Prune to Grow: A Pruning-Driven Framework for MV-PC Side-Channel Attacks against Kyber

Jinnuo Li[1,3], Hao Cheng[1,2(✉)], and Chi Cheng[4,5]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
lijinnuo89@gmail.com
[2] State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Qingdao, China
hao.cheng@sdu.edu.cn
[3] School of Computer Science, China University of Geosciences, Wuhan, China
[4] Pei Dingyi Institute of Cryptology, Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China
[5] Faculty of Data Science, City University of Macau, Macao, China
chengchizz@gmail.com

Abstract. Multiple-Valued Plaintext-Checking (MV-PC) side-channel attacks have been proven to be a significant threat to post-quantum key encapsulation mechanisms that are based on the Fujisaki-Okamoto transformation and its variants. However, achieving efficient MV-PC key recovery attacks under a non-ideal MV-PC oracle remains a challenge. This work presents the *recovery-with-pruning* framework, which unifies key recovery and error localization into a single phase. At the core of the framework lies a threshold-enabled oracle that leverages predicted probabilities to assess the reliability of each response, thereby eliminating the need for full post-recovery validation. Compared to the state-of-the-art method, our framework reduces the number of traces for full key recovery against Kyber by at least 36.46%.

Keywords: Side-Channel Attack, Post-Quantum Cryptography, Kyber

1 Introduction

The rapid development of quantum computing poses a serious threat to public-key cryptosystems currently in use, as a large-scale quantum computer would be capable of efficiently breaking RSA and ECC using Shor's algorithm [24]. To address this challenge, NIST initiated the Post-Quantum Cryptography (PQC) standardization process in 2017, aiming to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms that remain secure against quantum-capable adversaries. In August 2024, NIST finalized the first three PQC standards, i.e., FIPS 203 [2], FIPS 204 [1], and FIPS 205 [3]. Among these, FIPS 203 (ML-KEM, based on Kyber [8]) defines the standardized key encapsulation mechanism.

Side-Channel Attacks (SCAs), first introduced by Kocher [14], have since become a crucial component in evaluating the practical security of cryptographic implementations. NIST has also emphasized the importance of side-channel analysis and corresponding countermeasures throughout the PQC standardization process, expressing its expectation that post-quantum schemes will provide protection against timing, power, and fault attacks while maintaining efficient performance [4]. Consequently, evaluating the side-channel resilience of Kyber is essential to ensure the practical security of post-quantum schemes. Research on SCAs targeting lattice-based KEMs has become a well-established area, as evidenced by numerous recent publications [13, 6, 19, 7, 5, 15, 22]. These attacks have covered all parts of ML-KEM, employing a variety of side-channel techniques such as timing, power, electromagnetic, and fault injection.

In particular, SCAs against the decapsulation algorithm of KEMs have attracted considerable attention and are typically classified into three main categories [19]: 1) Plaintext-Checking (PC) oracle attacks [20, 26, 23], and their variants Multiple-Valued (MV-) PC oracle attacks [25, 18, 17]; 2) Decryption-Failure (DF) oracle attacks [15]; and 3) Full-Decryption (FD) oracle attacks [28, 10, 9]. For the PC oracle, the response indicates whether the decrypted ciphertext matches a specific given message. The MV-PC oracle extends this concept by allowing the extraction of multiple bits from a single query via multi-class classification, leveraging recent advances in Deep Learning (DL) techniques for SCAs [27]. In comparison, an FD oracle exploits the leakage from the message encoding/decoding function. Since this process consists of linear operations where each bit independently influences the leakage, the adversary can recover the entire decrypted message efficiently. Among these, the MV-PC oracle-based attack deserves special attention for its generic feature and the significant challenges it presents in deploying countermeasures, a characteristic it shares with the binary PC oracle-based attack. Although FD oracle attacks generally require fewer traces and are therefore more efficient, the operations they exploit are relatively fast compared to the entire re-encryption process targeted by PC and DF oracle attacks. As a result, countermeasures designed to defend against PC oracle-based attacks can incur a substantial overhead. In this work, we primarily focus on MV-PC oracle attacks.

1.1 Related work

Although MV-PC oracle attacks are relatively well studied from both implementation and attack-workflow perspectives, their practical deployment still faces challenges arising from environmental noise and classifier inaccuracies. Existing practical approaches to mitigating such noise in (MV-)PC oracle attacks can be broadly divided into two types. The first consists of traditional trace-based methods that enhance oracle reliability through redundancy, such as majority-voting schemes and statistical post-processing using negative-log-likelihood analysis. The second type follows the *recovery-then-check* framework proposed by Shen et al. [23], in which they perform an efficient key recovery attack under a non-ideal side-channel PC oracle, employing a "fast-checking" method that leverages

prior knowledge of approximately correct results to detect errors. However, their method is specifically designed for PC oracles and does not apply to the MV-PC setting. Furthermore, it is only applicable to Kyber-512 and cannot be used with Kyber-768 or Kyber-1024.

To address these limitations, Li et al. [17] subsequently extended the framework to the MV-PC oracle using a technique known as the "grafted tree". Their approach is more generic and can be applied not only to the NIST-standardized KEM, Kyber, but also to other candidates such as Saber and FrodoKEM. Compared to traditional approaches, the recovery-then-check framework based on the grafted tree offers higher efficiency, reducing the overall cost by more than 42.5%. However, it still incurs additional cost of over 65% compared with the ideal oracle, particularly in the checking phase.

