Grafted Trees Bear Better Fruit: An Improved Multiple-Valued Plaintext-Checking Side-Channel Attack against Kyber

Jinnuo Li*, Chi Cheng*✉, Muyan Shen†, Peng Chen*, Qian Guo‡ , Dongsheng Liu§ , Liji Wu¶ , and Jian Weng‖
*School of Computer Science, China University of Geosciences, Wuhan, China
†School of Cryptology, University of Chinese Academy of Sciences, Beijing, China
‡Lund University, Lund, Sweden
§ Huazhong University of Science and Technology, Wuhan, China
¶ School of Integrated Circuits, Tsinghua University, Beijing, China
‖College of Cyber Security, Jinan University, Guangzhou, China
Email: chengchi@cug.edu.cn

***Abstract*—As a prominent category of side-channel attacks (SCAs), plaintext-checking (PC) oracle-based SCAs offer the advantages of generality and operational simplicity on a targeted device. At TCHES 2023, Rajendran et al. and Tanaka et al. independently proposed the multiple-valued (MV) PC oracle, significantly reducing the required number of queries (a.k.a., traces) in the PC oracle. However, in practice, when dealing with environmental noise or inaccuracies in the waveform classifier, they still rely on majority voting or the other technique that usually results in three times the number of queries compared to the ideal case.**

In this paper, we propose an improved method to further reduce the number of queries of the MV-PC oracle, particularly in scenarios where the oracle is imperfect. Compared to the state-of-the-art at TCHES 2023, our proposed method reduces the number of queries for a full key recovery by more than 42.5%. The method involves three rounds. Our key observation is that coefficients recovered in the first round can be regarded as prior information to significantly aid in retrieving coefficients in the second round. This improvement is achieved through a newly designed grafted tree. Notably, the proposed method is generic and can be applied to both the NIST key encapsulation mechanism (KEM) standard Kyber and other significant candidates, such as Saber and Frodo. We have conducted extensive software simulations against Kyber-512, Kyber-768, Kyber-1024, FireSaber, and Frodo-1344 to validate the efficiency of the proposed method. An electromagnetic attack conducted on real-world implementations, using an STM32F407G board equipped with an ARM Cortex-M4 microcontroller and Kyber implementation from the public library *pqm4*, aligns well with our simulations.

***Index Terms*—NIST post-quantum cryptography standardization, Lattice-based cryptography, Kyber, Side-channel attacks, multiple-valued plaintext-checking oracle**

I. Introduction

With the advent of quantum computing, mathematical problems integral to current public-key cryptographic systems, such as integer factorization or discrete logarithms, will be efficiently solved by Shor's algorithm [1]. Recognizing this threat, in 2016, the National Institute of Standards and Technology (NIST) initiated the standardization process for post-quantum cryptography (PQC), seeking candidates conjectured to resist both classical and quantum attacks [2].

Lattice-based cryptographic algorithms constitute a significant portion of several rounds of NIST selection. Among them, CRYSTALS-Kyber [3], which is based on the Module Learning with Errors (M-LWE) problem, was ultimately selected for standardization of public-key encryption (PKE) or key encapsulation mechanism (KEM) in July 2022 [4]. As a consequence, on August 14th, 2024, NIST released FIPS 203 ML-KEM, a standard that adopts CRYSTALS-Kyber as the cornerstone.

Given the anticipated widespread implementation of CRYSTALS-Kyber across diverse computational platforms and applications, it is imperative to assess its implementation security comprehensively. This evaluation includes a thorough examination of vulnerability to Side-Channel Attacks (SCAs). Initially introduced by Kocher in 1996 [5], SCAs exploit various types of leaked information, such as timing, cache timing, power consumption or electromagnetic emanation, to glean information about the long-term secret key or message in cryptographic algorithms.

In the design of LWE-based NIST PQC PKEs/KEMs, a common approach is to utilize the Fujisaki-Okamoto (FO) transformation [6]. This involves starting with a construction secure against chosen plaintext attacks (CPA) and then transforming it to be secure against chosen ciphertext attacks (CCA). A prominent research focus has been to identify and mitigate side-channel leakages that could compromise the CCA protection offered by the FO transformation.

A. Related works

SCAs against CCA-secure KEMs. NIST encourages further research on the security of post-quantum cryptographic schemes against side-channel attacks (SCAs) to strengthen cryptographic systems. Several SCAs targeting CCA-secure KEMs have been proposed [7]–[14], varying in their models and attack complexities, such as the required number of queries or traces.

The first model, the plaintext-checking (PC) oracle, compares the decrypted message to a predetermined value, revealing 1 bit of key information per query [9], [15]. The second model,

978-3-9826741-0-0/DATE25/© 2025 EDAA

the decryption-failure (DF) oracle, exploits decryption failures correlated with the secret key, especially in non-constant-time implementations [8], [16], [17]. Both attack models extract limited information per query, making them more generic.

The full-decryption (FD) oracle model further reduces the required traces by exploiting specific leaky operations during decryption [10], [18]. While FD oracle attacks are more efficient, defending against PC oracle-based attacks can impose a significant performance cost.

Parallel variant of the PC oracle. The concept of the parallel PC oracle, or Multiple-valued (MV) PC oracle, was introduced at TCHES 2023 [19], [20]. It addresses a limitation of binary PC oracle-based attacks, which extract only one bit of information from extensive leakage traces during the re-encryption process. Since power traces leak partial secret information at multiple points, there is potential to extract several bits per trace, enhancing attack efficiency. The parallel PC oracle strikes a balance between binary PC oracle and FD oracle attacks in terms of trace requirements, needing fewer traces than the former but more than the latter. Notably, the MV-PC oracle attack poses significant challenges for countermeasures, similar to the binary PC oracle.

