.

Keywords: ML-KEM · side-channel analysis · plaintext checking oracles · masking

1 Introduction

Ever since its publication in 2018, CRYSTALS-Kyber [BDK+18], now standardized as ML-KEM [Nat24], has led to significant efforts towards understanding its implementation security against side-channel attacks. One line of research focused on single-trace attacks, which may lead to directly compromise ML-KEM's long-term key [PPM17, PP19, KPP20].[1] Besides, it has rapidly been observed that the structure of ML-KEM, and its reliance on the Fujisaki-Okamoto (FO) transform to ensure chosen ciphertext security, can lead to very powerful side-channel key recoveries relying on Plaintext-Checking Oracles (PCOs) [RRCB20, UXT+22, XPR+22, RRD+23, TUX+23]. Conceptually, they exploit the possibility of performing side-channel chosen-ciphertext attacks against the part of

[1] Such attacks have been extended to the k-trace setting in [HHP+21].

Table 1: Cortex-M4 cycle counts of ML-KEM-768 Decapsulation in kCycles, and percentage of time devoted to Keccak executions, for different number of shares d [BC22].

d	ML-KEM.Decaps	`Keccak`	Ratio (%)
2	10 018	4579	45.7
3	16 747	7225	43.1
4	24 709	10 664	43.2
5	34 683	14 897	43.0
6	45 950	19 923	43.4
7	58 473	25 752	44.0

ML-KEM that is only secure against chosen-plaintext attacks. Concretely, they exploit the possibility to force the re-encryption of key-dependent messages, so that just distinguishing two messages (i.e., instantiating a PCO) thanks to leakage can compromise ML-KEM's long-term key. As discussed in [ABF+, ABH+22], such attacks are very powerful because any leakage sample of the re-encryption step can be used to distinguish. Those seminal efforts have in turn led to various optimizations towards efficiently exploiting the information provided by PCOs. For example, early works like [RRCB20] relying on hard decisions such as majority voting gradually evolved towards probabilistic strategies [SCZ+23, HMS+23, LCS+25, HSMP25, PRI26], culminating with the recent [HMT25] which takes advantage of Soft Analytical Side-Channel Attacks (SASCA) [VGS14].

In parallel, research progressed in the direction of protecting ML-KEM implementations against these strong attacks. In particular, how to securely mask its various operations to mitigate side-channel leakage (e.g., in software, which will be our focus) has been an active research area. For Keccak, standard solutions developed for the AES can be directly used [GR17], possibly with some specialization [GD23]. For the other parts, new algorithmic solutions have been proposed, including [BPO+20, BDH+21] for polynomial comparisons, or [CGMZ22, D'A23] for conversion algorithms, and leading to a few complete proposals such as [BGR+21, BC22] (and [BDK+21] for the SABER algorithm).

As a result of these advances, it is to be expected that secure implementations of ML-KEM should be quite expensive. Yet, and it is also worth noticing that the amount of published works that consider actual measurements against protected (masked) implementations remains scarce, especially when it comes to systematically assessing security in a quantitative manner. Among the aforementioned papers, Primas et al. (CHES 2017) consider SASCA against masked software implementations of CRYSTALS-Kyber, but their (quantitative) investigations only evaluate single-trace attacks against the NTT [PPM17]. The others only assess a (qualitative) security order via leakage detection: for the polynomial comparisons in the case of Bhasin et al. [BDH+21], for complete implementations of Kyber (or SABER) in the case of Bos et al. [BGR+21], Van Beirendonck et al. [BDK+21] and Bronchain & Cassiers [BC22]. Besides, and browsing other recent works, Hermelink et al. (resp., Pay and Standaert) comprehensively evaluate the security of a masked software implementation against PCO-based attacks with SASCA-like techniques, but their (quantitative) evaluations only consider polynomial comparison [HNPS24] (resp., the NTT [PS26]); Wang et al. finally use deep learning techniques to evaluate a higher-order masked implementation of CRYSTALS-Kyber, but their (quantitative) evaluations only consider the message decoding function and the Barrett reduction steps during the decapsulation process [WBD24]; their study is extended to a masked hardware implementation in [JD25]. Somewhat surprisingly given that it accounts for a significant part of the overheads and leakage in masked implementations of ML-KEM (see Table 1), no analysis assessed the strength of PCOs instantiated against a masked Keccak so far.

Overall, and to the best of our knowledge, the state of the art therefore still lacks a systematic investigation of how strong PCO-based attacks can be against masked (e.g., software) implementations. This work addresses this gap in two main directions.

First, we evaluate message distinguishing attacks against masked software implementations of Keccak on an ARM Cortex-M4. This choice is motivated both by the fact that it is among the most important sources of leakage in ML-KEM and by the availability of open source prototypes. We consider the C implementation from Kundu et al. [KDB+22], which adapts the Domain Oriented Masking (DOM) implementation from [GSM17] to the software case, the C and assembly implementations of Bronchain and Cassiers [BC22], which rely on Probe Isolating Non Interferent (PINI) gadgets [CS20], and the Threshold (assembly) Implementation (TI) of [GD23], which is limited to two shares. This allows us to assess the impact of C implementations vs. assembly ones aimed at reducing transition-based leakages [BGG+14]. It also allows us to assess the difference between protections for arbitrary numbers of shares vs. optimized ones for two shares. We use state-of-the-art template attacks in principal subspaces for this purpose [SA08], which have been shown to be well adapted to target masked software implementations of the AES [BS21, BCS21]. Concretely, we compare our target software implementations with up to 7 shares and confirm the standard intuition that computing more implies leaking more. For the DOM implementation, which has the highest security level in our experiments, we show that high-accuracy PCOs can be easily obtained for up to 7 shares with the leakage corresponding to only 50 executions of ML-KEM. These results lead to the conclusion that ML-KEM is unlikely to be implementable securely on low-end microcontrollers without prohibitive overheads. They suggest the investigation of secure micro-controllers (e.g., with noise engines or instruction set extensions for leakage-sensitive operations) and hardware Keccak co-processors, for which parallelism could be leveraged to reduce the amount of leakage exploitable by PCOs, as interesting and practically relevant open problems. The side-channel setup for the four implementations and post-processing scripts are available at https://github.com/uclcrypto/FO_Calypse_software.

Second, we consolidate recent solutions for exploiting PCOs and turn them into full key recoveries. We start by introducing a simple and concrete model for PCOs, which we parametrize for masked implementations of Keccak with different number of shares. We then use this model to clarify that approaches based on hard decisions, like the initial [RRCB20], perform worse than soft (probabilistic) ones, like the recent [HMT25]. We show in particular that majority voting is suboptimal in the context of hard PCO-based attacks, since adversaries better use all their measurements to increase the accuracy of a single PCO call than spending them in voting. Finally, we initiate a study of the remaining challenges raised by soft approaches, namely how to best post-process the inference's outcomes in order to evaluate the security level given the adversary's computational power. Given the nature of the (lattice) assumption on which ML-KEM relies, key enumeration algorithms used in symmetric cryptography are unlikely to be sufficient [PSG16], and we therefore turn back to lattice estimators [APS15] or variants with hints thereof [DDGR20]. We show that a naive (likely conservative) use of these estimators already gives significant attack improvements and that hybrid approaches perform best in our setting [How07]. We also put forward that such estimators are for now unable to capture the granularity of the hints provided by side-channel leakages, leaving the design of a generic estimator that could efficiently leverage such physical hints as an interesting scope for further research.

2 Preliminaries

In this first section, we provide the background to understand our results.

Algorithm 1 ML-KEM.Decaps

Input: $dk = (dk_{\text{PKE}}, ek_{\text{PKE}}, h, z), c$
Output: Shared secret K'
1: $m' \leftarrow$ K-PKE.Decrypt(dk_{PKE}, c)
2: $K', r' \leftarrow$ SHA3-512$(m'||h)$
3: $\bar{K} \leftarrow \text{PRF}(z \,\|\, h, 256)$
4: $c' \leftarrow$ K-PKE.Encrypt$(ek_{\text{PKE}}, m', r')$
5: **if** $c' \neq c$ **then**
6: $\quad K' \leftarrow \bar{K}$
7: **end if**
8: **return** K'

Algorithm 2 K-PKE.Decrypt

Input: $dk_{\text{PKE}} = \boldsymbol{sk}, c$
Output: Decrypted message m'
1: $\boldsymbol{u}', v' \leftarrow \text{Decompress}(c)$
2: $\mu \leftarrow v' - \langle \boldsymbol{sk}, \boldsymbol{u}' \rangle$
3: $m' \leftarrow \text{Compress}(\mu, 1)$
4: **return** m'

Algorithm 3 K-PKE.Encrypt

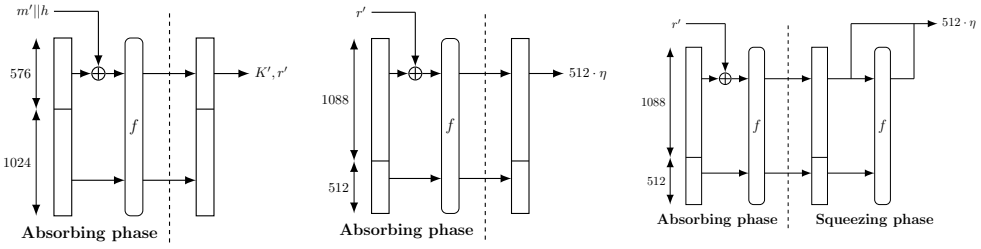
Input: $ek_{\text{PKE}} = (\boldsymbol{t}, \rho), m', r'$
Output: ciphertext $c' = (c_1, c_2)$
1: $\boldsymbol{A} \in \mathcal{R}_q^{k \times k} \leftarrow \mathcal{U}(\rho)$
2: $\boldsymbol{y} \leftarrow \boldsymbol{0} \in \mathcal{R}_q^k$
3: $\boldsymbol{e_1} \leftarrow \boldsymbol{0} \in \mathcal{R}_q^k$
4: **for** $i = 1$ to k **do**
5: $\quad \boldsymbol{y}[i] \leftarrow \text{CBD}_{\eta_1}(\text{PRF}(r', 512 \cdot \eta_1))$
6: $\quad \boldsymbol{e_1}[i] \leftarrow \text{CBD}_{\eta_2}(\text{PRF}(r', 512 \cdot \eta_2))$
7: **end for**
8: $e_2 \in \mathcal{R}_q \leftarrow \text{CBD}_{\eta_2}(\text{PRF}(r', 512 \cdot \eta_2))$
9: $\mu \leftarrow \text{Decompress}(m', 1)$
10: $\boldsymbol{u} \leftarrow \boldsymbol{A}^T \boldsymbol{y} + \boldsymbol{e_1}$
11: $v \leftarrow \langle \boldsymbol{t}, \boldsymbol{y} \rangle + e_2 + \mu$
12: $c_1 \leftarrow \text{Compress}(\boldsymbol{u}, d_u)$
13: $c_2 \leftarrow \text{Compress}(v, d_v)$
14: **return** c'

Figure 1: Simplified version of the ML-KEM Decapsulation algorithm, where $\text{PRF}(M, b)$ corresponds to $\text{SHAKE256}(M, b)$ with input message M and b-bit output length.