1.2 Contributions

To further reduce the cost, we develop new strategies for more efficient MV-PC oracle key recovery under noisy conditions, and our contributions are summarized as follows:

- We design a recovery-with-pruning framework that integrates selective pruning into the recovery phase to perform key reconstruction and error localization simultaneously. By identifying and re-querying only unreliable oracle responses, the framework eliminates full post-recovery validation, thereby improving efficiency without compromising accuracy.
- To enable this framework, we introduce a threshold-enabled oracle, which effectively enhances accuracy while maintaining high efficiency. Compared with the ideal oracle, the additional cost is only about 5%. Furthermore, we conduct an analysis across multiple factors to evaluate threshold selection strategies under various noise conditions.
- We provide a comprehensive comparison of experimental results on the MV-PC oracle attack against all security levels of Kyber. The significant reduction in the number of traces is achieved by our attack, i.e., over 36.46% compared to the state of the art, thereby greatly lowering the effort required for practical key recovery.

1.3 Organization

The rest of this paper is organized as follows. Section 2 provides the background on the Kyber KEM scheme and the MV-PC oracle, as well as the experimental setup. The design of our new framework and the evaluation of our attacks are presented in Section 3 and Section 4, respectively. Finally, we conclude our paper in Section 5.

2 Background

2.1 Kyber

Kyber (now ML-KEM [2]) operates over the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$, where $q = 3329$ is a prime and $n = 256$. The polynomial operations, including addition and multiplication, are thus performed modulo $X^n + 1$. In an element of the ring, namely a polynomial $f(X) = f_0 + f_1X + \cdots + f_{n-1}X^{n-1} \in \mathcal{R}_q$, each coefficient f_j where $0 \leq j \leq n-1$ belongs to $\mathbb{Z}_q$ (i.e., the ring of integers modulo q). In this paper, we use a bold lowercase letter $\mathbf{a} \in \mathcal{R}_q$ and its vector representation $(\mathbf{a}[0], \ldots, \mathbf{a}[n-1])$ interchangeably to denote a polynomial.

The security of Kyber relies on the computational difficulty of solving certain systems of noisy linear equations, specifically the Module Learning With Errors (MLWE) problem [16]. In linear algebra, it is straightforward to solve for $\mathbf{s}$ in $\mathbf{b} = \mathbf{As}$, where $\mathbf{A}$ is a matrix while $\mathbf{b}$ and $\mathbf{s}$ are vectors. However, according to the Learning With Errors (LWE) problem [21], adding small random noise to this relation, namely $\mathbf{b} = \mathbf{As} + \mathbf{e}$, makes recovering $\mathbf{s}$ computationally hard. The MLWE problem then generalizes this to a module structure: it asks one to distinguish $(\mathbf{A}, \mathbf{B} = \mathbf{As} + \mathbf{e}) \in \mathcal{R}_q^{l \times l} \times \mathcal{R}_q^l$ from uniformly random $(\mathbf{A}, \mathbf{B}) \in \mathcal{R}_q^{l \times l} \times \mathcal{R}_q^l$, and the parameter l determines the security level, with $l = 2$, 3, and 4 corresponding to Kyber-512, Kyber-768, and Kyber-1024, respectively.

In Kyber, all secret key coefficients and error vectors are sampled from a Centered Binomial Distribution (CBD) $\mathcal{B}\eta$, defined as $\sum_{i=1}^{\eta}(a_i - b_i)$, where a_i and b_i are independent and uniformly random samples from $\{0, 1\}$. The parameter η varies with the security levels, and we provide the information of η as well as other main parameters in Table 1.

Let $\lfloor x \rfloor$ denote the maximum integer not exceeding x, and $\lceil x \rfloor$ is the rounding function, i.e., $\lceil x \rfloor = \lfloor x + \frac{1}{2} \rfloor$. In the following, we first give the definitions of two functions: $\mathbf{Comp}_q(x, d)$ and $\mathbf{DeComp}_q(x, d)$.

Definition 1. *The **Comp** function:* $\mathbb{Z}_q \to \mathbb{Z}_{2^d}$*:*

$$\mathbf{Comp}_q(x, d) = \lceil (2^d/q) \cdot x \rfloor \pmod{2^d}.$$

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

$$\mathbf{DeComp}_q(x, d) = \lceil (q/2^d) \cdot x \rfloor .$$

In $\mathbf{Comp}_q(x, d)$ and $\mathbf{DeComp}_q(x, d)$, the input x is selected from $\mathbb{Z}_q$. Likewise, when the input is a polynomial, the above operation will be performed separately for each coefficient. Similarly, in $\mathbf{Comp}_q(x, d)$ and $\mathbf{DeComp}_q(x, d)$, the value of d is set as $d_{\mathbf{u}}$ or d_v for the different security levels of Kyber.

Table 1: Parameter sets of Kyber.

Scheme	NIST Security	n	l	q	η	$d_{\mathbf{u}}$	d_v
Kyber-512	Category 1	256	2	3329	3	10	4
Kyber-768	Category 3	256	3	3329	2	10	4
Kyber-1024	Category 5	256	4	3329	2	11	5

Algorithm 1 Kyber KEM scheme.

KYBER.CPA.Gen
Ensure: sk, pk
1: $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{l\times l}$
2: $\mathbf{s}, \mathbf{e} \xleftarrow{\$} \mathcal{B}_{\eta_1}^{l}$
3: $sk = \mathbf{s}$
4: $pk := \mathbf{A}^T\mathbf{s} + \mathbf{e}$
5: $sk' := (sk||pk||\mathcal{H}(pk)||z)$

KYBER.CPA.Enc
Require: $pk, \mathbf{m}, r$
Ensure: ct
1: Generate $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{l\times l}$ from pk
2: $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2 \xleftarrow{\$} \mathcal{B}_{\eta_2}^{l}$
3: $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$
4: $\mathbf{v} = \mathbf{b}^T\mathbf{r} + \mathbf{e}_2 + \mathrm{DeComp}_q(\mathbf{m}, 1)$
5: $\mathbf{c}_1 := \mathrm{Comp}_q(\mathbf{u}, d_u)$
6: $\mathbf{c}_2 := \mathrm{Comp}_q(\mathbf{v}, d_v)$
7: $ct := (\mathbf{c}_1, \mathbf{c}_2)$