Robust recovery in practical settings. A key challenge in implementing SCAs in real-world scenarios is the presence of noise or classifier inaccuracies [21], [22], which can introduce errors in oracle responses and affect key recovery. A common solution is majority voting. For example, Ravi et al. [9] used three-vote majority voting, increasing the total number of queries by threefold. More advanced techniques, like negative-log likelihood (NLL), can improve oracle accuracy, especially when deep learning classifiers are used [15], [20], [23].

Shen et al. [24] proposed an error detection method for binary PC attacks, reducing the number of traces by 45.9% to 55.4% compared to previous attacks on Kyber512. However, handling errors in MV-PC oracles remains unsolved. Integrating Shen et al.'s error detection into MV-PC oracles presents challenges, as their method is incompatible with parallel key recovery.

In this paper, we propose a new approach for locating and correcting error coefficients in MV-PC oracles, particularly in noisy environments. Our method aims to enable parallel key recovery while significantly reducing the number of required queries.

B. Contributions

The principal contributions of this study are:

- We propose an efficient MV-PC oracle-based side-channel attack against Kyber, designed for an imperfect oracle. Our innovative approach involves a novel method for efficient detection and correction of errors in the MV-PC oracle. There are 3 rounds in our improved method. In round 1, a parallel key recovery is executed. Our key observation is that the coefficients recovered in the first round can be regarded as prior information to facilitate key recovery in the second round. As a result, in the second round, we can considerably decrease the number of queries required through a newly created grafted tree. The grafted tree is designed to enable retrieval of each coefficient in nearly two queries. Compared to the latest attack [20] [19] at TCHES 2023, our proposed attack reduces the number of queries for a full key recovery by more than 42.5%.
- We center our research on Kyber, recognized as the primary future NIST KEM standard. Notably, our method is generic: It is also applicable to similar schemes, such as Saber and Frodo, two LWE/LWR-based candidates from the second/third rounds of the NIST PQC competition. Our extensive experiments cover all security levels of Kyber (Kyber-512, Kyber-768, Kyber-1024), FireSaber, and Frodo-1344. We base the MV-PC oracle on a multiple-classification neural network model. Further, we validate our proposed method in real-world scenarios by conducting an electromagnetic attack on an STM32F407G board equipped with an ARM Cortex-M4 microcontroller and Kyber implementation from the publicly available testing and benchmarking library *pqm4*. The results from the real-world implementation align well with our prior simulations.
- In high-noise environments, we have enhanced our method to fit such scenarios. Further analysis reveals that compared to majority voting, our method can decrease the number of queries by 29.4% and 22.8% at an accuracy of 95% and 90%, respectively. This underscores the versatility and effectiveness of our approach.

II. Background

A. Kyber

The security of Kyber is based on the M-LWE problem. As a variant of the LWE problem, the M-LWE problem is to distinguish $(\mathbf{A}, \mathbf{B} = \mathbf{As} + \mathbf{e}) \in \mathcal{R}_q^{l \times l} \times \mathcal{R}_q^l$ from uniformly selected $(\mathbf{A}, \mathbf{B}) \in \mathcal{R}_q^{l \times l} \times \mathcal{R}_q^l$. The parameter l is set to be 2, 3, or 4, which corresponds to three different security levels in Kyber: Kyber-512, Kyber-768, and Kyber-1024, respectively.

Typically, a KEM consists of three parts: key generation, encapsulation, and decapsulation. The CCA-secure Kyber KEM is constructed from a simple CPA-secure PKE, which can be divided into KYBER.CPA.KeyGen, KYBER.CPA.Enc and KYBER.CPA.Dec. The process of transforming a CPA-secured PKE into a CCA-secure KEM is based on Fujisaki-Okamoto transform and its variants. Two hash functions $\mathcal{G}(\cdot)$ and $\mathcal{H}(\cdot)$ are used in the encapsulation and decapsulation process, and $\mathrm{KDF}(\cdot)$ denotes a key-derivation function. The main parts of Kyber encapsulation and decapsulation, ignoring some details such as number-theoretic transformations (NTTs), are depicted in Algorithm 1.

The following discusses the PC oracle and the corresponding MV-PC oracle.

B. From PC oracle to MV-PC oracle

1) PC oracle: In this paper, we take into consideration a chosen-ciphertext attack against FO transformation with the help of side-channel leakages, which can be modeled as the well-known PC oracle-based attack [9]. In a PC oracle, there

Algorithm 1 KYBER.CCA KEM

KYBER.KEM.KeyGen
Output: sk', pk
1: Generate a pseudo-random coin z
2: ∘ **KYBER.CPA.KeyGen**
3: $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{l\times l}$, $\mathbf{s}, \mathbf{e} \xleftarrow{\$} \mathcal{B}_{\eta_1}^l$
4: $\mathbf{b} := \mathbf{A}^T\mathbf{s} + \mathbf{e}$
5: $sk = \mathbf{s}$, $pk := (\mathbf{A}, \mathbf{b})$
6: $sk' := (sk||pk||\mathcal{H}(pk)||z)$