2.1 ML-KEM

ML-KEM is a key encapsulation mechanism standardized by the NIST [Nat24]. Its security relies on the hardness of the Module Learning With Errors (MLWE) problem and it is instantiated for the three NIST security levels (i.e., L1, L3 and L5) under the names ML-KEM-512, ML-KEM-768 and ML-KEM-1024. All standardized instantiations operate on the same polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$, with prime modulus $q = 3329$ and ring degree $n = 256$. This ring is used to construct the module $\mathcal{R}_q^k$ whose rank k equals 2, 3 and 4 depending on the security level. The parameter sets η_1 and η_2, which define the distributions used to sample the secret and error polynomials, are chosen as $\eta_1 = (3, 2, 2)$ and $\eta_2 = (2, 2, 2)$ for the security levels (L1, L3, L5). ML-KEM is constructed from a Public-Key Encryption (PKE) scheme which is transformed into an IND-CCA2-secure KEM using the FO transform. In this paper, we focus on the decapsulation algorithm of ML-KEM, which is the target of message distinguishing attacks. We briefly recall this decapsulation procedure below and refer the reader to [Nat24] for the details.

ML-KEM Decapsulation. The decapsulation procedure of ML-KEM is depicted in Algorithm 1. Given the decapsulation key $dk_{\text{PKE}} = \boldsymbol{sk}$, which is a secret coefficients' vector where each coefficient is denoted by $\boldsymbol{sk}_i$ with $i \in \{1, n \cdot k\}$, and a received ciphertext c, the algorithm first recovers the decrypted message m' using Algorithm 2. This message is then re-encrypted with the encapsulation key (i.e., the public key) ek_{PKE} and some randomness r' using Algorithm 3, in order to obtain a re-computed ciphertext c'. The decapsulation procedure finishes by outputting the shared secret key K', if the recomputed ciphertext c' matches the received ciphertext c, or a random value $\bar{K}$, when the ciphertexts are different. This re-encryption step is essential for achieving IND-CCA2 security. It ensures that the randomness r' (which is deterministically derived from the message) used during the encapsulation and decapsulation procedures is identical. This is checked by verifying that the original ciphertext is matching with the re-computed one.



(a) SHA3-512($m'||h$) (b) SHAKE256($r', 512 \cdot 2$) (c) SHAKE256($r', 512 \cdot 3$)

Figure 2: Sponge construction of the attacked SHA-3 functions.

2.2 SHA-3

The SHA-3 algorithm is a family of four hash functions and two eXtendable-Output Functions (XOFs) standardized by the NIST [oSN15]. In this work, we will adopt the same terminology as in the FIPS 202 documentaion [oSN15]. The SHA-3 functions sensitive to message distinguishing attacks (highlighted in Algorithms 1 and 2) correspond to the hash function SHA3-512 and the XOF SHAKE256 defined as:

$$\text{SHA3-512}(m'||h) = \text{Keccak}[1024](m'||h, 512),$$
$$\text{SHAKE256}(r', 512 \cdot \eta) = \text{Keccak}[512](r', 512 \cdot \eta).$$

The input size of both functions is constant across all ML-KEM security levels, with $|(m'||h)| = 512$ bits and $|r'| = 256$ bits. In contrast, while the output length of SHA3-512 is fixed to 512 bits, the output length of SHAKE depends on η, which varies depending on the security levels. Both functions use the Keccak-$f[1600]$ permutation (next simply referred to as Keccak) in sponge constructions depicted in, Figure 2. The number of Keccak permutation executions that can be exploited by a PCO on ML-KEM re-encryption is 8, 8 and 10 for ML-KEM-512, ML-KEM-768 and ML-KEM-1024 respectively.

Permutation. The Keccak permutation consists of 24 rounds, each composed of five sequential operations: θ, ρ, π, χ and ι. All operations apply a linear transformation to the state, except for χ, which is a non-linear transformation. We omit the detailed description of $(\theta, \rho, \pi, \chi, \iota)$ because they are not necessary for understanding our results. We just note that in software implementations, the ρ and π operations are often combined together for efficiency, reducing the number of operations per round to four: $(\theta, \pi \circ \rho, \chi, \iota)$.

2.3 Profiled attack on masked implementations

Our methodology for building PCOs is similar to the approach proposed in [BS21]. In this section, we recall its main steps and tools. At a high level, this methodology identifies secret-dependent variables in the computation, which we name *sensitive variables*. We then target the shares of these variables with a profiled attack to retrieve probabilistic information on their values. We finally recombine the shares' distributions to learn a distribution of the sensitive variables, which we translate to the main secret.

The simplicity of this method rests on the knowledge of the shares' values in a profiling dataset. It is instrumental to evaluate large numbers of shares and confirm scaling trends from a designer/evaluator's viewpoint (which is our goal). When such a knowledge is not available, deep learning techniques [PPM+23, MCLS23] can often be used as an alternative to achieve similar results. We note that, in a similar worst-case evaluation spirit, we perform our profiling campaigns and attacks against the same device, ignoring templates' portability challenges. We do this based on [BS21], which showed that inter-device variability only gives heuristic security margins and cannot serve as a sound security parameter.

Point-of-interest (POI) selection. The first step is to select the time samples that will be used for the attack, for each share of every sensitive variable. We use the SNR as a standard leakage detection tool [Man04] for this purpose. It models the signal as the variance of the mean leakage of each class x and the noise as the average intra-class variance:

$$\hat{\mathrm{SNR}} = \frac{\hat{\mathbb{V}}_x(\hat{\mathbb{E}}_i(l_i^x))}{\hat{\mathbb{E}}_x(\hat{\mathbb{V}}_i(l_i^x))},$$

where $\hat{\mathbb{V}}$ and $\hat{\mathbb{E}}$ denote the sample variance and the sample mean, estimated on $l_i^x \in \mathcal{L}$, with l_i^x the i-th side-channel leakage with intermediate value x. The SNR can be computed independently for each time sample of a leakage trace. High SNR indicates locations where (first-order) leakage is present and can be exploited in an attack.

Gaussian templates in principal subspaces. After selecting the POIs, Gaussian templates are used to model the multivariate leakages [CRR02]. They operate by first estimating a Probability Density Function (PDF) $f[x|\boldsymbol{l}]$, where $\boldsymbol{l}$ is a n_{POIs}-dimensional leakage. This is done by modeling it as a multivariate Gaussian distribution:

$$\hat{f}[\boldsymbol{l}|x] = \frac{1}{\sqrt{(2\pi)^n |\hat{\boldsymbol{\Sigma}}|}} \exp^{-\frac{1}{2}(\boldsymbol{l}-\hat{\boldsymbol{\mu}}_x)^\top \hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{l}-\hat{\boldsymbol{\mu}}_x)}, \tag{1}$$

where $\hat{\boldsymbol{\mu}}_x$ is the leakage mean for each value of x and $\hat{\boldsymbol{\Sigma}}$ is the pooled covariance matrix, both estimated using a profiling dataset for which the shares' values are known. If x is uniformly distributed, the distribution of x given $\boldsymbol{l}$ is then computed as:

$$\hat{\mathrm{Pr}}_{\mathrm{model}}[x|\boldsymbol{l}] = \frac{\hat{f}[\boldsymbol{l}|x]}{\sum_{x' \in \mathcal{X}} \hat{f}[\boldsymbol{l}|x']}. \tag{2}$$

When the leakage dimension n_{POIs} is large, accurately estimating $\hat{\boldsymbol{\mu}}_x$ and $\hat{\boldsymbol{\Sigma}}$ may require a prohibitive amount of data. To reduce this profiling effort, leakage traces can be projected onto a lower-dimensional linear subspace of dimension n_{LDA}, with $n_{\mathrm{LDA}} \ll n_{\mathrm{POIs}}$, using Linear Discriminant Analysis (LDA) [SA08]. LDA concentrates the side-channel information into a few dimensions by optimizing the SNR after projection: it computes a SNR-maximizing orthogonal projection $\boldsymbol{W}$ and updates Equation 1 with projected leakage $\boldsymbol{l}' = \boldsymbol{W}\boldsymbol{l}$. In practice, the initial number of dimensions n_{POIs} is chosen as the number of time samples with large enough SNR and the (reduced) number of LDA dimensions n_{LDA} is a trade-off between model performance and number of traces needed for profiling.

In order to optimize the model's hyper-parameters and the number of profiling traces, we use information-theoretic metrics. A standard metric for evaluating the amount of information that concrete leakages provide about an intermediate variable is the Mutual Information (MI) between this variable and the leakage [SMY09]. However, estimating the MI is challenging in practice, since it requires an accurate knowledge of the true leakage distribution. The Perceived Information (PI) is a lower-bound of the MI between a variable X and its the leakage $\boldsymbol{L}$, and it describes how much information a trained model can recover about the variable from the leakage [BHM+19]. It is defined as:

$$\hat{\mathrm{PI}}(X;\boldsymbol{L}) = \mathrm{H}(X) + \sum_{x \in \mathcal{X}} \mathrm{Pr}[x] \sum_{\boldsymbol{l} \in \mathcal{L}_{\mathrm{test}}} \frac{1}{|\mathcal{L}_{\mathrm{test}}|} \log_2 \hat{\mathrm{Pr}}_{\mathrm{model}}[x|\boldsymbol{l}], \tag{3}$$

where $\mathrm{H}(X)$ denotes the entropy of X and $\mathcal{L}_{\mathrm{test}}$ is a set of (fresh) leakage samples that is not used during the profiling phase. In order to assess whether a given model has converged, a convenient complementary metric is the Training Information (TI), which provides an upper bound to the PI [MCHS23]. The TI is computed similarly to Equation 3, but the test set $\mathcal{L}_{\mathrm{test}}$ is replaced by the profiling set used to estimate the model.