KYBER.CPA.Dec
Require: sk, ct
Ensure: $\mathbf{m}'$
1: $\mathbf{u}' = \mathrm{DeComp}_q(\mathbf{c}_1, d_u)$
2: $\mathbf{v}' = \mathrm{DeComp}_q(\mathbf{c}_2, d_v)$
3: $\mathbf{m}' = \mathrm{Comp}_q(\mathbf{v}' - \mathbf{s}^T\mathbf{u}', 1)$

KYBER.KEM.KeyGen
Ensure: sk', pk
4: Generate a pseudo-random coin z
5: $sk, pk :=$ **KYBER.CPA.Gen**()
6: $sk' := (sk||pk||\mathcal{H}(pk)||z)$

KYBER.CCA.Encaps
Require: pk
Ensure: ct, K
1: $\mathbf{m} \xleftarrow{\$} \{0,1\}^{256}$
2: $(\bar{K}, r) = \mathcal{G}(\mathbf{m}, \mathcal{H}(pk))$
3: $ct :=$ **KYBER.CPA.Enc**$(pk, \mathbf{m}, r)$
4: $K := \mathrm{KDF}(\bar{K}, \mathcal{H}(ct))$

KYBER.CCA.Decaps
Require: sk', ct
Ensure: K
1: $\mathbf{m}' :=$ **KYBER.CPA.Dec**(sk, ct)
2: $(\bar{K}', r') = \mathcal{G}(\mathbf{m}', \mathcal{H}(pk))$
3: $ct' =$ **KYBER.CPA.Enc**$(pk, \mathbf{m}', r')$
4: **if** $ct = ct'$ **then**
5: $\quad K := \mathrm{KDF}(\bar{K}', \mathcal{H}(ct))$
6: **else**
7: $\quad K := \mathrm{KDF}(z, \mathcal{H}(ct))$
8: **end if**

A KEM scheme typically consists of three algorithms: key generation, encapsulation, and decapsulation. The CCA-secure KEM of Kyber is derived from a simpler CPA-secure Public-Key Encryption (PKE) scheme (comprising the modules **KYBER.CPA.Gen**, **KYBER.CPA.Enc**, and **KYBER.CPA.Dec**) via the Fujisaki-Okamoto (FO) transformation [11]. The details of the Kyber KEM scheme are described in Algorithm 1.

2.2 Multiple-Valued Plaintext-Checking (MV-PC) oracle

Table 2: The value of $\mathbf{m}'[j]$ corresponding to different k_j and $\mathbf{s}_0[j]$.

$\mathbf{s}_0[j] =$	-2	-1	0	1	2
$k_j = 7$	1	0	0	0	0
$k_j = 8$	1	1	0	0	0
$k_j = 9$	1	1	1	0	0
$k_j = 10$	1	1	1	1	0

The concept of the MV-PC oracle was independently proposed by Rajendran et al. [18] and Tanaka et al. [25], enabling an efficient SCA on post-quantum KEMs. The core idea of it is to construct special ciphertexts such that the first N bits of $\mathbf{m}'$ depend on the N corresponding coefficients of the secret key $\mathbf{s}$. In the following, we take Kyber-1024 as an example to introduce the MV-PC oracle-based attack. Assume the secret key of the victim is $\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$. Specifically, targeting the first coefficient block, namely the N coefficients from $\mathbf{s}_0[0]$ to $\mathbf{s}_0[N-1]$, the adversary selects

$$\begin{aligned}\mathbf{u} &= (\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3) = ((\lceil \frac{q}{32} \rfloor, 0, \cdots, 0), \mathbf{0}, \mathbf{0}, \mathbf{0}) \\ \mathbf{c}_2 &= (k_0, k_1, \cdots, k_{N-1}, 0, \cdots, 0),\end{aligned}$$

computes $\mathbf{c}_1 = \mathbf{Comp}_q(\mathbf{u}, d_u)$ and sends ciphertext $ct := (\mathbf{c}_1, \mathbf{c}_2)$ to the victim. With the received ct, the victim decompresses it and then gets $\mathbf{u}'$ and $\mathbf{v}'$. Later, the victim computes the $\mathbf{m}'$ as follows:

$$\begin{aligned}\mathbf{m}'[j] &= \mathbf{Comp}_q((\mathbf{v}' - \mathbf{s}^T\mathbf{u}')[j], 1) \\ &= \mathbf{Comp}_q((\mathbf{v}' - \mathbf{s}_0\mathbf{u}'_0 - \mathbf{s}_1\mathbf{u}'_1 - \mathbf{s}_2\mathbf{u}'_2 - \mathbf{s}_3\mathbf{u}'_3)[j], 1) \\ &= \left\lceil \frac{2}{q}(\mathbf{v}'[j] - \mathbf{s}_0[j]\mathbf{u}'_0[0]) \right\rfloor \bmod 2 \\ &= \begin{cases} \left\lceil \frac{2}{q}\left(\left\lceil \frac{q}{32} \right\rfloor k_i - \mathbf{s}_0[j]\left\lceil \frac{q}{32} \right\rfloor\right) \right\rfloor \bmod 2, & j < N, \\ \left\lceil \frac{2}{q}\left(0 - \mathbf{s}_0[j]\left\lceil \frac{q}{32} \right\rfloor\right) \right\rfloor \bmod 2 = 0, & j \geq N. \end{cases}\end{aligned}$$

The equation above reveals that a specific bit of the message $\mathbf{m}'[j]$ relates to an *adversary-controlled* ciphertext coefficient k_j and a private-key coefficient $\mathbf{s}_0[j]$. Therefore, this relationship enables recovery of the values of the private-key coefficients, and the detail is indicated in Table 2. Recall that in Kyber-1024, coefficients $\mathbf{s}_i[j]$ are chosen from the interval $[-2, 2]$, therefore the maximal value

Algorithm 2 MV-PC oracle and MV-PC attack.