KYBER.CCA.Encaps
Input: pk
Output: ct, K
1: $\mathbf{m} \xleftarrow{\$} \{0,1\}^{256}$, $(\bar{K}, r) = \mathcal{G}(\mathbf{m}, \mathcal{H}(pk))$
2: ∘ **KYBER.CPA.Enc**$(pk, \mathbf{m}, r)$
3: $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2 \xleftarrow{\$} \mathcal{B}_{\eta_2}^l$
4: $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$, $\mathbf{v} = \mathbf{b}^T\mathbf{r} + \mathbf{e}_2 + \text{DeComp}_q(\mathbf{m}, 1)$
5: $\mathbf{c}_1 := \text{Comp}_q(\mathbf{u}, d_u)$, $\mathbf{c}_2 := \text{Comp}_q(\mathbf{v}, d_v)$
6: $ct := (\mathbf{c}_1, \mathbf{c}_2)$
7: $K := \text{KDF}(\bar{K}, \mathcal{H}(ct))$

KYBER.CCA.Decaps
Input: sk', ct
Output: 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: $K := \text{KDF}(\bar{K}', \mathcal{H}(ct))$
6: **else**
7: $K := \text{KDF}(z, \mathcal{H}(ct))$
8: **end if**

is an honest user Alice who implements Kyber on her devices for key establishment. An adversary Eve acts as a valid user Bob to negotiate shared keys with Alice. The adversary sends a series of well-chosen ciphertexts to Alice and exploits side-channel leakage to help recover Alice's secret key sk. To be specific, in a PC oracle, for chosen ciphertext ct and message $\mathbf{m}$, the oracle tells whether the decrypted $\mathbf{m}'$ (line 1 of **KYBER.CCA.Decaps()** in Algorithm 1) equals $\mathbf{m}$ or not. Since $\mathbf{m}$ and $\mathbf{m}'$ determine the final key K, we can also know whether the keys generated by the two parties match or not. Hence, a PC oracle-based attack is also called the key mismatch attack [25], [26].

In the following, we take Kyber-1024 as an example to introduce the PC oracle-based attack. Assume Alice's secret key is $\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1)$. At first Eve sets reference plaintext $\mathbf{m} = (1, 0, \ldots, 0)$, $\mathbf{u} = (\lceil q/32 \rfloor, 0, \cdots, 0)$ and $\mathbf{c}_2 = (k, 0, \ldots, 0)$, where $k \in \mathbb{Z}_q$ is a parameter used to help recover the secret key. Then, Eve computes $\mathbf{c}_1 = \mathbf{Comp}_q(\mathbf{u}, d_u)$ and packs $\mathbf{c}_1$ and $\mathbf{c}_2$ into ct and sends it to Alice. With $(\mathbf{c}_1, \mathbf{c}_2)$, Alice computes $\mathbf{u}' = \mathbf{DeComp}_q(\mathbf{c}_1, d_u)$ and $\mathbf{v}' = \mathbf{DeComp}_q(\mathbf{c}_2, d_v) = (\lceil (q/32)k \rfloor, 0, \cdots, 0)$. Alice goes on calculating $\mathbf{m}'$, and the relationship between $\mathbf{m}'[0]$ and $\mathbf{s}_0[0]$ can be built as follows:

$$\mathbf{m}'[0] = \mathbf{Comp}_q((\mathbf{v} - \mathbf{s}_0^T\mathbf{u})[0], 1) \quad (1)$$
$$= \lceil (2/q) \left(\lceil (q/32)k \rfloor - \mathbf{s}_0[0] \lceil q/32 \rfloor \right) \rfloor \bmod 2. \quad (2)$$

We can see that the value of $\mathbf{m}'[0]$ relies only on k and $\mathbf{s}_0[0]$. Therefore, we can pre-calculate the relationship between k and $\mathbf{m}'[0]$ for each coefficient $\mathbf{s}_0[0]$. Subsequently, by constructing a specialized ciphertext ct and making multiple queries to the oracle, the adversary can progressively narrow down the possible values for $\mathbf{s}_0[0]$, eventually obtaining its precise value. Within an iteration, $\mathbf{u}$ remains fixed, and the value of k is adjusted based on the responses received from the PC oracle. Consequently, if k is appropriately chosen, the attacker could efficiently recover $\mathbf{s}_0[0]$ with a minimal number of queries.

2) BRT for PC oracle: In [25], Qin et al. introduced the concept of the binary recovery tree (BRT), which serves as a common framework for explaining both binary PC oracle and MV-PC oracle concepts. The BRT serves as a visual representation of the adaptive selection of ciphertext based on oracle responses. Here, in the context of a BRT, the depth represents the number of steps or hops from a leaf node to the root node. It signifies the cumulative number of queries made during the process. The BRT is constructed with the help of Huffman coding, i.e., the higher the probability of occurrence of a secret coefficient, the fewer number of queries (lower depth in the BRT) it will require to uniquely distinguish it.

For PC oracle-based attack against Kyber-1024, the BRTs are shown in Figure 1. Note that the BRT for Kyber-1024 is not uniquely determined, and we can find two such trees with the same number of queries.