Secret recovery. Given a leakage trace $\boldsymbol{l}$ and a d-share Boolean encoding $(V_1, \ldots, V_d)$ of a sensitive variable $V \in \mathbb{F}_{2^k}$, such that $V = V_1 \oplus \cdots \oplus V_d$, the Gaussian models described above give the shares posteriors:

$$\hat{\boldsymbol{p}}_V^i = \left(\hat{\Pr}[V_i = v_i \mid \boldsymbol{L} = \boldsymbol{l}]\right)_{v_i \in \mathbb{F}_{2^k}}.$$

We can then compute the posterior of the sensitive variable:

$$\hat{\boldsymbol{p}}_V = \left(\hat{\Pr}[V = v \mid \boldsymbol{L} = \boldsymbol{l}]\right)_{v \in \mathbb{F}_{2^k}} = \hat{\boldsymbol{p}}_V^1 \star \cdots \star \hat{\boldsymbol{p}}_V^d, \tag{4}$$

where $\star$ is the convolution product. A naive implementation of Equation 4 has complexity $\mathcal{O}(2^{2k}d)$, which can be costly (even though it remains practical for small parameters like $k = 8, d \leq 7$). Therefore, we use the optimized implementation based on the Walsh-Hadamard transform [KPP20] from SCALib [CB23], with complexity $\mathcal{O}(2^k kd)$. Eventually, we combine information in different sensitive variables $\left(V^j\right)_{j=1,\ldots,n}$ by assuming that the leakages on different variables are independent. For a uniformly-distributed secret input K and a public input T, it gives:

$$\hat{\Pr}[K = k \mid \boldsymbol{V} = \boldsymbol{l}, T = t] = \frac{\prod_{j=1}^{n} \hat{\Pr}[V^j = f_j(k,t) \mid \boldsymbol{V} = \boldsymbol{l}]}{\sum_{k'} \prod_{j=1}^{n} \hat{\Pr}[V^j = f_j(k',t) \mid \boldsymbol{V} = \boldsymbol{l}]}, \tag{5}$$

where function $f_j(k,t)$ is the value of V^j when $K = k$ and $T = t$. If multiple leakage traces are available to the adversary, the posteriors are multiplied similarly.

3 Distinguishing attacks on masked Keccak

In this first main section, we present distinguishing attacks against different masked implementations of Keccak that can be used in order to instantiate PCOs.

3.1 Target implementations and setup

We first present our target implementations and the measurement setup we used.

Target Implementations. To the best of our knowledge, there are only three open source masked software implementations of Keccak [BC22, KDB+22, GD23]. These implementations, written in C or assembly (ASM), mainly differ in the strategy used to mask the non-linear χ layer of Keccak (as the linear layers can be performed share-wise). We next refer to these implementations as PINI, DOM and TI:

PINI. The Probe-Isolating Non-Interferent (PINI) implementation from [BC22] relies on PINI gadgets to mask the non-linear operation χ. It can be masked with any number of shares and is written in C, with sensitive PINI gadgets implemented in C or assembly. The ASM version explicitly controls register allocation and instruction scheduling to mitigate transition-based leakages between shares, which may occur with the gadgets written in C due to compiler optimizations [BGG+14].

DOM. The Domain-Oriented Masking (DOM) implementation from [KDB+22] uses DOM rather than PINI gadgets to protect the non-linear operation χ. It is written in C and supports any number of shares (like the PINI implementation).

TI. The Threshold Implementation (TI) described in [GD23] is a hand-optimized ASM implementation that adapts the TI concept to software. It is designed to prevent mirco-architectural leakage (e.g., due to transitions or glitches). Unlike PINI and DOM, this implementation is optimized for and only supports two shares.

Table 2: Cortex-M4 cycle counts for one Keccak round across different implementations and number of shares d, compiled using `GCC` 13.3.0 with `-O3` optimization flag.

d	PINI C	PINI ASM	DOM C	TI ASM
2	21 406	21 746	5350	6336
3	31 198	34 822	10 030	–
4	43 286	50 594	18 450	–
5	58 534	70 162	24 350	–
6	76 642	93 378	32 894	–
7	96 784	119 480	42 312	–

Table 2 reports the runtimes of a single Keccak round for implementations up to 7 shares. The runtime of one Keccak execution corresponds to these values multiplied by 24.

Measurement Setup. Measurements are performed on an STM32F415RG (ARM Cortex-M4) mounted on a ChipWhisperer CW308 UFO board and the firmware binaries are compiled using `GCC` 13.3.0 with `-O3` optimization. The processor operates at a 125 MHz clock frequency, derived from an external 15.625 MHz square wave clock generated by a PicoScope 5244D. Side-channel leakage is captured, with the same Picoscope, using a Tektronix CT-1 current probe placed on a jumper between the on-board power regulator and the CPU. Traces are sampled at 500 MSamples/s with a 12-bit resolution.

3.2 Attack methodology

Our distinguishing attack against the Keccak permutation is based on the methodology highlighted in Section 2.3 and implemented using SCALib [CB23]. This methodology is "order-preserving" in the sense that it recombines the shares' information by applying a convolutional product, which prevents exploiting (e.g., transition-based) flaws that may re-combine this information at the implementation level, and reduce the security amplification that masking ideally provides. This means that the attacks we describe next may be further improved in case such flaws are observed. We leave the investigation of such improvements as an open problem because they apply similarly to any gadget and the same holds for solutions to prevent them [CGP+12, BGG+14, CS21]. Besides, previous work showed that for low-noise platforms like the ARM Cortex-M4 devices that we consider, such order-preserving attacks are usually the most efficient [BS21].

As attack target, we selected the Keccak permutation state at the output of the θ, $\pi \circ \rho$ and χ layers.[2] The sensitive Keccak permutations, highlighted in Figure 1, can be divided in two cases. In the first Keccak call of Algorithm 1, the input depends directly on the message m' which can be chosen by the adversary. In the subsequent Keccak invocations of Algorithm 3, the inputs depends on the randomness r'. We consider a distinguishing setting where m' can take only two different values. Consequently, r' also takes only two possible values, fully determined by the chosen message m' and the public value h.

In order to better match the (independence) assumption we leverage when recombining the side-channel information of multiple rounds of the Keccak permutation in the first case, we systematically discard the first four rounds (which ensures full diffusion for a single-bit input difference). For simplicity, we do the same for the second case. As a result, our analysis reduces to distinguishing between two random-looking target state distributions, regardless of whether the corresponding inputs differ in a single or many bits.

[2] The ι layer adds a constant to the state such that target intermediate variables corresponding to the outputs of χ and ι are bijectively related and can be exploited with the same templates.

Table 3: Number of Keccak permutation executions and total number of leaking Keccak rounds exploitable for PCO instantiation, for the different versions of ML-KEM.

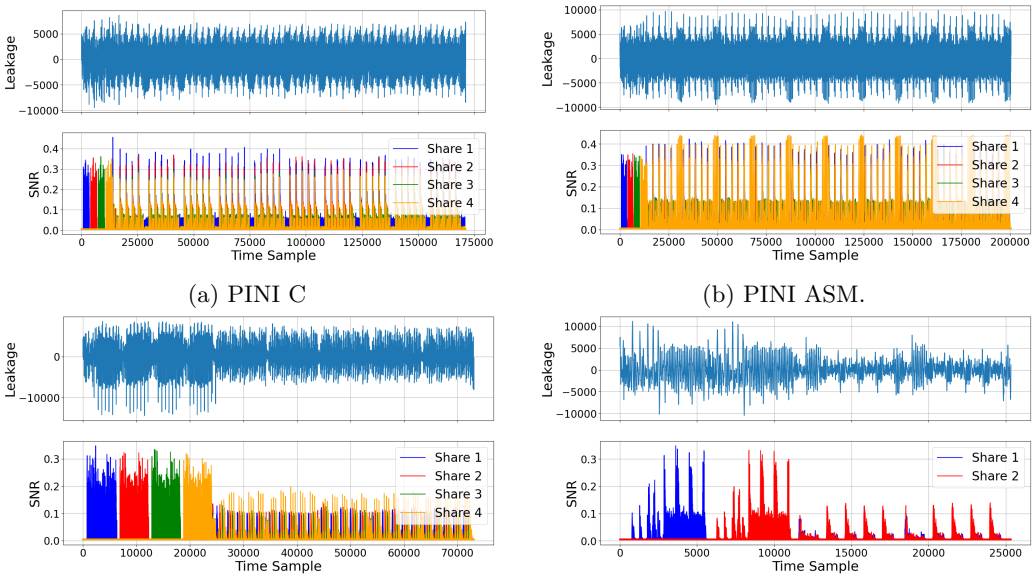
	# Keccak executions	Leaking rounds
ML-KEM-512	8	160
ML-KEM-768	8	160
ML-KEM-1024	10	200

This leaves us with 20 exploitable leaking rounds per Keccak execution, leading to the exploitable round counts for the whole ML-KEM re-encryption given in Table 3. For the sake of simplicity, we partition each 1600-bit state it into 200 independent 8-bit sensitive variables. This results in $3 \times 200 = 600$ sensitive values per round and $600 \times 20 = 12\,000$ values per Keccak execution. It is possible to further increase the number of intermediate value. However, in this work we restrict ourselves to the state variables to enable a generic comparison across the four target implementations. Alternatively, adopting 32-bit variables could help reduce algorithmic noise. Again, we leave these extensions (i.e., incorporating additional intermediate values and exploring 32-bit models) for future work.

In the first stage of the attack, we compute the shares' SNR in order to select the POIs. In practice, the leakage traces corresponding to a single Keccak execution become quite large when the number of shares increases. For example, with our measurement setup, one Keccak execution using the DOM C implementation with $d = 6$ shares produces more than 4×10^6 leakage samples, corresponding to approximately 6 MB of data per trace. Given the large number of shares to profile, handling such traces directly results in substantial computational costs. In order to mitigate this issue, we store and process the leakage on a per-round basis: we split each Keccak trace into sub-traces corresponding to a round. This optimization may reduce the success rate of our distinguisher by eliminating potential leakage of a variable that spills over neighboring rounds, but was deemed worthwhile since it results in reducing the traces' length by a factor 20 in the SNR computation stage.