SCA-based MV-PC oracle $\mathcal{O}_{mv}$
Require: Ciphertext ct, Model $\mathcal{M}$
Ensure: $\mathbf{m}'$
$\mathcal{T} \leftarrow \text{SCA}(\text{Decaps}(sk, ct))$ ▷ $\mathcal{T}$: single side-channel trace
$\mathbf{m}' = \mathcal{M}.Classify(\mathcal{T})$
Return $\mathbf{m}'$

SCA-based MV-PC attack
Require: MV-PC oracle $\mathcal{O}_{mv}$
Ensure: Secret key $\mathbf{s}$
for $i = 0 \rightarrow l - 1$ **do**
  for $j = 0 \rightarrow (256/N) - 1$ **do**
    Set response sequence $seq \leftarrow$ “”
    while block cannot be recovered **do**
      Generate ct according to seq
      Append $\mathcal{O}_{mv}(ct, \mathbf{m})$ to seq
    end while
    Get the values of the block $\mathbf{s}_i[8j : 8j + 7]$
  end for
  Get the values of coefficients in $\mathbf{s}_i$
end for
Return Recovered $\mathbf{s}$

of $\mathbf{s}_i[j]$ is 2, which means

$$\frac{2}{q}\left(\mathbf{s}_0[j]\left\lceil\frac{q}{32}\right\rfloor\right) \approx 0.125 < 1/2.$$

That is, for $j \geq N$, $\mathbf{m}'[j] = 0$. For attacking other coefficient blocks of $\mathbf{s}_0$ such as $\mathbf{s}_0[x : x + 7]$, the attacker just constructs $\mathbf{u}_0$ as $\mathbf{u}_0[256 - x] = -\lceil\frac{q}{32}\rfloor$ with all other elements being 0. In addition, as for attacking the coefficients in $\mathbf{s}_1$ (resp. $\mathbf{s}_2$, or $\mathbf{s}_3$), the attacker simply utilizes $\mathbf{u}_1$ (resp. $\mathbf{u}_2$ or $\mathbf{u}_3$), instead of $\mathbf{u}_0$.

The purpose of this setting is to ensure that $\mathbf{m}'$ has only 2^N possible cases, which is directly related to the complexity of the SCA. Hereafter, we set $N = 8$ for the ease of analysis and testing. We note that, when $N > 8$, the attack remains effective, but the side-channel complexity increases accordingly.

During an attack, by selecting k_j and making multiple queries, the adversary is able to progressively narrow down the possible values for $\mathbf{s}_i[j]$, eventually obtaining its precise value as shown in Table 3.

Table 3: Selections of k_j and the corresponding states for Kyber-1024.

State	**State 1**	**State 2**	**State 3**	**State 4**
k_j	8	9	10	7
$\mathbf{m}'[j] = 0$	State 2	State 3	$\mathbf{s}_i[j] = 2$	$\mathbf{s}_i[j] = -1$
$\mathbf{m}'[j] = 1$	State 4	$\mathbf{s}_i[j] = 0$	$\mathbf{s}_i[j] = 1$	$\mathbf{s}_i[j] = -2$

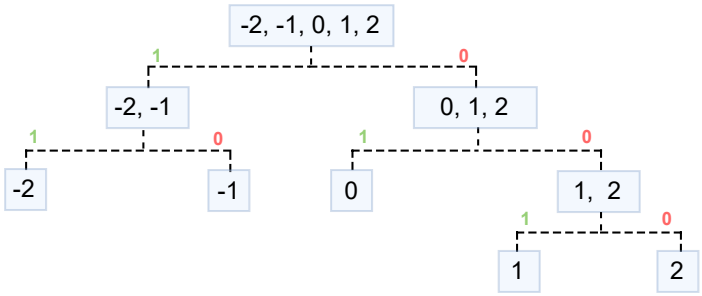

Fig. 1: BRT for MV-PC oracle against Kyber-1024.

Within an iteration, $\mathbf{u}$ remains fixed, and the value of k_j is adjusted based on $\mathbf{m}'[j]$. Algorithm 2 summarizes the entire attack process of the MV-PC oracle: the adversary repeatedly queries the oracle and obtains the values of $\mathbf{m}'[j]$ as responses; the resulting response sequence is then used to recover the coefficient values $\mathbf{s}_i[j]$.

In a real-world attack, the adversary exploits side-channel leakage from the entire re-encryption process[6] to train a classifier. For each attack instance, the adversary uses the classifier to infer the $\mathbf{m}'$, which is generated during the computation of **KYBER.CCA.Decaps**.

Another key component of the attack is the Binary Recovery Tree (BRT). As illustrated in Figure 1, the BRT provides a visual representation of the adaptive selection of ciphertexts based on oracle responses. It is organized as a rooted binary tree with a single root node and multiple leaf nodes, where each leaf node corresponds to a coefficient of the secret key. For each internal node with children, the left child is labeled as `1` and the right child as `0`, corresponding respectively to the oracle responses of 1 and 0. When applied to the MV-PC oracle, an adversary only needs to select ciphertexts and get response sequences for each bit using the BRT.

2.3 Experimental setup

To permit practical evaluation, we use the ChipWhisperer CW312 SAM4S board as our target board, which hosts an ARM Cortex-M4 core, and is plugged into a ChipWhisperer CW313 board. We connected the ChipWhisperer-Husky to the CW313 in order to measure the power consumption of the target, at a sampling rate of 4 samples per cycle. We use the software implementation of Kyber contained in the `pqm4` framework [12], which is then compiled using `arm-none-eabi-gcc 10.2.1` with the optimization level `-O3`. In addition, we use the same Neural Network (NN) model architecture presented in [25, 17], with the details summarized in Table 4.