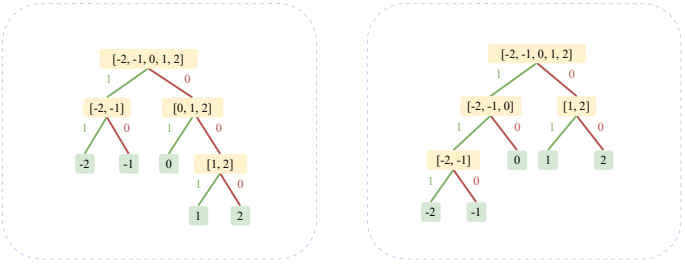

Fig. 1: BRTs in PC oracle-based attack against Kyber-1024

3) MV-PC oracle: The core idea of MV-PC oracle is to construct special ciphertexts such that the first N bits of $\mathbf{m}'$ depend on the N corresponding coefficients of the secret key. Specifically, Eve selects $\mathbf{u} = (\lceil q/32 \rfloor, 0, \cdots, 0)$, and $\mathbf{c}_2 = (k_0, k_1, \cdots, k_{N-1}, 0, \cdots, 0)$, then Eve compresses $\mathbf{c}_1 = \mathbf{Comp}_q(\mathbf{u}, d_u)$ and sends $ct := (\mathbf{c}_1, \mathbf{c}_2)$ to Alice. With received ct, Alice decompresses $\mathbf{u} = \mathbf{DeComp}_q(\mathbf{c}_1, d_u)$ and $\mathbf{v} = \mathbf{DeComp}_q(\mathbf{c}_2, d_v)$. Similar to the analysis in (2), Alice decrypts and gets:

$$\mathbf{m}'[i] = \begin{cases} \lceil (2/q) \left(\lceil (q/32)k_i \rfloor - \mathbf{s}_0[i] \lceil q/32 \rfloor \right) \rfloor \bmod 2, i < N \\ \lceil (2/q) \left(0 - \mathbf{s}_0[i] \lceil q/32 \rfloor \right) \rfloor \bmod 2 = 0, i \geq N. \end{cases} \quad (3)$$

With the parallelization factor N, the ciphertext ct can be decrypted to one of the 2^N known plaintexts. Similarly, we need to adjust our strategy to employ a parallel approach in an SCA-assisted chosen ciphertext attack. First, the waveforms generated during the decryption process are collected to train a multiple-valued classifier. We use $\mathcal{W}_0, \cdots, \mathcal{W}_{2^N-1}$ to denote different waveforms corresponding to $\mathbf{m}_0, \cdots, \mathbf{m}_{2^N-1}$, respectively. Then, for each attack, the waveform $\mathcal{W}$ generated in the hash function $\mathcal{G}(\cdot)$ is collected and classified as one of $\mathcal{W}_0, \cdots, \mathcal{W}_{2^N-1}$. $\mathcal{W} = \mathcal{W}_i$ means that $\mathcal{W}$ is classified to $\mathcal{W}_i$ and the attacker will be able to derive $\mathbf{m}'$ generated in **KYBER.CCA.Decaps**, which is equal to $\mathbf{m}_i$.

III. Challenges and our new method

A. Challenges and our observations

Recall that both the PC oracle and MV-PC oracle are instantiated with side-channel attacks. In practice, the accuracy

of the oracle may be influenced by environmental noises and countermeasures against side-channel attacks, such as shuffling and masking. Furthermore, the performance of the classifier itself can also impact accuracy, especially for the MV-PC oracle, since 2^N (where $N > 1$) classifications are more complex than binary classification, which can lead to lower accuracy.

When the retrieved results contain errors, the traditional approach is to use majority voting, which leverages multiple inference results to improve the accuracy of the classifier further. Instead of majority voting, Shen et al. treat the detection of errors as a coding problem and propose an efficient method called fast-checking to identify error locations [24]. For the PC oracle, an attacker with two accesses to the oracle can employ fast-checking to verify the accuracy of secret key coefficients at four locations. To be specific, there are three steps in [24]. In Step 1, a roughly correct key is retrieved. Then, in Step 2 the attacker selects some attack parameters to detect errors in the retrieved key coefficients. For example, to check whether the retrieved coefficient block $\mathbf{s_0}[0], \mathbf{s_0}[1], \mathbf{s_0}[2], \mathbf{s_0}[3]$ are correct or not, $\mathbf{u} = \mathbf{u_{atk}}[0] - \mathbf{u_{atk}}[1]x^{255} - \mathbf{u_{atk}}[2]x^{254} - \mathbf{u_{atk}}[3]x^{253}$ and $\mathbf{v} = \mathrm{v_{atk}}$ are generated from the values of $\mathbf{s_0}[0], \mathbf{s_0}[1], \mathbf{s_0}[2], \mathbf{s_0}[3]$. Here, the selection of attack parameters $\mathbf{u_{atk}}[0], \mathbf{u_{atk}}[1], \mathbf{u_{atk}}[2], \mathbf{u_{atk}}[3], \mathrm{v_{atk}}$ ensures that if the recovered coefficients $\mathbf{s_0}[0], \mathbf{s_0}[1], \mathbf{s_0}[2], \mathbf{s_0}[3]$ are correct, the resulted codeword is special. So an interesting question arises here: Can we adapt fast-checking to the MV-PC oracle? In fact, this is challenging. The challenge stems from conflicts in the selection of attack parameters. In MV-PC oracle, if we want to check two different coefficient blocks $\mathbf{s_0}[0], \mathbf{s_0}[1], \mathbf{s_0}[2], \mathbf{s_0}[3]$ and $\mathbf{s_0}[4], \mathbf{s_0}[5], \mathbf{s_0}[6], \mathbf{s_0}[7]$ at the same time, we need to set $\mathbf{u} = \mathbf{u_{atk}}[0] - \mathbf{u_{atk}}[1]x^{255} - \mathbf{u_{atk}}[2]x^{254} - \mathbf{u_{atk}}[3]x^{253}$ and $\mathbf{v} = \mathrm{v_{atk}}$ and gets 1 bit of information from $\mathbf{m'}[0]$:

$$\mathbf{m'}[0] = \left\lceil \frac{2}{q}(\mathrm{v_{atk}} - (\sum_{n=0}^{3} \mathbf{s_0}[n]\mathbf{u_{atk}}[n])) \right\rfloor \bmod 2. \quad (4)$$