During the profiling phase, a set of 2×10^5 traces of Keccak executions was recorded for each target implementation. The input was randomly selected from two different message values, then masked, and the senstive values of all shares were recorded throughout the Keccak execution. The goal of the profiling phase is to characterize the intermediate computations independently of the particular input value. After splitting the traces into rounds, the SNR was computed and used to identify the POIs of all the shares for each sensitive value. We then selected n_{POIs} sample points exhibiting the highest SNR values. For illustration, the shares' SNR computed on the leakage traces obtained from our four different targets is shown in Figure 3 for the θ operation. In this figure, both the linear (i.e., $\theta, \pi \circ \rho$) and non-linear (i.e., χ) transformations are clearly distinguishable from the SNR curves. The linear operations are performed independently on each share, whereas the non-linear operation requires the joint processing of all shares. Each SNR plot associated with a given share represents the SNR of its 200 sensitive variables, all displayed using the same color. Consequently, peaks appear throughout the entire Keccak round.

Using the same training dataset restricted to the selected POIs, we estimated the parameters of the LDA-based template model described in Equation 1. The parameters n_{POIs} and n_{LDA} are chosen in order to maximize the perceived information (PI) of the resulting model. We analyzed the PI of the LDA models for n_{POIs} values up to 1600 and n_{LDA} values up to 8, and observed that the best overall results were obtained with $n_{\text{POIs}} = 1600$ and $n_{\text{LDA}} = 4$ across all implementations. For example, the PI and TI for the 4-share DOM implementation is shown in Figure 4a: increasing n_{LDA} up to 4 makes the model more informative but pushing it further is detrimental. Indeed, models with



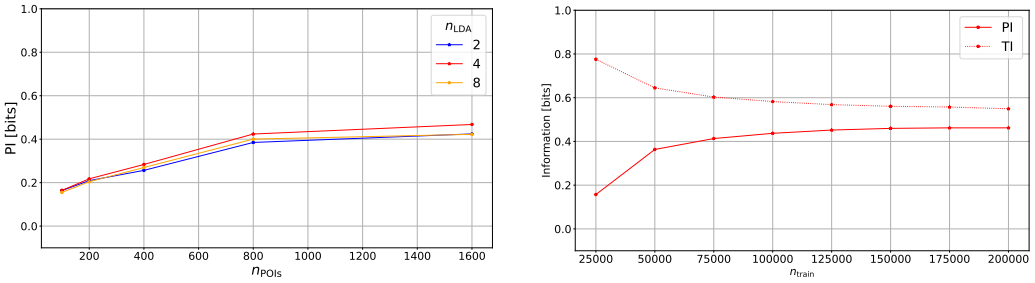
(a) PINI C

(b) PINI ASM.

(c) DOM C.

(d) TI ASM.

Figure 3: Leakage traces and corresponding SNRs for the intermediate values after θ in one round. Implementation masked with 4 shares (2 shares for the TI).



(a) PI curves in function of n_{POIs} and n_{LDA}.

(b) PI/TI curves for $n_{\text{POIs}} = 1600$ and $n_{\text{LDA}} = 4$.

Figure 4: Information metrics on a share of the DOM $d = 4$ implementation.

larger parameters need more profiling traces to converge, and the plots of Figure 4b suggest that models have not fully converged already with $n_{\text{POIs}} = 1600$ and $n_{\text{LDA}} = 4$. Hence, larger n_{POIs} and n_{LDA} values would require even more profiling traces.

Next, we collected an attack dataset of $n_{\text{atk}} = 10^4$ traces in order to evaluate the accuracy of our distinguishers. For this dataset, the input of the Keccak permutation is sampled either as the all-zero state $\{0\}^{512}$ or as the all-zero state except for a single bit flip among the first 256 bits, in order to emulate the Keccak message-sensitive part of Algorithm 1. Any pair of distinct input states could be used equivalently since the profiling phase is designed to be input-independent, as argued above. We can evaluate the success rate of each distinguisher by repeatedly applying the methodology of Section 2.3 (Equations 1, 2, 4 and 5) with a varying amount of traces. We note that the above attacks need to acquire and store large datasets, in particular for their training. As the number of shares increases, the total size of the training data quickly becomes prohibitive. For instance, as shown in Table 4, the size of training datasets can exceed 1 TB, for a total size across all datasets of approximatively 20 TB. Therefore, and as an optimization, we

Table 4: Memory footprint of each dataset in GB. Each dataset consists of 200k training traces and 10k attack traces (20 Keccak rounds per trace).

d	PINI C		PINI ASM		DOM C		TI ASM	
	Train	Atk	Train	Atk	Train	Atk	Train	Atk
2	632	32	672	40	163	8	196	10
3	930	47	1 040	52	303	15	–	–
4	1 362	67	1 520	76	544	28	–	–
5	1 830	88	2 112	106	724	36	–	–
6	2 404	122	2 806	142	1 010	48	–	–
7	–	–	–	–	66*	33*	–	–

* For the DOM 7-share case, only the optimized dataset with a single round per Keccak execution was measured and stored. The number of attack traces was increased to 100k.

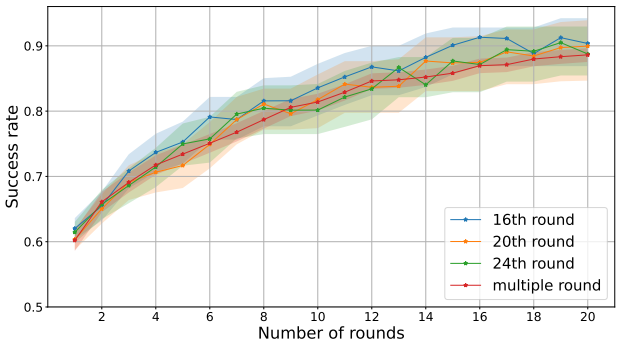
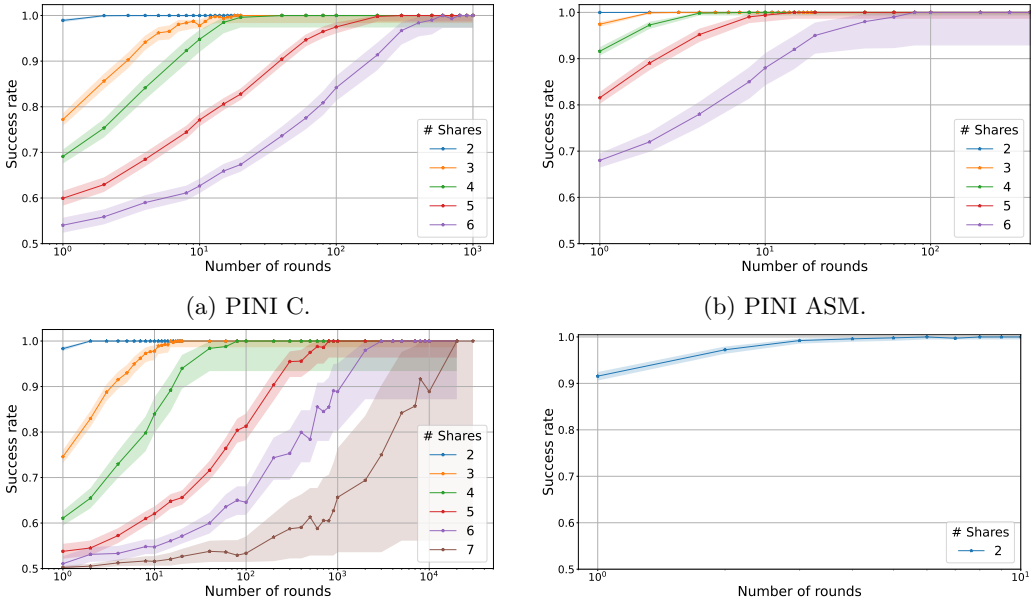


Figure 5: Equivalence between traces and round aggregation (DOM C, 4 shares).

considered a technique to reduce the size of datasets by measuring and storing only one round of the Keccak execution, instead of 20. Thanks to the diffusion in Keccak, we expect that all rounds leak the same amount of information and, moreover, that the states of consecutive rounds are sufficiently decorrelated, such that their leakage is not redundant. In order to validate this hypothesis, we compared in Figure 5 the success rate of attacks that aggregate the leakage of consecutive rounds of a single Keccak execution and attacks that use the leakage of a single round from multiple independent traces, showing that this success rate is essentially the same in both experiments. This equivalence between leakage of rounds of a single traces and rounds of different traces also allowed us to simplify the evaluation of the distinguisher: we next evaluate the success rate as a function of a number of rounds (independently of whether they come for the same execution of Keccak).

Distinguisher accuracy The accuracy of our distinguishers on the different implementations for up to 6 shares (and 7 shares for DOM C) is given in Figure 6.

A first observation is that "computing more implies leaking more". That is, when comparing different masking techniques (or the C vs. assembly implementations) with the same number of shares, the faster implementations, as reported in Table 2, are marginally harder to attack. This explains why the PINI C implementation is slightly more "secure" than the PINI ASM one: the instructions added to prevent lower-order transition-based leakage result in an increased amount of leakage on the shares. Furthermore, the PINI C implementation provides stronger composability guarantees than the DOM C one (proving the absence of lower-order leakage due to gadget compositions), at the cost of



(a) PINI C.

(b) PINI ASM.

(c) DOM C.

(d) TI ASM.

Figure 6: Success rates of message distinguishing attacks against masked implementations with different number of shares. The x-axis varies between the figures.

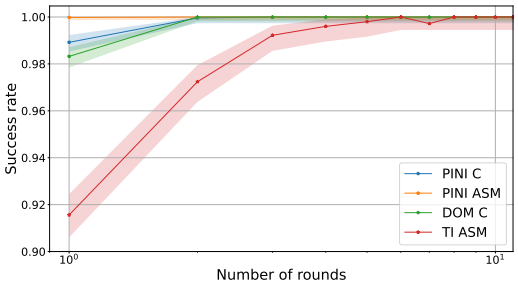


Figure 7: Success rates of message distinguishing attacks against 2-share implementations.

more computation, and thus leakage, on the shares. Finally, we remark that the TI implementation leaks less than the others due to its 2-share specialization (see Figure 7), although this specialization also makes it trivial to break in our context.

The second observation is that these implementations all lead to successful message distinguishing attacks with low complexities. For example, the most "secure" implementation in our experiments is the 7-share DOM one, for which a 90 % accuracy distinguisher is obtained with the leakage of 10^4 Keccak rounds. Given the counts in Table 3, this only corresponds to 50 ML-KEM executions. We conclude that in order to ensure security against message distinguishing attacks without prohibitive overheads, one would highly benefit from implementations in secure microcontrollers with noise engines and instruction set extensions for sensitive operations (e.g., Keccak), or from hardware co-processors in which parallelism could limit the amount of leakage exploitable by such attacks.