[6] A concrete example is lines 147 to 150 at `https://github.com/mupq/pqm4/blob/master/crypto_kem/ml-kem-768/m4fspeed/kem.c` in `pqm4` [12].

Table 4: The hyperparameters of the NN for the classifier used in the experiments.

	Operator	Activation function	BN	Pooling	Stride
Conv1	conv1d (3)	SELU	Yes	Avg (2)	2
Conv2	conv1d (3)	SELU	Yes	Avg (2)	2
Conv3	conv1d (3)	SELU	Yes	Avg (2)	2
Conv4	conv1d (3)	SELU	Yes	Avg (2)	2
Conv5	conv1d (3)	SELU	Yes	Avg (2)	2
Conv6	conv1d (3)	SELU	Yes	Avg (2)	2
FLT	flatten	-	-	-	-
FC1	dense	SELU	No	No	-
FC2	dense	SELU	No	No	-
FC3	dense	Softmax	No	No	-

3 Design

3.1 Challenge in previous work

As mentioned in Section 1, the methods of previous work are based on a common framework, namely *recovery-then-check*, whose workflow is as follows: 1) The key recovery yields approximate secret coefficients; 2) Check the results, then flag inconsistent coefficient blocks; 3) Refine suspicious blocks via additional recovery.

A common premise of these methods, which operate under the recovery-then-check framework, stems from the challenge identified in [23]: determining the positions of errors is computationally intractable, resulting in prohibitively high brute-force complexity. Consequently, during the checking phase, *all candidates* produced in the recovery phase must be validated. Although techniques such as the fast-checking [23] or the grafted tree [17] can accelerate this process, they still introduce substantial overhead.

3.2 Our recovery-with-pruning framework

To address this challenge, we propose a novel *recovery-with-pruning* framework, which identifies unreliable coefficient positions *during the recovery phase itself*. By integrating a selective pruning mechanism, the framework combines key recovery and error localization, eliminating full post-recovery validation (i.e., the checking process) and thus reducing overhead by focusing on the most promising candidates. In more detail, during the recovery process, after each oracle query, the system assesses the reliability of the obtained response. If the result is identified as potentially erroneous, it is pruned, i.e., discarded instead of being accepted into the recovery sequence, and the oracle is re-queried until a response with sufficient confidence is achieved. The reliable result is then adopted for updating the recovery sequence.

Figure 2 illustrates a representative example executed under our framework. When recovering a private key coefficient with a true value of 2, an error occurs

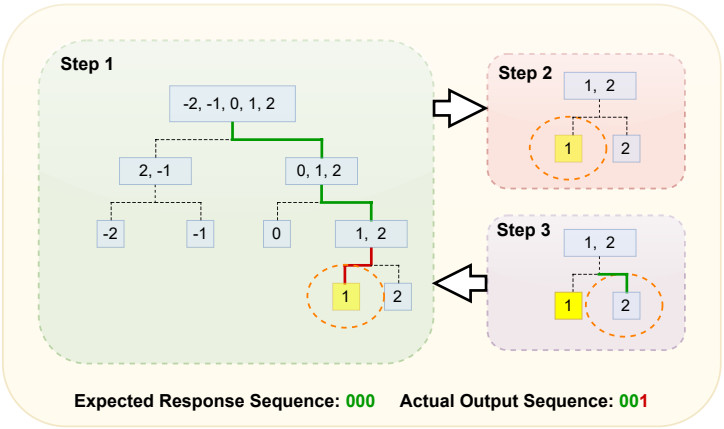

Fig. 2: An example executed under our recovery-with-pruning framework.

Algorithm 3 Threshold-enabled MV-PC oracle $\mathcal{O}_T$.

Require: Ciphertext ct, model $\mathcal{M}$, and threshold t.
Ensure: $\mathbf{m}'$.

$\mathcal{T} \leftarrow \text{SCA}(\text{Decaps}(sk, ct))$ ▷ $\mathcal{T}$: single side-channel trace
$(\mathbf{m}', \sigma) = \mathcal{M}.Classify(\mathcal{T})$ ▷ σ: predicted probability
while $\sigma < t$ **do**
    $\mathcal{T} \leftarrow \text{SCA}(\text{Decaps}(sk, ct))$
    $(\mathbf{m}', \sigma) = \mathcal{M}.Classify(\mathcal{T})$
end while
Return $\mathbf{m}'$

in the third query, where the oracle response flips from `0` to `1`, producing an incorrect result of `1` (Step 1). The system recognizes this query as unreliable (Step 2) and performs a re-query, which later returns a correct response. Consequently, the sequence is corrected from `001` to `000`, and the accurate value `2` is finally recovered (Step 3). Note that the first two queries were accepted directly without additional validation, while only the erroneous one is corrected through re-querying, thereby achieving higher efficiency without compromising accuracy.

3.3 Threshold-enabled oracle

At the core of our framework, we introduce a *threshold-enabled oracle*, described in Algorithm 3. Instead of relying on conventional hard-decision outputs, the oracle leverages probabilistic estimation scores from a DL classifier. By setting a confidence threshold on the prediction probabilities of the classifier, the oracle detects query results with a high likelihood of error and triggers re-querying until the response meets the confidence criterion. This mechanism enables adaptive refinement during the recovery process, effectively filtering out potentially

erroneous candidates and thus achieving higher accuracy with fewer redundant checks.