Meanwhile, we need to set $\mathbf{u} = \mathbf{u'_{atk}}[0] - \mathbf{u'_{atk}}[1]x^{255} - \mathbf{u'_{atk}}[2]x^{254} - \mathbf{u'_{atk}}[3]x^{253}$ and $\mathbf{v} = \mathrm{v'_{atk}}$ to get another 1 bit of information from $\mathbf{m'}[4]$:

$$\mathbf{m'}[4] = \left\lceil \frac{2}{q}(\mathrm{v'_{atk}} - (\sum_{n=0}^{3} \mathbf{s_0}[n+4]\mathbf{u'_{atk}}[n])) \right\rfloor \bmod 2. \quad (5)$$

However, this incurs conflicts in the setting of $\mathbf{u}$ for two blocks with different coefficients. An obvious question is whether, in the parallel case (where the attacker can get multiple bits of information), a more efficient method could be found – potentially by better leveraging prior knowledge – than the one proposed by [24] to expedite the process.

B. Our basic idea

In this part, we describe the general full-key recovery framework of the new attack. We start by introducing the basic ideas.

In Figure 2, we illustrate the main idea of our key recovery strategy. In majority voting, we need to repeat the same key recovery procedure several times. Our key observation is that the coefficients recovered in the previous round can be regarded as prior information to facilitate the recovery of coefficients in the subsequent round. Consequently, in our strategy, after employing an MV-PC oracle to recover the initial round coefficients, we establish a grafted tree to guide the recovery of the second round coefficients. This approach enables us to achieve significant reductions in the number of required queries. We summarize the main process as follows:

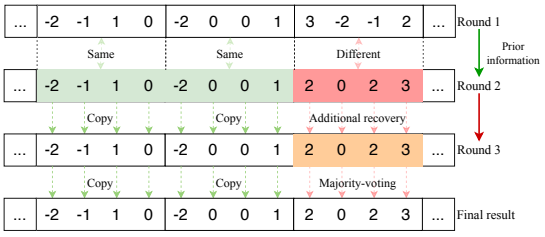

Fig. 2: Our main idea

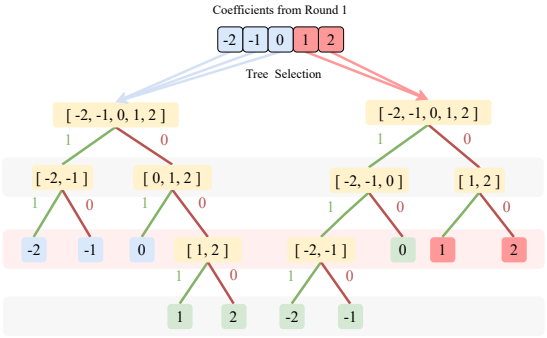

Fig. 3: A grafted tree for Kyber-1024 and Kyber-768, including a BRT for coefficients -2, -1, 0 (left) and a BRT for coefficients 1, 2 (right)

- Round 1: Parallel key recovery is executed to obtain approximately correct secret key coefficients using MV-PC oracle.
- Round 2: After the parallel key recovery, with the prior information acquired in round 1, we construct the corresponding grafted tree and perform another parallel key recovery with the grafted tree. We compare each coefficient block recovered in round 2 with the corresponding coefficient block in round 1. If a recovered coefficient block exhibits a value different from that in the round 1 recovery, the corresponding coefficient block is labeled as suspicious.
- Round 3: If the retrieved coefficient blocks match in round 1 and round 2, we directly classify these blocks as correctly recovered. For the identified suspicious coefficient blocks, an additional parallel recovery is conducted on these blocks. Subsequently, the coefficient values obtained from the three rounds undergo a majority voting process. The result of the majority voting for coefficients is considered the correct outcome.

C. The grafted tree for key recovery

The grafted tree. Our key observation is that, with prior information about which coefficient has been retrieved in round

1, we can choose different branches in different BRTs to further reduce the needed number of queries. We call the new construct the grafted tree. More specifically, taking Kyber-1024 as an example, for coefficients recovered as [-2, 0] in round 1, we employ the left and right branches of the left BRT in Figure 1; While for [1, 2], the right branch of the right BRT in Figure 1 is leveraged. In Figure 3, it can be seen that each coefficient retrieved in round 1 has depth of 2.

Here a question arises, for each coefficient, can we find a BRT in which the depth of the corresponding leaf node has the depth of 2? We summarize the results in the following.

Proof 1: The problem is equivalent to proving that for each coefficient, we can distinguish it from other coefficients by accessing the oracle twice. We summarize all the results for Kyber in Table I. In Table I, k_1, k_2 denotes the value of k in the ciphertext constructed during the first and second access to the oracle, respectively. 2(11) means the retrieved coefficient is 2, and 11 is the output sequence of the oracle. From the result in Table I, we conclude that for each coefficient, it is always possible to construct a BRT in which the leaf node corresponding to this coefficient has a depth of 2.