4 Exploiting plaintext-checking oracles

As described in [RRCB20, QCZ+21], an adversary can easily craft ciphertexts such that the decrypted message m' in Algorithm 1 takes one out of two known values, next denoted as m_0 and m_1. For instance, m_0 can be the zero bit vector whereas m_1 is identical except for a single bit set to one. The decrypted message m' then takes one of these two values depending on whether a chosen secret-key coefficient exceeds a given threshold τ. More precisely, for any $i \in \{1, n \cdot k\}$ and threshold τ, the adversary can construct a ciphertext c such that the value of m' is m_b, where $b = 0$ if $\boldsymbol{sk}_i \leq \tau$ and $b = 1$ otherwise.

Recovering b therefore reveals information on the secret key coefficient $\boldsymbol{sk}_i$ and its actual value can be narrowed down by crafting ciphertexts for different thresholds. To determine b, the adversary can exploit the re-encryption leakage and, in particular, all the Keccak calls highlighted in Figure 1. As shown in Table 3, this re-encryption leakage corresponds to 8 (resp., 10) calls to Keccak for ML-KEM-512/768 (resp., ML-KEM-1024). Omitting the first 4 rounds of each permutation, as explained in Section 3.2, this gives us a total of 160 (resp., 200) Keccak round leakages per ML-KEM decapsulation. These rounds can be leveraged using the Keccak-based message distinguisher of Section 3, which is in turn used to instantiate the PCO for distinguishing between $b = 0$ and $b = 1$.

In this section, we focus on full key-recovery attacks against the long-term secret of ML-KEM-768. We illustrate our approaches with the DOM C implementation (i.e., the one which is the hardest to attack) and we use the number of ML-KEM decapsulations M needed to reach a (full) key-recovery success rate SR_{KR} as our main figure-of-merit.

We begin by constructing a parametric model for soft PCOs (i.e., PCOs whose output is a probability distribution) based on side-channel leakage. We fit this model to the distinguishers of Section 3, showing a good match between the models and the real distinguishers. This model simplifies the analysis of key recovery strategies at high success rates, since accurately estimating the success rates in these conditions requires (very) large datasets. Furthermore, we argue that it comes with sound theoretical grounding and empirical validation with concrete instances of parametrizations. It is therefore an easy-to-use benchmark for future work on the design of PCO-based attacks.

We next investigate three attack strategies exploiting PCOs. First, we directly use the output of the message distinguisher as hard information to instantiate the PCO, optionally with majority-voting as proposed in [RRCB20, UXT+22]. Then, we consider a more recent attack strategy [HMT25] that treats the message distinguisher outputs as soft information and combines multiple PCO queries probabilistically. Finally, we investigate how much state-of-the-art cryptanalysis techniques covered by lattice estimators [APS15] or variants with hints thereof [DDGR20] can further improve these cryptanalytic results.

4.1 A simple and re-usable model for message distinguishers

Evaluating many side-channel attacks against ML-KEM-768's full key may require a large number of oracle calls, and the corresponding acquisition of many leakage traces. Since this quickly becomes extremely time- and storage-consuming as the number of shares d increases, it motivates the introduction of a model for message-distinguishing oracles that can then be used to compare implementations and attack methodologies. In this section, we introduce such a model, ground it on realistic assumptions and validate its excellent fit with the message-distinguishers from Section 3. At high-level, our model is similar to the one derived in [LPR+14], specialized to the distinguishing (i.e., two-class) situation.

Let us denote the output probability of a message-distinguishing oracle that uses R Keccak rounds by $\hat{p}_R^b(x)$ when the input message is $m_b = b$. We define $\lambda_R^b = \log_2\left(\frac{\hat{p}_R^b(b)}{\hat{p}_R^b(1-b)}\right)$

which corresponds to the ratio between probabilities assigned to the correct and incorrect hypotheses. When considering one Keccak round, we remark that λ_1^b nearly follows a Gaussian distribution, as shown experimentally in Figure 8. This observation is grounded in the structure of the Keccak round distinguisher, which evaluates a product of 600 single-byte likelihoods (see Section 3.2 and Equation 5). These likelihoods exhibit low correlation, as they are derived from different sharings and follow similar distributions. Consequently, their aggregation naturally leads to an approximately Gaussian behavior based on Central Limit Theorem (CLT). Furthermore, since the sensitive values inside Keccak can be viewed as pseudo-random (with b as a seed) and since λ_1^b only depends on b through those sensitive values, we have identical distributions for λ_1^0 and λ_1^1 for some μ and σ^2:

$$\lambda_1^b \sim \mathcal{N}(\mu_1, \sigma_1^2).$$

Experimentally, we confirm visually in Table 5 that the means and variances are the same for both classes, for different number of shares on the DOM C implementation.

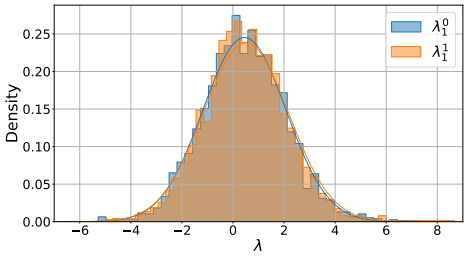


Figure 8: Empirical dist. and Gaussian approx. of λ_1^b (DOM C imple., 4 shares).

d	μ_1^0	μ_1^1	μ_1	$(\sigma_1^0)^2$	$(\sigma_1^1)^2$	$(\sigma_1)^2$
2	2.41e1	2.64e1	2.53e1	1.47e2	1.63e2	1.57e2
3	2.44e0	2.58e0	2.51e0	1.45e1	1.41e1	1.43e1
4	4.49e-1	5.03e-1	4.76e-1	2.64e0	2.80e0	2.72e0
5	3.68e-2	3.18e-2	3.43e-2	1.71e-1	1.58e-1	1.64e-1
6	8.11e-3	8.97e-3	8.54e-3	3.51e-2	4.10e-2	3.81e-2
7	7.26e-4	7.20e-4	7.22e-4	4.28e-3	4.16e-3	4.22e-3

Table 5: Mean and variance of λ_1^b for DOM C implem. with different number of shares d.

It then follows from Equation 5 that λ_R^b is a sum of R i.i.d. λ_1^b variables (assuming independent leakage for the rounds, as validated in Section 3.2), and therefore:

$$\lambda_R^b \sim \mathcal{N}(R\mu_1, R\sigma_1^2).$$

The success rate of the message-distinguishing oracle is finally given by:

$$SR_{\lambda_R^b} = \Pr[\lambda_R^b > 0], \tag{6}$$

which is easily computed from the Gaussian Cumulative Distribution Function (CDF).

Concretely, as we consider the whole ML-KEM-768, we are interested in the modeling of λ_{160D}^b, where D denotes to the number of ML-KEM decapsulations used to build the oracle (160 being the number of exploitable Keccak rounds, see Table 3).[3] As a final validation of our model, Figure 9 compares the success rate of real message distinguishers from Section 3 with the ones predicted by Equation 6 from the parameters of Table 5.

We can further simplify our model if we assume that it is well-calibrated, i.e.,

$$\lambda_1^b = \frac{\Pr[\boldsymbol{l}' \mid B = b]}{\Pr[\boldsymbol{l}' \mid B = 1 - b]},$$

where $\boldsymbol{l}'$ is some observation. Note that this assumption does not require to have an optimal oracle that computes the log-likelihood ratio of the leakage: $\boldsymbol{l}'$ can be any observable derived from the leakage, which can be less informative than the true leakage. However, it means

[3] So the M (resp., D) notation is for the complexity of full key recovery (resp., message distinguishing).

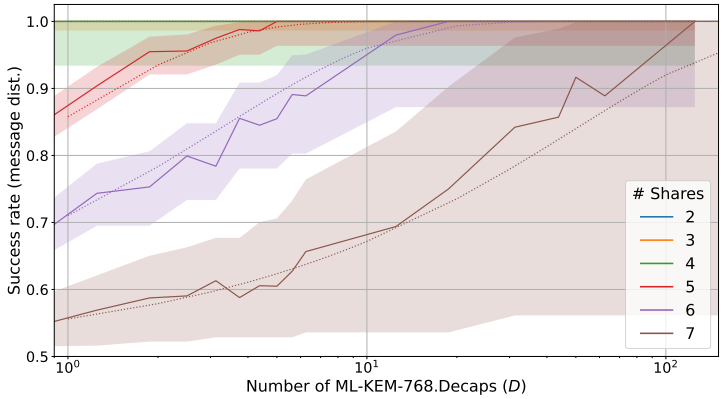


Figure 9: Success rate of message dist. attack using experimental (plain) and λ_{160D} (dotted) message-distinguishing oracle on DOM C implementation. The Success rate curves for shares 2 to 4 are directly stuck at one and therefore superimposed.

that the distinguisher cannot be biased, or be over-confident in its output distribution. From the well-calibrated assumption, it can be proven [vLB13] that:

$$\lambda_1^b \sim \mathcal{N}(\mu, 2\mu), \tag{7}$$

reducing the model to a single parameter. When looking at our distinguishers' parameters in Table 5, we observe that they do not perfectly fit this model, with σ^2 a bit larger than expected: $\sigma^2 \approx 6\mu$. Such a result is expected from an over-confident model built by aggregating correlated sub-distinguishers: assume that a well-calibrated distinguisher is used three times in Equation 5, we get $3\lambda_1^b \sim \mathcal{N}(3\mu, 18\mu)$, with a mean to standard deviation ratio that matches our experiments. Therefore, we hypothesize that our distinguishers are a bit over-confident, due to targeting sensitive variables whose shares are sometimes strongly correlated (e.g., the inputs and outputs of $\pi \circ \rho$ are very similar).

For future works that develop side-channel attacks based on soft PCOs, our baseline model (resp., the simplification proposed in Equation 7) should be relevant and convenient to re-use. Indeed, such models have two (resp., one) parameter(s) that can easily be related to the widely-used success rate metric, which reduces the need for arbitrary choices of parameters. Moreover, they are based on reasonable assumptions. Namely, our baseline model assumes that the distinguisher is a maximum-likelihood combination of independent distinguishers (e.g., on multiple executions of Keccak, and/or on different states within one execution of Keccak). Its simplification further assumes that the distinguishers are well-calibrated, which is a natural goal when optimizing soft distinguishers.