In the following, we provide the formal theoretical analysis of the proposed oracle. Set the original accuracy as α, the original cost as κ, the threshold as t, and the maximum reselection count as r. Let c_t be the probability that a correctly predicted result has a prediction probability higher than the threshold t, and let e_t be the probability that an incorrectly predicted result has a prediction probability higher than the threshold t. We can obtain the enhanced accuracy

$$\alpha' = 1 - (1-\alpha)e_t \tag{1}$$

and the updated cost

$$\begin{aligned}\kappa' \leq\ & \kappa + \kappa(\alpha(1-c_t) + (1-\alpha)(1-e_t)) \\ & + \kappa(\alpha(1-c_t) + (1-\alpha)(1-e_t))^2 \\ & + \cdots \\ & + \kappa(\alpha(1-c_t) + (1-\alpha)(1-e_t))^{r-1} \\ \leq\ & \kappa\left(\frac{1-[\alpha(1-c_t) + (1-\alpha)(1-e_t)]^r}{1-[\alpha(1-c_t) + (1-\alpha)(1-e_t)]}\right).\end{aligned}$$

The cost is increased due to the extra reselection attempts. Denoting the probability that a result triggers reselection as

$$p = \alpha(1-c_t) + (1-\alpha)(1-e_t),$$

the total cost after at most r reselections is bounded by

$$\kappa' \leq \kappa\left(\frac{1-p^r}{1-p}\right).$$

3.4 Choice of threshold

We then analyze how to choose an optimal threshold value, for which we conducted a series of practical experiments. The results are shown in Figure 3, which illustrates that the choice of threshold has a systemic impact on many metrics. For example, lowering the threshold increases both the correct acceptance probability c_t and the incorrect acceptance probability e_t (see (a)). The increase in c_t reduces the likelihood of re-querying, thereby decreasing the additional cost (see (c)). However, the simultaneous increase in e_t diminishes the enhanced accuracy α' defined in Equation 1 (see (b)), which in turn leads to a higher number of errors (see (d)). In our experiments, we set the threshold to 76.5%, as indicated by the black dashed line in Figure 3. We stress that an optimal threshold is supposed to balance cost reduction and accuracy degradation, which depends on the tolerance for errors of the system. To be more specific, under ideal or low-noise conditions a lower threshold is preferred for reducing cost, whereas in noisy environments, a higher threshold can be employed to preserve accuracy. In subsequent experiments under more adverse conditions, we accordingly increased the threshold, and the detailed analysis is provided in the next section.

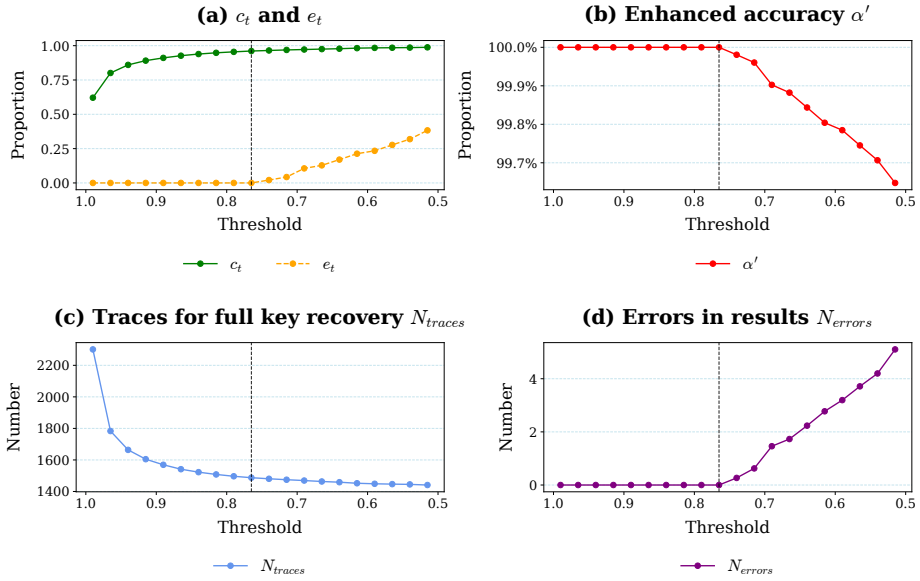

Fig. 3: Experimental results under different threshold conditions.

4 Evaluation

In this section, we present the results of evaluating our framework through empirical testing. We compare our recovery-with-pruning framework with the ideal oracle (i.e., the accuracy of the oracle is 100%) and the state-of-the-art recovery-then-check framework (i.e., the method of Li et al. [17]), using the average number of traces required for full key recovery as the evaluation metric, and in both cases the average number of erroneous coefficients in the final recovered key is ensured to be below 1.0. The results are shown in Table 5.

Notably, our initially trained models achieved accuracy levels that were slightly higher than those assumed in the simulated settings of [17]. To ensure a fair comparison with their simulated settings, we deliberately injected a small amount of noise into the dataset so that the final accuracies were adjusted to be slightly lower than those in [17]. To be more specific, the high, medium, and low accuracies reported in [17] are respectively 99.90%, 95.00%, and 90.00%, while in our practical experiments they are respectively 99.08%, 94.06%, and 89.30%. This discrepancy stems from the difficulty of controlling the accuracy of the model to a precise value in a practical experiment. Therefore, we adopted the slightly lower accuracy levels. Although this may affect the reported performance, our framework still demonstrates superior results, and we stress that such differences in accuracy are non-negligible in attacks.

Table 5: Comparison of the number of required power traces for full key recovery ($N = 8$). Our attacks were conducted 10 thousand times with randomly selected secret keys.