TABLE I: Distinguishing coefficients by ciphertext pairs.

(k_1, k_2)	Distinguished	Others
(7,8)	-2(11), -1(01)	{0, 1, 2}(00)
(8,9)	0(01)	{-1, -2}(11), {1, 2}(00)
(9,10)	1(01), 2(00)	{0, 1, 2}(11)

In contrast to previous attack methods, in round 2, our approach enables the selection of distinct BRTs. This flexibility arises from our knowledge of the approximate correct coefficient values, which serve as valuable prior information when constructing the grafted tree. This enables us to utilize the optimal BRT for each coefficient, accelerating our attack and reducing overall attack overheads.

The remaining problem is how to construct a grafted tree. We treat the generation of a grafted tree as an expansion of the BRT generation process. Certain principles need to be followed: For each coefficient, position this coefficient as low as possible in the tree, without regard for the positions of the leaf nodes where the other coefficients are situated. There is no need to construct a large number of BRTs during preparation, as coefficients only need to be placed on the second level of a binary tree. In Kyber-1024 and Kyber-768, only 2 BRTs are needed to form a grafted tree, instead of 5, as shown in Figure 3. For Kyber-512 we need only 4 BRTs.

IV. Experiments and analysis

This section presents the results and analysis of empirical studies designed to objectively assess the effectiveness of our improved method.

A. Software simulations

1) Simulation settings: All our simulations are conducted on a desktop featuring a 3 GHz Intel Core i5-7400 CPU and 16 GB RAM. Our code is derived from the C implementation of Kyber submitted to the third round of the NIST PQC project.

TABLE II: Comparison with existing methods for full key recovery. We conduct the full key recovery process $10,000$ times with random selected secret keys.

Schemes	N	**Method**	**#EvQuery**	**#EvError**
Kyber-512	4	Tanaka et al.	1152 (ref)	0.00
		Rajendran et al.	1150.8	0.001
		Our Method	642.0 (-44.2%)	0.002
	8	Tanaka et al.	576 (ref)	0.00
		Rajendran et al.	576.0	0.001
		Our Method	321.2 (-44.2%)	0.002
Kyber-768	4	Tanaka et al.	1728 (ref)	0.00
		Rajendran et al.	1599.3	0.001
		Our Method	919.7 (-42.5%)	0.004
	8	Tanaka et al.	864 (ref)	0.00
		Rajendran et al.	849.5	0.0005
		Our Method	476.7 (-43.9%)	0.002
Kyber-1024	4	Tanaka et al.	2304 (ref)	0.00
		Rajendran et al.	2132.1	0.001
		Our Method	1226.4 (-42.5%)	0.002
	8	Tanaka et al.	1152 (ref)	0.00
		Rajendran et al.	1132.9	0.001
		Our Method	635.5 (-43.9%)	0.001
FireSaber	4	Tanaka et al.	2304(ref)	0.00
		Rajendran et al.	2301.9	0.002
		Our Method	1283.9 (-44.2%)	0.004
	8	Tanaka et al.	1152 (ref)	0.00
		Rajendran et al.	1152.0	0.003
		Our Method	642.4 (-44.2%)	0.004
Frodo-1344	4	Tanaka et al.	21504(ref)	0.00
		Our Method	16253.6 (-24.4%)	0.014
	7	Tanaka et al.	18432 (ref)	0.00
		Our Method	11434.0 (-38.0%)	0.218

2) Comparison with existing work: In our software simulations, we validate the improvement of our proposed method over the previous work of Tanaka et al [20] and Rajendran et al [19]. They instantiate the MV-PC oracle using a multiple-classification Neural Network (NN) model and a t-test-based classifier, respectively. Accuracy is the same as that given in [20], which is around 99%, varying according to the scheme.

We employ $\#\mathbf{EvQuery}$ to signify the average number of queries needed for full key recovery and **#EvError** to denote the average number of error coefficients present in the final result. The latter serves as a crucial indicator of the effectiveness of the full key recovery. Both our approach and previous methods aim to minimize **#EvError** to below 1.0. In comparison to existing work, our proposed method exhibits significant improvements. Specifically, for Kyber-1024, we achieve a reduction of 42.5% and 43.9% in total queries when $N = 4$ and 8, respectively. Similarly, for other systems, our method also shows significant reductions.

The experimental results presented in Table II also demonstrate that our new method can be easily applied to other KEM schemes that use variations of FO such as Saber [27] and Frodo.

3) Experiments for lower accuracy: Next, we will further evaluate the performance of our model with lower accuracy to demonstrate the adaptive capability of the proposed method even in high noise environments, with some adjustments. Specifically, we conduct experiments for Kyber-1024 with $N = 8$ and compare our results with those in [19] (with

majority voting). The results are displayed in Table III.

TABLE III: Comparison between majority voting and proposed method for full key recovery in low accuracy.$\mathbf{t}$ represents the number of votes cast.