4.2 Standard approaches for full key recovery

A PCO can be instantiated from the message-distinguishing oracle λ_{160D}^b described in the previous section. Given a ciphertext crafted for a threshold τ and its associated decapsulation leakage trace, the PCO leverages the output of λ_{160D}^b to infer information about the predicate $\boldsymbol{sk}_i \leq \tau$. We consider two main approaches to process the output of the distinguisher. In the first approach [RRCB20], the output is treated as hard information, i.e., the PCO returns a binary decision for the predicate, with a success rate determined by the accuracy of the message distinguisher. In the second approach [HMT25], the output is treated as soft information: the attack makes full use of the probabilistic information (e.g., log-likelihood ratio). Recovering the exact value of a key coefficient $\boldsymbol{sk}_i$ requires multiple PCO queries with different thresholds τ and the way these queries are chosen

and combined determines the attack performances. In the following, we investigate both approaches and evaluate their impact on the (full) key-recovery success rate.

PCO with hard-information. When the PCO treats the output of λ^b_{160D} as hard-information (by returning the result of the test $\lambda^b_{160D} > 0$), it outputs 1 if $\boldsymbol{sk}_i \leq \tau$ and 0 otherwise, with success rate $SR_{\lambda_{160D}}$. Since the secret key coefficients $\boldsymbol{sk}_i$ follow a centered binomial distribution CBD_2, whose range is $\{-2, \ldots, 2\}$, a single coefficient can be recovered using at most $\lceil \log_2 5 \rceil$ PCO queries using a dichotomic search. By exploiting the prior distribution of the secret, an optimal threshold selection strategy can recover a coefficient with 2.3125 queries on average [QCZ+21]. In such a per-coefficient dichotomic search, the full key-recovery process succeeds iff every PCO call succeeds. Hence its success rate for ML-KEM-768 is given by:

$$SR_{\mathrm{KR}} = (SR_{\mathrm{CR}})^{768} = ((SR_{\mathrm{MD}})^{2.3125})^{768} \approx (SR_{\lambda_{160D}})^{1776}, \tag{8}$$

with SR_{CR} the coefficient-recovery success rate and 768 the number of key coefficients.

As shown in Equation 8, the (full) key-recovery performances deteriorate rapidly as the success rate of the message distinguisher degrades. To mitigate this limitation, [RRCB20] propose a majority-voting technique. This approach constructs a stronger distinguisher by repeating the same query v times, giving v independent decapsulation leakage traces, and aggregating the PCO results for each query via majority voting. This improves the overall message-distinguishing success rate at the cost of additional decapsulation executions. In this setting, each query can be modeled as a Bernoulli trial with success probability $SR_{\lambda_{160D}}$. Thus, the majority-vote distinguisher, denoted $\mathrm{MV}(\lambda_{160D}, v)$, follows a binomial distribution $B(v, SR_{\lambda_{160D}})$. Its success rate is given by the probability that a strict majority of the v trials are correct:

$$SR_{\mathrm{MV}(\lambda_{160D},v)} = \Pr\left(\mathrm{MV}(\lambda_{160D}, v) > \left\lfloor \frac{v}{2} \right\rfloor\right) = 1 - I_{1-SR_{\lambda_{160D}}}\left(v - \left\lfloor \frac{v}{2} \right\rfloor, \left\lfloor \frac{v}{2} \right\rfloor + 1\right),$$

where I denotes the regularized incomplete beta function. The corresponding key-recovery success rate for majority-voting therefore becomes $SR_{\mathrm{KR}} = (SR_{\mathrm{MV}(\lambda_{160D},v)})^{1776}$.

Based on these techniques, a natural question arises on how to best exploit D decapsulation traces available for a given ciphertext. Should they be combined to construct a single stronger distinguisher λ_{160D}, or instead used to build a majority-vote distinguisher $\mathrm{MV}(\lambda_{160}, D)$ from single-trace distinguishers? Figure 10a reports the corresponding key-recovery success rates for both approaches, showing that aggregating all decapsulations into a stronger distinguisher λ_{160D} becomes more effective as the masking order increases. This is not surprising: soft aggregation of classifiers is better than hard aggregation in general. A similar observation is reported in [UXT+22] where that soft aggregation requires less PCO queries than majority voting when targeting both unmasked and first-order masked implementations. The advantages of soft aggregation for message distinguishing suggest moving towards a soft treatment of PCO outputs across different messages.

PCO with soft information. The limitations of the hard-PCO approach stem from the reduction of each oracle output to a binary decision, thereby discarding valuable information. A more efficient alternative is to directly exploit the full probabilistic output of the distinguisher, as proposed in [HMT25]. Our results can be viewed as a practical instantiation of the theoretical framework introduced in this work, applied to real masked measurements and tailored to our message distinguishing oracle λ_{160D}. In the single-bit PCO setting, the approach from [HMT25] can be formulated as a sequential Bayesian inference problem. Instead of making binary decisions on the predicate $\boldsymbol{sk}_i \leq \tau$, each oracle query provides probabilistic information that is used to update a posterior distribution

over the possible values of the coefficient $\boldsymbol{sk}_i \in \{-2, \ldots, 2\}$. The overall attack procedure is summarized in Algorithm 4, which we briefly describe in the following lines.

Algorithm 4 SoftCoefRecovery($\rho, \boldsymbol{sk}_i$)

Input: $\rho \in [0,1]$, $\boldsymbol{sk}_i \in \{-2, \ldots, 2\}$
Output: $\hat{SR}_{\mathrm{CR},\boldsymbol{sk}_i} \in \{0,1\}$, $\hat{M}_{\mathrm{CR},\boldsymbol{sk}_i}$

$\pi^{(0)} \leftarrow \mathrm{Prior}_{\mathrm{CBD}_2}$
$t \leftarrow 0$
while $\max_x \pi^{(t)}(x) < \rho$ **do**
  $\tau_t \leftarrow \mathrm{SelectThreshold}(\pi^{(t)})$
  $\lambda_t \leftarrow \mathrm{Oracle}(\tau_t, \boldsymbol{sk}_i)$
  $\pi^{(t+1)} \leftarrow \mathrm{BayesUpdate}(\pi^{(t)}, \lambda_t, \tau_t)$
  $t \leftarrow t + 1$
end while
$\hat{\boldsymbol{sk}}_i \leftarrow \arg\max_x \pi^{(t)}(x)$
return $\hat{\boldsymbol{sk}}_i$, t

Algorithm 5 SelectThreshold(π)

Input: π : probability vector of length 5
Output: $\tau \in [-2, 2]$

$\Delta \leftarrow 1$
$\tau \leftarrow -2$
for $i = -2$ to 2 **do**
  $\mathrm{L} \leftarrow \mathrm{Sum}(\pi[-2, i])$
  $\mathrm{R} \leftarrow \mathrm{Sum}(\pi[i+1, 2])$
  if $|\mathrm{L} - \mathrm{R}| < \Delta$ **then**
    $\Delta \leftarrow |\mathrm{L} - \mathrm{R}|$
    $\tau \leftarrow i$
  end if
end for
return τ

Let $\pi^{(t)}(x) = \Pr[\boldsymbol{sk}_i = x \mid \{\lambda_{160D}^{(j)}\}_{j=1}^{t}]$ denote the secret coefficient posterior distribution after t queries, where $\lambda_{160D}^{(j)}$ are the observed log-likelihood ratios. Starting from the prior induced by the CBD_2 distribution, the threshold τ_t is selected at each iteration to balance the posterior mass on both sides of the split using Algorithm 5. Each query yields a log-likelihood term depending on whether $x \leq \tau_t$ or $x > \tau_t$. The oracle output follows:

$$\lambda_{160D}^{(t)} \sim \begin{cases} \mathcal{N}(\mu_{160D}, \sigma_{160D}^2) & \text{if } \boldsymbol{sk}_i \leq \tau_t, \\ \mathcal{N}(-\mu_{160D}, \sigma_{160D}^2) & \text{otherwise.} \end{cases}$$

The posterior distribution is then updated according to Bayes' rule as:

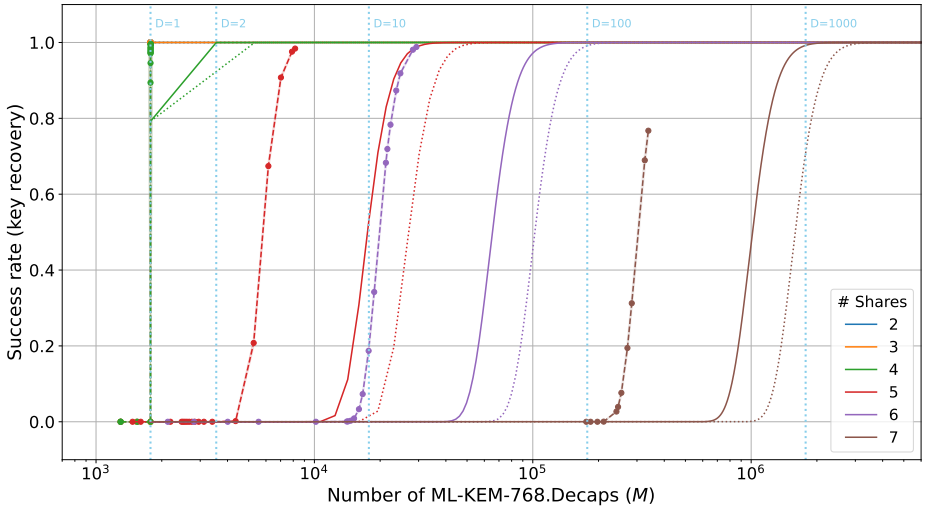
$$\pi^{(t+1)}(x) \propto \pi^{(t)}(x) \cdot 2^{\left(\lambda_{160D}^{(t)} \cdot \mathbb{1}_{x \leq \tau_t}\right)},$$

where the update probability follows directly from the definition of λ_{160D}. The process continues until a confidence condition $\max_x \pi^{(t)}(x) \geq \rho$ is met, at which point the recovered coefficient is given by $\hat{\boldsymbol{sk}}_i = \arg\max_x \pi^{(t)}(x)$. The attack then recovers successively all key coefficients $(\hat{\boldsymbol{sk}}_i)_{i=1,\ldots,768}$, succeeding if $\hat{\boldsymbol{sk}}_i = \boldsymbol{sk}_i$ for all i. Algorithm 4 also returns the number t of oracle calls it makes, and the number of oracle calls for the whole key recovery is simply the sum of t for all coefficients. In this attack, we use the oracle λ_{160}, with $D = 1$: since the attack algorithm uses a Bayesian approach, we do not loose performance while doing so, and we get a finer granularity for the adaptive choice of thresholds.