(a) Comparison with Li et al. [17]

Scheme	Li et al. [17]		This work		
	accuracy	#traces	accuracy	threshold	#traces
Kyber-512	99.90%	321.2	99.08%	76.50%	200.1 (-37.70%)
Kyber-768	99.90%	476.7	99.08%	76.50%	301.4 (-36.77%)
Kyber-1024	99.90%	635.5	99.08%	76.50%	403.8 (-36.46%)
	95.00%	1336.8	94.06%	91.50%	698.1 (-47.78%)
	90.00%	2041.6	89.30%	94.00%	1278.7 (-37.37%)

(b) Comparison with ideal oracle

Scheme	Ideal oracle	Li et al. [17]		This work	
	#traces	accuracy	#traces	accuracy	#traces
Kyber-512	192	99.90%	321.2 (+67.29%)	99.08%	200.1 (+4.21%)
Kyber-768	288	99.90%	476.7 (+65.52%)	99.08%	301.4 (+4.65%)
Kyber-1024	384	99.90%	635.5 (+65.49%)	99.08%	403.8 (+5.16%)
		95.00%	1336.8 (+248.12%)	94.06%	698.1 (+81.80%)
		90.00%	2041.6 (+431.67%)	89.30%	1278.7 (+232.99%)

4.1 Comparison in high accuracy

Firstly, we examine the experimental results obtained under the high-accuracy setting (i.e., the accuracy of the oracle is 99.90/99.08% and the threshold is set to 76.50%), which correspond to the first three rows of Table 5.

In particular, on average, our framework only requires 200.1, 301.4, and 403.8 traces for full key recovery against Kyber-512, Kyber-768, and Kyber-1024, respectively. Compared with the method of Li et al., our framework yields a more than 36.46% reduction in the number of required traces. Meanwhile, it incurs only 4.21% to 5.16% additional overhead compared with the ideal oracle, which is substantially lower than the over 65% reported in the work of Li et al.

4.2 Extension to lower accuracy

In contrast to previous work, i.e., [23] and [17], that evaluates the low-accuracy conditions through simulation, we define a concrete adversarial scenario:

- The attacker is assumed to have limited SCA expertise and employs raw traces without any pre-processing.
- The model is a GPT-assisted baseline that is used without parameter tuning or task-specific optimization.

Due to the constraints above, the overall accuracy is reduced; however, this practical setup allows us to examine the robustness of our approach under realistic and suboptimal conditions.

To this end, the following experiments further evaluate the performance of our method under these circumstances and compare it with the existing approach. The corresponding results are summarized in the last two rows of Table 5, alongside the previous results, for a unified comparison.

The experimental results demonstrate that the proposed framework still maintains a significant advantage over the baseline [17], even under degraded measurement accuracy. In particular, the proposed framework reduces the number of traces required for complete key recovery against Kyber-1024 by 47.78% at the medium-accuracy level (i.e., the accuracy of the oracle is 95.00/94.06% and the threshold is set to 91.50%) and by 37.37% at the low-accuracy level (i.e., the accuracy of the oracle is 90.00/89.30% and the threshold is set to 94.00%).

As the accuracy decreases, the relative advantage of our proposed framework becomes less pronounced, compared to the method of Li et al. It is worth noting that, under low-accuracy conditions, Li et al. typically combine their method with additional optimization techniques such as majority voting, whereas our results rely solely on the core algorithm described in the previous sections. This behavior is consistent with expectations.

Overall, the analysis demonstrates that our proposed framework consistently achieves lower additional cost compared to prior approaches. The most significant improvement is observed under high-accuracy conditions, where our method operates with minimal extra cost while maintaining stable performance as the accuracy decreases. These results highlight the practical efficiency and robustness of the proposed framework under various noise levels.

5 Conclusion

This paper presents a novel recovery-with-pruning framework for MV-PC side-channel attacks against Kyber. Employing a threshold-enabled oracle as its kernel, the framework eliminates the need for full post-recovery validation, and our experimental results show that the framework significantly reduces the number of side-channel traces required for the key-recovery attack. Furthermore, by directly optimizing the oracle, the framework exhibits strong generality and flexibility and can be combined with existing optimization techniques (e.g., [25, 23, 17]). Importantly, our framework is broadly applicable across different KEM schemes, overcoming the limitation of attacks that depend on specific schemes or parameters.

Beyond these technical advancements, the primary implication of our work lies in its direct bearing on the practical security of post-quantum cryptosystems. Specifically, by showing that efficient key recovery is feasible under realistic, noisy conditions with far fewer side-channel traces than previously thought, our results reveal how low the cost of a practical attack on unprotected implementations can be. In turn, as Kyber moves toward practical deployment, this evidence underscores the necessity of side-channel countermeasures and offers a concrete, quantitative basis for more stringent and realistic trace-count requirements in certification profiles.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (Grant Nos. 62502275 and 62172374).