Accuracy	Method	#EvQuery	#EvError
95.00%	Majority Voting ($\mathbf{t}=5$)	1888.0	0.419
	Our Method	1336.8 (-29.2%)	0.324
90.00%	Majority Voting ($\mathbf{t}=7$)	2642.6	0.460
	Our Method	2041.6 (-22.7%)	0.629
80.00%	Majority Voting ($\mathbf{t}=15$)	5665.78	0.005
	Our Method	4625.8 (-18.3%)	0.097
70.00%	Majority Voting ($\mathbf{t}=23$)	8691.9	0.110
	Our Method	7295.8 (-16.1%)	0.089

In low-accuracy scenarios, we still need three rounds. To compensate for the problems associated with accuracy degradation, we use majority voting to help refine the output of the MV-PC oracle. As depicted in Table III, although the use of oracle combined with majority voting increases the overhead of the attack, our method applied to Kyber-1024 exhibits comparable **#EvError** to majority voting. These findings highlight the efficiency of our method even in high-noise environments, where the performance of the MV-PC oracle is typically hindered.

B. Real-world experiments

1) Experiment settings: In this subsection, we conduct experiments to validate the feasibility and efficiency of our proposed attack in real-world scenarios. The experiments are implemented on an STM32F407G board, featuring an ARM Cortex-M4 microcontroller. We choose Cortex-M4 since NIST has recommended it for efficiency evaluation in their postquantum cryptography standardization. The ARM-optimized Kyber-512 implementation from the publicly available testing and benchmarking library *pqm4* [28] is executed on the board.

Then, we employ a PicoScope 3403D oscilloscope and a CYBERTEK EM5030-3 EM Probe to collect the waveforms. Each collected waveform contains 50,000 sample points, and the sample rate is set at 500 MHz.

We first construct the special ciphertext and call the **KYBER.CCA.Decaps** function to collect the waveform. By constantly adjusting the value of $\mathbf{c}_2$, we can obtain waveforms corresponding to $\mathbf{m}_0, \cdots, \mathbf{m}_{2^N-1}$, respectively. For each plaintext, we collect 200 waveforms, of which 100 are used for training, 50 for validation, and 50 for testing. We aim to train a 2^8 class classifier with the collected waveforms. Specifically, we employ CUDA 12.2, cuDNN 8.3, Keras 2.10.0, and Tensorflow-gpu 2.10.1 on a desktop equipped with Intel Core i9-12900K and NVIDIA GeForce RTX 3030 to train an NN model.

We adopt the NN framework proposed by Tanaka et al. [20] with some modifications according to the format of our collected waveforms. To ensure the robustness of our model, we get an average accuracy of 99.86% over 100 iterations. Finally, we perform full key recovery as shown in Section III using the trained classifier.

TABLE IV: Comparison of results between real-world and simulated attacks on Kyber-512. We conduct the full key recovery process 10 times with random secret keys.

$N=8$	$\#\mathbf{EvQuery}$	#EvError
Real-world	321.9	0.20
Simulations	321.2	0.003

2) Experiment results: In Table IV, we present the results of both software simulations and real-world experiments. In the real-world experiments, we achieve a full key recovery with an average of only 321.9 queries using the proposed new method. The close correspondence between the results of real-world attacks and software simulations validates the usability and efficiency of our method in real-world scenarios.

V. Countermeasures

An effective countermeasure against the presented attack is to filter out illegal ciphertexts through sanity checking. Another approach involves refreshing the secret key when detecting illegal ciphertexts. Both methods inevitably introduce additional computational overhead. Achieving a balance between efficiency and security remains a critical challenge in practical implementations.

VI. Conclusions

In this paper, we have proposed an improved MV-PC oracle-based side-channel attack against LWE-based KEMs, leveraging grafted BRTs to efficiently utilize prior information. Simulations and real-world implementations have shown that our method reduces the number of queries for a full key recovery by more than 42.5%, compared to the state-of-the-art at TCHES 2023.

In practice, the number of queries can be further reduced by combining our method with post-processing techniques such as lattice reduction. We can recover only a part of the coefficients and then recover the remaining ones via the lattice reduction framework in an offline manner. For example, following the work in [29], we can use the LWE estimator in [30] to estimate the coefficients to be recovered and further reduce the number of queries for Kyber-512, Kyber-768, and Kyber-1024 by 34%, 29%, and 27%, respectively, with the ability to perform 2^{32} offline computations.

Acknowledgment

The work was supported in part by the National Natural Science Foundation of China (Nos. 62172374, 62332007, U22B2028), the Science and Technology Major Project of Tibetan Autonomous Region of China (No. XZ202201ZD0006G), Open Research Fund of Machine Learning and Cyber Security Interdiscipline Research Engineering Center of Jiangsu Province(No. SDGC2131), National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong Hong Kong Joint Laboratory for Data Security and Privacy Protection, and Engineering Research Center of Trustworthy AI, Ministry of Education, Swedish Civil Contingencies Agency (No. 2020-11632) and the Crafoord Foundation.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] D. Moody, *Post Quantum Cryptography Standardization: Announcement and outline of NIST's Call for Submissions*. PQCrypto 2016, Fukuoka, Japan, 2016, https://csrc.nist.gov/Presentations/2016/Announcement-and-outline-of-NIST-s-Call-for-Submis.

[3] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber: Algorithm specification and supporting documentation (version 2.0)," in *Submission to the NIST post-quantum project (2019)*, 2019, https://pq-crystals.org/kyber.

[4] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody *et al.*, "Status report on the third round of the nist post-quantum cryptography standardization process," *US Department of Commerce, NIST*, 2022.

[5] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology - CRYPTO*, 1996, pp. 104–113.

[6] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Annual International Cryptology Conference*. Springer, 1999, pp. 537–554.