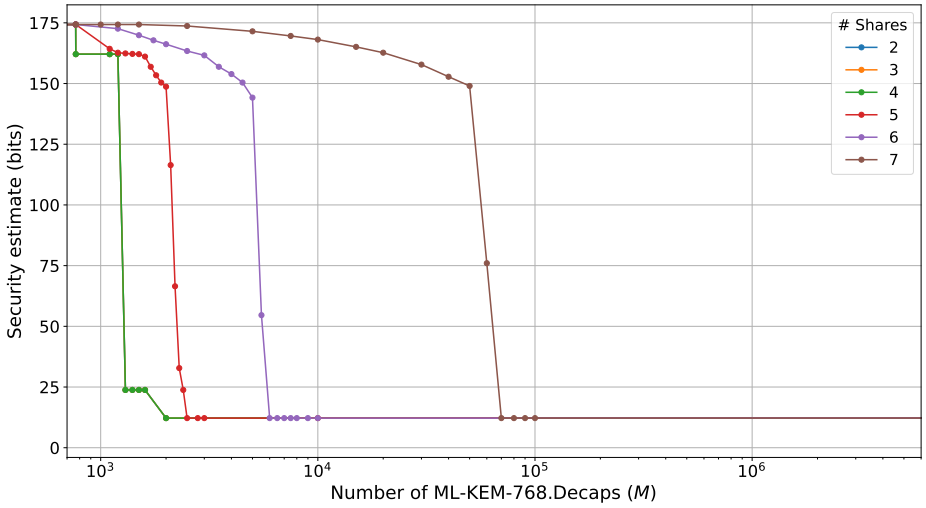
The performance of this attack is reported in Figure 10a (dashed curve). Overall, the adaptive Bayesian attack reduces the number of required decapsulation traces by approximately a factor of five compared to the hard-PCO attack. This improvement stems from two effects. First, removing the tree-based threshold structure yields a better choice of thresholds than a strictly sequential dichotomic process, and combines the results of all oracle calls jointly, reducing the impact of any single "bad" oracle result. Second, the attack dynamically adapts the number of traces used per coefficient depending on obtained information, which allows to often stop earlier than when a fixed number of traces has to be used for all coefficients – an idea that was previously explored in [PRI26].

4.3 Towards more efficient information exploitation

The attacks in the previous section are information-centric: they recover the secret key by accumulating enough information so that each coefficient is recovered in full. This raises



(a) Key-recovery success rate using PCOs on the DOM C implementation for different numbers of shares (curves for 2 and 3 shares are superimposed). Results are given for PCOs with hard-information and message distinguishers λ_{160D} (solid) or $\mathrm{MV}(\lambda_{160}, D)$ (dotted), and for PCOs with soft-information and message distinguisher λ_{160} (dashed).



(b) Security estimates from the best lattice reduction attacks [APS15, DDGR20], combining soft-information PCOs and message distinguisher λ_{160} on the DOM C implementation, for different numbers of shares (curves for 2, 3 and 4 shares are superimposed).

Figure 10: Results of key-recovery attacks on ML-KEM-768.

as a natural question whether trading information for computation can give better results. In the context of symmetric cryptography, the answer is well-known to be positive and key enumeration can be used for this purpose [PSG16]. In the context of post-quantum cryptography, it is expected that such a direct approach should not give the best results, since it does not take advantage of the structure of the (e.g.,) lattice assumption on which (e.g.,) ML-KEM relies. In this section, we therefore turn back to lattice-based cryptanalysis and investigate the extent to which it can reduce the number of ML-KEM decapsulations needed for (full) key recovery at affordable computational cost. As a first step in this direction, we use standard estimators [APS15] and variants with hints thereof [DDGR20]. As will be seen next, they are not perfectly suited to exploit the fine granularity of side-channel leakage. Yet, they can already lead to improved (and likely conservative) estimates that should motivate further research in this direction.

Our approach follows an iterative attack strategy. The 768 posterior distributions over the secret coefficients are initialized according to the prior induced by the CBD_2 distribution. Before each PCO query, the empirical variance of each coefficient is computed from its current posterior distribution, and the coefficient with the largest variance is selected. The soft-information PCO is then queried for this coefficient, and the corresponding posterior distribution is updated following the same threshold selection, oracle response, and Bayesian update rules as in Section 4.2. After each update, the empirical variance across all coefficients is recomputed and the maximum is provided as input to the lattice estimator. This procedure is repeated iteratively until full key recovery.

Since the variance of the secret key distribution is the main parameter of lattice estimators like [APS15, DDGR20], this allows us to leverage them directly in order to obtain security estimates. Yet, it is worth observing that such estimators do not provide the granularity needed to exploit the probabilistic coefficient-based information that side-channel leakage provides. Feeding them with the maximum variance (over the coefficients), combined with the fact that exploiting leakage by reducing the variance implicitly assumes Gaussian distributions, implies that the resulting estimates are likely conservative.

The security estimates obtained with this approach are depicted in Figure 10b, where we only report the best attack results. For completeness, all attack results are provided in Appendix A, Table 6, for the DOM C implementation with 7 shares. The same trends are observed for implementations with fewer shares. The most effective attacks systematically correspond to hybrid attacks [How07], evaluated with the estimator by Albrecht et al. [APS15]. They combine a guessing strategy with lattice reduction, by first guessing a part of the secret key coefficients according to the secret distribution and then applying lattice reduction algorithms to the remaining lattice. As the secret distribution becomes more concentrated, a larger fraction of the key can be guessed, leading to a smaller lattice dimension and more efficient lattice reduction. These attacks significantly improve the number of PCO queries required for key recovery compared to the approach presented in the previous section. We observe that the overall security remains essentially unchanged until all coefficients have been at least updated once (i.e., when $M < 768$). For the implementations with small number of shares, a second drop in security appears around $M \approx 1500$, explained by the need of at least two threshold queries per coefficient to sufficiently reduce their empirical variance to an extent that is reflected by the estimator.

5 Conclusions and open problems

Concretely, the results in this paper substantiate and quantify the worrying threat of "FO-Calypse" that has been reported by several pieces of works over the last years. To the best of our knowledge, they bring the first systematic investigation of concrete attacks taking advantage of the sensitive (masked) Keccak executions taking place in

ML-KEM's decapsulation, in order to instantiate PCOs. We used three open source masked implementations for this purpose and were able to break implementations with up to 7 shares. Given that the attacks we provide are not even optimal, this leads us to conclude that securing ML-KEM in low-cost embedded devices is unlikely to be affordable. Improving them, whether by leveraging transition-based leakages, exploiting more intermediate computations or better dealing with the dependence of different target intermediate variables, is an interesting scope for further research. Taking advantage of secure micro-controllers (e.g., with noise engines or instruction set extensions for leakage-sensitive operations) and hardware co-processors, for which parallelism could be leveraged to reduce the amount of leakage exploitable, is an equally important design challenge.

Besides, our results also allow observing that the lack of systematic and quantitative security evaluations of PCO-based attacks against masked implementations does not come without reason: such evaluations can be extremely time- and storage-consuming. In order to mitigate this difficulty, we also present a simple, concrete and re-usable model for PCOs targeting masked implementations of Keccak. We hope this model can be adopted by designers and evaluation laboratories in order to better assess the scaling trends of masked implementations of ML-KEM. We also hope it can help researchers towards comparing implementations and attack methodologies with convenient (abstract) models that are grounded on realistic assumptions. In this respect, one interesting open problem is to generalize this model to multi-bit PCOs, as for example proposed in [TUX+23, RRD+23], the concrete evaluation of which being even more resource-intensive than single-bit PCOs. This would allow better understanding the contexts in which such generalizations are the most useful, which we expect to be different depending on the accuracy of the PCOs (e.g., in case of low-accuracy PCO, it is possible that the gains of multi-bit variants are in part compensated by the decreased accuracy due to targeting larger secrets [BRM+26].

Eventually, Section 4.3 suggests the specialization of lattice estimators to side-channel leakage and the concrete evaluation of lattice cryptanalysis for low security levels as a promising research direction. Being able to take advantage of any type of probabilistic hint provided at the coefficient level appears natural for this purpose. Better comprehending lattice cryptanalysis with physical hints could also lead to motivate the extraction of other pieces of information from implementation leakage (e.g., on the noise distribution).

Acknowledgments. Gaëtan Cassiers and François-Xavier Standaert are respectively postdoctoral researcher and research director of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in part by the European Union through the ERC projects 101077506 (AMAskZONE) and 101096871 (BRIDGE), and by the Belgian defense through the project 25DEFRA007 (INES). Views and opinions are those of the authors only and do not necessarily reflect those of the EU, the Belgian defense or the ERC. Neither the EU nor the granting authorities can be held responsible for them. Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

A Lattice estimator results

Table 6: Security estimates (in bits) for all lattice-based attacks from [APS15, DDGR20] against the DOM C implementation with 7 shares. Entries marked "∞" indicates security estimates exceeding 1000 bits while "$-$" denotes cases where the lattice estimator failed.

M	Maximal secret std	Albrecht et al. estimator [APS15]								Leaky estimator [DDGR20]
		BKW	uSVP	BDD	BDD hybrid	BDD mitm hybrid	dual	dual hybrid	arora-gb	DBDD
0	1.0	238.2	182.2	183.1	183.1	440.8	189.2	**174.3**	∞	182.5
1000	1.0	238.2	182.2	183.1	183.1	440.8	189.2	**174.3**	∞	182.5
2500	0.98	238.0	181.6	182.6	182.6	439.5	188.6	**173.7**	∞	182.0
5000	0.92	237.9	179.9	180.7	180.7	434.0	186.9	**171.5**	∞	180.4
7500	0.87	230.7	178.4	179.3	179.3	430.7	185.1	**169.6**	∞	179.0
10 000	0.83	227.1	177.2	178.1	178.1	426.7	184.0	**168.1**	∞	177.9
15 000	0.74	226.0	174.3	175.2	175.2	418.9	–	**165.1**	–	175.4
20 000	0.68	214.5	172.0	172.8	172.8	412.1	–	**162.7**	–	173.4
30 000	0.59	203.0	168.2	169.2	169.2	401.1	–	**157.8**	–	170.3
40 000	0.52	195.7	164.7	165.7	165.7	350.8	–	**152.8**	–	167.4
50 000	0.45	187.6	161.2	162.0	157.3	249.2	–	**149.0**	–	164.6
60 000	0.41	179.2	158.6	159.6	**76.0**	95.4	–	146.6	–	162.6
70 000	0.35	167.6	154.2	155.0	**12.2**	12.3	–	142.3	168.3	159.1
80 000	0.28	145.0	148.6	149.6	**12.2**	12.3	–	138.9	135.2	155.1
90 000	0.24	131.9	144.2	145.0	**12.2**	12.3	–	136.4	135.2	151.8
100 000	0.19	109.1	138.4	139.4	**12.2**	12.3	–	132.6	135.2	147.9

References

[ABF+] Melissa Azouaoui, Joppe W. Bos, Bjorn Fay, Marc Gourjon, Yulia Kuzovkova, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Surviving the FO-Calypse: Securing PQC Implementations in Practice. *Real World Crypto Symposium*, 2022.