References

1. Module-lattice-based digital signature standard. National Institute of Standards and Technology (NIST) Federal Information Processing Standard (FIPS) 204 (2024), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf
2. Module-lattice-based key-encapsulation mechanism standard. National Institute of Standards and Technology (NIST) Federal Information Processing Standard (FIPS) 203 (2024), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf
3. Stateless hash-based digital signature standard. National Institute of Standards and Technology (NIST) Federal Information Processing Standard (FIPS) 205 (2024), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf
4. Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Kelsey, J., Liu, Y., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D.: Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. National Institute of Standards and Technology (2020), https://doi.org/10.6028/NIST.IR.8309
5. Amer, S., Wang, Y., Kippen, H., Dang, T., Genkin, D., Kwong, A., Nelson, A., Yerukhimovich, A.: Pq-hammer: End-to-end key recovery attacks on post-quantum cryptography using rowhammer. In: IEEE Symposium on Security and Privacy (SP). pp. 3567–3582. IEEE (2025), https://doi.org/10.1109/SP61157.2025.00048
6. Aydin, F., Aysu, A., Tiwari, M., Gerstlauer, A., Orshansky, M.: Horizontal side-channel vulnerabilities of post-quantum key exchange and encapsulation protocols. ACM Transactions on Embedded Computing Systems (TECS) **20**(6), 110:1–110:22 (2021), https://doi.org/10.1145/3476799
7. Bernstein, D.J., Bhargavan, K., Bhasin, S., Chattopadhyay, A., Chia, T.K., Kannwischer, M.J., Kiefer, F., Paiva, T.B., Ravi, P., Tamvada, G.: KyberSlash: Exploiting secret-dependent division timings in kyber implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) **2025**(2), 209–234 (2025), https://doi.org/10.46586/tches.v2025.i2.209-234
8. Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS - kyber: A CCA-secure module-lattice-based KEM. In: IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367. IEEE (2018), https://doi.org/10.1109/EuroSP.2018.00032
9. Dubrova, E., Ngo, K., Gärtner, J., Wang, R.: Breaking a fifth-order masked implementation of crystals-kyber by copy-paste. In: Asia Public-Key Cryptography Workshop (APKC). pp. 10–20. ACM (2023), https://doi.org/10.1145/3591866.3593072
10. Guo, Q., Nabokov, D., Nilsson, A., Johansson, T.: SCA-LDPC: A code-based framework for key-recovery side-channel attacks on post-quantum encryption schemes. In: Advances in Cryptology (ASIACRYPT). pp. 203–236. LNCS 14441, Springer-Verlag (2023), https://doi.org/10.1007/978-981-99-8730-6_7
11. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. In: Theory of Cryptography (TCC). pp. 341–371. LNCS 10677, Springer-Verlag (2017), https://doi.org/10.1007/978-3-319-70500-2_12

12. Kannwischer, M., Petri, R., Rijneveld, J., Schwabe, P., Stoffelen, K.: PQM4: Post-quantum crypto library for the ARM Cortex-M4, https://github.com/mupq/pqm4
13. Khalid, A., Oder, T., Valencia, F., O'Neill, M., Güneysu, T., Regazzoni, F.: Physical protection of lattice-based cryptography: Challenges and solutions. In: Great Lakes Symposium on VLSI (GLSVLSI). pp. 365–370. ACM (2018), https://doi.org/10.1145/3194554.3194616
14. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology (CRYPTO). pp. 388–397. LNCS 1666, Springer-Verlag (1999), https://doi.org/10.1007/3-540-48405-1_25
15. Kundu, S., Chowdhury, S., Saha, S., Karmakar, A., Mukhopadhyay, D., Verbauwhede, I.: Carry your fault: A fault propagation attack on side-channel protected lwe-based KEM. IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) **2024**(2), 844–869 (2024), https://doi.org/10.46586/tches.v2024.i2.844-869
16. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography (DCC) **75**(3), 565–599 (2015), https://doi.org/10.1007/s10623-014-9938-4
17. Li, J., Cheng, C., Shen, M., Chen, P., Guo, Q., Liu, D., Wu, L., Weng, J.: Grafted trees bear better fruit: An improved multiple-valued plaintext-checking side-channel attack against kyber. In: Design, Automation, and Test in Europe (DATE). pp. 1–7. IEEE (2025), https://doi.org/10.23919/DATE64628.2025.10992764
18. Rajendran, G., Ravi, P., D'Anvers, J.P., Bhasin, S., Chattopadhyay, A.: Pushing the limits of generic side-channel attacks on lwe-based kems - parallel PC oracle attacks on kyber KEM and beyond. IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) **2023**(2), 418–446 (2023), https://doi.org/10.46586/tches.v2023.i2.418-446
19. Ravi, P., Chattopadhyay, A., D'Anvers, J.P., Baksi, A.: Side-channel and fault-injection attacks over lattice-based post-quantum schemes (kyber, dilithium): Survey and new results. ACM Transactions on Embedded Computing Systems (TECS) **23**(2), 35:1–35:54 (2024), https://doi.org/10.1145/3603170
20. Ravi, P., Roy, S.S., Chattopadhyay, A., Bhasin, S.: Generic side-channel attacks on cca-secure lattice-based PKE and KEMs. IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) **2020**(3), 307–335 (2020), https://doi.org/10.13154/tches.v2020.i3.307-335
21. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Symposium on Theory of Computing (STOC). pp. 84–93. ACM (2005), https://doi.org/10.1145/1060590.1060603
22. Rezaeezade, A., Yap, T., Jap, D., Bhasin, S., Picek, S.: Side-channel power trace dataset for kyber pair-pointwise multiplication on cortex-m4. Cryptology ePrint Archive, Paper 2025/811 (2025), https://eprint.iacr.org/2025/811
23. Shen, M., Cheng, C., Zhang, X., Guo, Q., Jiang, T.: Find the bad apples: An efficient method for perfect key recovery under imperfect SCA oracles - A case study of kyber. IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) **2023**(1), 89–112 (2023), https://doi.org/10.46586/tches.v2023.i1.89-112
24. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Foundations of Computer Science (FOCS). pp. 124–134. IEEE Computer Society (1994), https://doi.org/10.1109/SFCS.1994.365700
25. Tanaka, Y., Ueno, R., Xagawa, K., Ito, A., Takahashi, J., Homma, N.: Multiple-valued plaintext-checking side-channel attacks on post-quantum KEMs. IACR

Transactions on Cryptographic Hardware and Embedded Systems (TCHES) **2023**(3), 473–503 (2023), https://doi.org/10.46586/tches.v2023.i3.473-503
26. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/em analysis on post-quantum kems. IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) **2022**(1), 296–322 (2022), https://doi.org/10.46586/tches.v2022.i1.296-322
27. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a methodology for efficient CNN architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) **2020**(3), 147–168 (2020), https://doi.org/10.13154/tches.v2020.i3.147-168
28. Xu, Z., Pemberton, O., Roy, S.S., Oswald, D.F., Yao, W., Zheng, Z.: Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. IEEE Transactions on Computers (TC) **71**(9), 2163–2176 (2022), https://doi.org/10.1109/TC.2021.3122997