[7] J.-P. D'Anvers, M. Tiepelt, F. Vercauteren, and I. Verbauwhede, "Timing attacks on error correcting codes in post-quantum schemes," in *Proceedings of ACM Workshop on Theory of Implementation Security Workshop*, 2019, pp. 2–9.

[8] Q. Guo, T. Johansson, and A. Nilsson, "A key-recovery timing attack on post-quantum primitives using the fujisaki-okamoto transformation and its application on frodokem," in *Annual International Cryptology Conference*. Springer, 2020, pp. 359–386.

[9] P. Ravi, S. S. Roy, A. Chattopadhyay, and S. Bhasin, "Generic side-channel attacks on cca-secure lattice-based pke and kems." *CHES*, pp. 307–335, 2020.

[10] K. Ngo, E. Dubrova, Q. Guo, and T. Johansson, "A side-channel attack on a masked ind-cca secure saber kem," *CHES*, p. 676–707, 2021.

[11] M. Hamburg, J. Hermelink, R. Primas, S. Samardjiska, T. Schamberger, S. Streit, E. Strieder, and C. van Vredendaal, "Chosen ciphertext k-trace attacks on masked cca2 secure kyber," *CHES*, pp. 88–113, 2021.

[12] Q. Guo, D. Nabokov, A. Nilsson, and T. Johansson, "SCA-LDPC: A code-based framework for key-recovery side-channel attacks on post-quantum encryption schemes," in *Advances in Cryptology - ASIACRYPT*, 2023, pp. 203–236.

[13] Z. Ni, A. Khalid, W. Liu, and M. O'Neill, "Bitstream fault injection attacks on crystals kyber implementations on fpgas," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024, pp. 1–6.

[14] S. Kundu, S. Chowdhury, S. Saha, A. Karmakar, D. Mukhopadhyay, and I. Verbauwhede, "Carry your fault: A fault propagation attack on side-channel protected lwe-based kem," *CHES*, pp. 844–869, 2024.

[15] R. Ueno, K. Xagawa, Y. Tanaka, A. Ito, J. Takahashi, and N. Homma, "Curse of re-encryption: A generic power/em analysis on post-quantum kems," *CHES*, pp. 296–322, 2022.

[16] S. Bhasin, J.-P. D'Anvers, D. Heinz, T. Pöppelmann, and M. Van Beirendonck, "Attacking and defending masked polynomial comparison for lattice-based cryptography," *CHES*, pp. 334–359, 2021.

[17] J.-P. D'Anvers, D. Heinz, P. Pessl, M. Van Beirendonck, and I. Verbauwhede, "Higher-order masked ciphertext comparison for lattice-based cryptography," *CHES*, pp. 115–139, 2022.

[18] Z. Xu, O. M. Pemberton, S. Sinha Roy, D. Oswald, W. Yao, and Z. Zheng, "Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber," *IEEE Transactions on Computers*, 2021.

[19] G. Rajendran, R. Ravi, J.-P. D'Anvers, S. Bhasin, and A. Chattopadhyay, "Pushing the limits of generic side-channel attacks on lwe-based kems-parallel pc oracle attacks on kyber kem and beyond," *CHES*, pp. 418–446, 2023.

[20] Y. Tanaka, R. Ueno, K. Xagawa, A. Ito, J. Takahashi, and N. Homma, "Multiple-valued plaintext-checking side-channel attacks on post-quantum kems," *CHES*, pp. 473–503, 2023.

[21] J.-P. D'Anvers, M. Van Beirendonck, and I. Verbauwhede, "Revisiting higher-order masked comparison for lattice-based cryptography: Algorithms and bit-sliced implementations," *IEEE Transactions on Computers*, vol. 72, no. 2, pp. 321–332, 2023.

[22] R. Ueno, N. Homma, and T. Aoki, "Toward more efficient dpa-resistant aes hardware architecture based on threshold implementation," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, 2017, pp. 50–64.

[23] A. Ito, K. Saito, R. Ueno, and N. Homma, "Imbalanced data problems in deep learning-based side-channel attacks: analysis and solution," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3790–3802, 2021.

[24] M. Shen, C. Cheng, X. Zhang, Q. Guo, and T. Jiang, "Find the bad apples: An efficient method for perfect key recovery under imperfect sca oracles–a case study of kyber," *CHES*, pp. 89–112, 2023.

[25] Y. Qin, C. Cheng, X. Zhang, Y. Pan, L. Hu, and J. Ding, "A systematic approach and analysis of key mismatch attacks on lattice-based nist candidate kems," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2021, pp. 92–121.

[26] X. Zhang, C. Cheng, and T. Yu, "Further theoretical analysis of key mismatch attacks —a case study of ntru-hrss," *Acta Electronica Sinica*, vol. 51, no. 4, pp. 1081–1092, 2023.

[27] J.-P. D'Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren, "Saber: Mod-lwr based kem algorithm specification and supporting documentation," in *Submission to the NIST post-quantum project (2019)*, 2019, https://www.esat.kuleuven.be/cosic/publications/article-3055.pdf.

[28] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and benchmarking nist pqc on arm cortex-m4," 2019, https://github.com/mupq/pqm4.

[29] R. Mi, H. Jiang, and Z. Zhang, "Lattice reduction meets key-mismatch: New misuse attack on lattice-based nist candidate kems," *Cryptology ePrint Archive*, 2022.

[30] D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi, "Lwe with side information: attacks and concrete security estimation," in *Annual International Cryptology Conference*. Springer, 2020, pp. 329–358.