[ABH+22] Melissa Azouaoui, Olivier Bronchain, Clément Hoffmann, Yulia Kuzovkova, Tobias Schneider, and François-Xavier Standaert. Systematic Study of Decryption and Re-encryption Leakage: The Case of Kyber. In Josep Balasch and Colin O'Flynn, editors, *Constructive Side-Channel Analysis and Secure Design - 13th International Workshop, COSADE 2022, Leuven, Belgium, April 11-12, 2022, Proceedings*, volume 13211 of *Lecture Notes in Computer Science*, pages 236–256. Springer, 2022.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

[BC22] Olivier Bronchain and Gaëtan Cassiers. Bitslicing Arithmetic/Boolean Masking Conversions for Fun and Profit with Application to Lattice-Based KEMs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):553–588, 2022.

[BCS21] Olivier Bronchain, Gaëtan Cassiers, and François-Xavier Standaert. Give Me 5 Minutes: Attacking ASCAD with a Single Side-Channel Trace. *IACR Cryptol. ePrint Arch.*, page 817, 2021.

[BDH+21] Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. Attacking and Defending Masked Polynomial Comparison for Lattice-Based Cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):334–359, 2021.

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 353–367. IEEE, 2018.

[BDK+21] Michiel Van Beirendonck, Jan-Pieter D'Anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. A Side-Channel-Resistant Implementation of SABER. *ACM J. Emerg. Technol. Comput. Syst.*, 17(2):10:1–10:26, 2021.

[BGG+14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the Cost of Lazy Engineering for Masked Software Implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.

[BGR+21] Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal. Masking Kyber: First- and Higher-Order Implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):173–214, 2021.

[BHM+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage Certification Revisited: Bounding Model Errors in Side-Channel Security Evaluations. In Alexandra Boldyreva

and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 713–737. Springer, 2019.

[BPO+20] Florian Bache, Clara Paglialonga, Tobias Oder, Tobias Schneider, and Tim Güneysu. High-Speed Masking for Polynomial Comparison in Lattice-based KEMs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):483–507, 2020.

[BRM+26] Julien Béguinot, Olivier Rioul, Loïc Masure, François-Xavier Standaert, Wei Cheng, and Sylvain Guilley. Scalable Information Theoretic Evaluation of the Rank Statistics in Side-Channel Attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2026:53–81, 2026.

[BS21] Olivier Bronchain and François-Xavier Standaert. Breaking Masked Implementations with Many Shares on 32-bit Software Platforms or When the Security Order Does Not Matter. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):202–234, 2021.

[CB23] Gaëtan Cassiers and Olivier Bronchain. SCALib: A Side-Channel Analysis Library. *J. Open Source Softw.*, 8(86):5196, 2023.

[CGMZ22] Jean-Sébastien Coron, François Gérard, Simon Montoya, and Rina Zeitoun. High-order Table-based Conversion Algorithms and Masking Lattice-based Encryption. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(2):1–40, 2022.

[CGP+12] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of Security Proofs from One Leakage Model to Another: A New Issue. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, Lecture Notes in Computer Science, pages 69–81. Springer, 2012.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

[CS21] Gaëtan Cassiers and François-Xavier Standaert. Provably Secure Hardware Masking in the Transition- and Glitch-Robust Probing Model: Better Safe than Sorry. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):136–158, 2021.

[D'A23] Jan-Pieter D'Anvers. One-Hot Conversion: Towards Faster Table-Based A2B Conversion. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 628–657. Springer, 2023.

[DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with Side Information: Attacks and Concrete Security Estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 329–358. Springer, 2020.

[GD23] John Gaspoz and Siemen Dhooghe. Threshold Implementations in Software: Micro-architectural Leakages in Algorithms. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):155–179, 2023.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How Fast Can Higher-Order Masking Be in Software? In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 567–597, 2017.

[GSM17] Hannes Groß, David Schaffenrath, and Stefan Mangard. Higher-Order Side-Channel Protected Implementations of KECCAK. In Hana Kubátová, Martin Novotný, and Amund Skavhaug, editors, *Euromicro Conference on Digital System Design, DSD 2017, Vienna, Austria, August 30 - Sept. 1, 2017*, pages 205–212. IEEE Computer Society, 2017.

[HHP+21] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):88–113, 2021.

[HMS+23] 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 Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(4):287–317, 2023.

[HMT25] Julius Hermelink, Erik Mårtensson, and Maggie Tran. Noise-Tolerant Plaintext-Checking Oracle Attacks - A Soft-Analytic Approach Applied to ML-KEM. *IACR Cryptol. ePrint Arch.*, page 1496, 2025.

[HNPS24] 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, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 2430–2444. ACM, 2024.

[How07] Nick Howgrave-Graham. A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, Lecture Notes in Computer Science, pages 150–169. Springer, 2007.

[HSMP25] Julius Hermelink, Silvan Streit, Erik Mårtensson, and Richard Petri. A Generic Framework for Side-Channel Attacks Against LWE-Based Cryptosystems. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques Madrid, Spain, May 4-8, 2025,*

Proceedings, Part VIII, volume 15608 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2025.

[JD25] Yanning Ji and Elena Dubrova. A Side-Channel Attack on a Masked Hardware Implementation of CRYSTALS-Kyber. *J. Cryptogr. Eng.*, 15(1):7, 2025.

[KDB+22] Suparna Kundu, Jan-Pieter D'Anvers, Michiel Van Beirendonck, Angshuman Karmakar, and Ingrid Verbauwhede. Higher-Order Masked Saber. In Clemente Galdi and Stanislaw Jarecki, editors, *Security and Cryptography for Networks - 13th International Conference, SCN 2022, Amalfi, Italy, September 12-14, 2022, Proceedings*, volume 13409 of *Lecture Notes in Computer Science*, pages 93–116. Springer, 2022.

[KPP20] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-Trace Attacks on Keccak. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):243–268, 2020.

[LCS+25] 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.

[LPR+14] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. How to Estimate the Success Rate of Higher-Order Side-Channel Attacks. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2014.

[Man04] Stefan Mangard. Hardware Countermeasures against DPA ? A Statistical Analysis of Their Effectiveness. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[MCHS23] Loïc Masure, Gaëtan Cassiers, Julien M. Hendrickx, and François-Xavier Standaert. Information Bounds and Convergence Rates for Side-Channel Security Evaluators. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):522–569, 2023.

[MCLS23] Loïc Masure, Valence Cristiani, Maxime Lecomte, and François-Xavier Standaert. Don't Learn What You Already Know Scheme-Aware Modeling for Profiling Side-Channel Analysis against Masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):32–59, 2023.

[Nat24] National Institute of Standards and Technology. Module-Lattice-Based Key-Encapsulation Mechanism Standard. Federal Information Processing Standards Publication (FIPS) NIST FIPS 203, U.S. Department of Commerce, Gaithersburg, MD, 2024.

[oSN15] National Institute of Standards and Technology (NIST). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Information Processing Standards Publication NIST FIPS 202, National Institute of Standards and Technology, August 2015.

[PP19] Peter Pessl and Robert Primas. More Practical Single-Trace Attacks on the Number Theoretic Transform. In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, volume 11774 of *Lecture Notes in Computer Science*, pages 130–149. Springer, 2019.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 513–533. Springer, 2017.

[PPM+23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. SoK: Deep Learning-based Physical Side-channel Analysis. *ACM Comput. Surv.*, 55(11):227:1–227:35, 2023.

[PRI26] Paco Poilbout, Thomas Roche, and Laurent Imbert. Handling Noisy Plaintext Checking Oracles with SPiRiT - Application to Kyber. In Magali Bardet and Ruben Niederhagen, editors, *Post-Quantum Cryptography - 17th International Workshop, PQCrypto 2026, Saint-Malo, France, April 14-16, 2026, Proceedings, Part II*, volume 16492 of *Lecture Notes in Computer Science*, pages 311–341. Springer, 2026.

[PS26] Duyên Pay and François-Xavier Standaert. Keep it Simple: Refreshing the NTT of Kyber's Decapsulation to Prevent Plaintext-Checking Side-Channel Attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2026:472–499, 2026.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016.

[QCZ+21] 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, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 92–121. Springer, 2021.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):307–335, 2020.

[RRD+23] 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 Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):418–446, 2023.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.

[SCZ+23] 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 Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):89–112, 2023.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[TUX+23] 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 Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):473–503, 2023.

[UXT+22] 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 Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):296–322, 2022.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft Analytical Side-Channel Attacks. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

[vLB13] David A. van Leeuwen and Niko Brümmer. The Distribution of Calibrated Likelihood-Ratios in Speaker Recognition. In Frédéric Bimbot, Christophe Cerisara, Cécile Fougeron, Guillaume Gravier, Lori Lamel, François Pellegrino, and Pascal Perrier, editors, *14th Annual Conference of the International Speech Communication Association, INTERSPEECH 2013, Lyon, France, August 25-29, 2013*, pages 1619–1623. ISCA, 2013.

[WBD24] Ruize Wang, Martin Brisfors, and Elena Dubrova. A Side-Channel Attack on a Higher-Order Masked CRYSTALS-Kyber Implementation. In Christina Pöpper and Lejla Batina, editors, *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part III*, volume 14585 of *Lecture Notes in Computer Science*, pages 301–324. Springer, 2024.

[XPR+22] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David F. Oswald, Wang Yao, and Zhiming Zheng. Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems With Chosen Ciphertexts: The Case Study of Kyber. *IEEE Trans. Computers*, 71(9):2163–2176, 2022.