Multiple-Valued Plaintext-Checking Side-Channel Attacks on Post-Quantum KEMs

Yutaro Tanaka[1,2], Rei Ueno[1,2], Keita Xagawa[3], Akira Ito[3], Junko Takahashi[3] and Naofumi Homma[1,2]

[1] Tohoku University, 2–1–1 Katahira, Aoba-ku, Sendai-shi, 980-8577, Japan
yutaro.tanaka.t6@dc.tohoku.ac.jp, rei.ueno.a8@tohoku.ac.jp, naofumi.homma.c8@tohoku.ac.jp
[2] CREST, JST, 4–1–8 Honcho, Kawaguchi, Saitama, 332-0012, Japan
[3] Social Informatics Laboratories, NTT Corporation, 3–9–11 Midori-cho, Musashino-shi, Tokyo, 180-8535, Japan
keita.xagawa.zv@hco.ntt.co.jp, akira.ito.as@hco.ntt.co.jp, junko.takahashi.fc@hco.ntt.co.jp

Abstract. In this paper, we present a side-channel analysis (SCA) on key encapsulation mechanisms (KEMs) based on the Fujisaki–Okamoto (FO) transformation and its variants. Many post-quantum KEMs usually perform re-encryption during key decapsulation to achieve chosen-ciphertext attack (CCA) security. The side-channel leakage of re-encryption can be exploited to mount a key-recovery plaintext-checking attack (KR-PCA), even if the chosen-plaintext attack (CCA) secure decryption constructing the KEM is securely implemented. Herein, we propose an efficient side-channel-assisted KR-PCA on post-quantum KEMs, and achieve a key recovery with significantly fewer attack traces than existing ones in TCHES 2022 and 2023. The basic concept of the proposed attack is to introduce a new KR-PCA based on a multiple-valued (MV-)PC oracle and then implement a dedicated MV-PC oracle based on a multi-classification neural network (NN). The proposed attack is applicable to the NIST PQC selected algorithm Kyber and the similar lattice-based Saber, FrodoKEM and NTRU Prime, as well as SIKE. We also present how to realize a sufficiently reliable MV-PC oracle from NN model outputs that are not 100% accurate, and analyze the tradeoff between the key recovery success rate and the number of attack traces. We assess the feasibility of the proposed attack through attack experiments on three typical symmetric primitives to instantiate a random oracle (SHAKE, SHA3, and AES software). The proposed attack reduces the number of attack traces required for a reliable key recovery by up to 87% compared to the existing attacks against Kyber and other lattice-based KEMs, under the condition of 99.9999% success rate for key recovery. The proposed attack can also reduce the number of attack traces by 85% for SIKE.

Keywords: Side-channel analysis · Fujisaki–Okamoto transformation · Key encapsulation mechanism · Public key encryption · Post-quantum cryptography · Deep learning

1 Introduction

1.1 Background

Post-quantum cryptography (PQC) has been actively studied in recent years. To construct PQC, a public key encryption scheme (PKE) with chosen-plaintext attack (CPA)

security is first developed, and then a key encapsulation mechanism (KEM) with chosen-ciphertext attack (CCA) security is obtained by combining the PKE with either the Fujisaki–Okamoto (FO) transform [FO99] or its variants. Thus, the re-encryption of the FO(-like) transform plays an essential role in the decapsulation of PQC.

In practical uses of PQC, security evaluation against implementation attacks, such as side-channel attacks, in addition to mathematical cryptanalysis attacks is inevitable. Side-channel attacks on PQC have been first studied on the PKE decryption part that exploits the secret key directly, similar to those on the modular exponentiation/scalar multiplication in RSA/elliptic curve cryptography (ECC). In recent years, another attack aspect has been studied [GTN20,RRCB20,UXT+21], in which the attacker focuses on the leakage of re-encryption to implement a *decryption oracle*, which enables one to mount a chosen-ciphertext attack (CCA) on the underlying CPA-secure PKE. Note that the decryption oracle is not necessarily a full decryption oracle but it can be, for example, plaintext-checking (PC) and decryption-failure (DF) oracles. These attacks suggest that we should protect not only the PKE decryption but also the whole KEM decapsulation, including re-encryption and the equality/validity checks of the re-encrypted ciphertext, against side-channel attacks. Ueno et al. [UXT+21] showed that such an attack was generally applicable to many post-quantum KEMs equipped with an FO-like transforms, and demonstrated that their attack could recover eight of the nine KEM candidates in the third round of the NIST PQC standardization.

These studies suggest that the potential and limitation of such attacks (i.e., the least number of side-channel traces for a successful key recovery) should be investigated to develop secure KEM implementations, including the design of countermeasures and/or cryptographic protocols. The number of side-channel traces required for a successful attack is commonly determined by two factors: (1) the number of decryption oracle accesses required for key recovery and (2) the number of traces required to realize the reliable decryption oracle. Therefore, a tight evaluation of the factors, that is, an efficient key-recovery algorithm with a decryption oracle that can be realized from side-channel traces, would contribute to understanding the least number of traces (i.e., the optimal attack cost) for a successful key recovery.

1.2 Our contribution

In this paper, we present an efficient side-channel-assisted key-recovery plaintext-checking attack (KR-PCA) by focusing on the re-encryption of an FO(-like) transformation, which is derived as an extension of the attack in [UXT+21]. The basic concept of the proposed attack is twofold: first, key-recovery attacks are presented using a multiple-valued plaintext-checking (MV-PC) oracle, which is a generalization of a binary PC oracle used in, for example, [RRCB20,BDH+21,UXT+21]; second, the MV-PC oracle is implemented with a multi-classification neural network (NN) from side-channel traces. Essentially, our 2^N-valued PC oracle ($N \in \mathbb{N}$) provides at most N-bit information of a secret key per access for lattice-based KEMs, while conventional binary PC oracles provide at most one bit per access.

In general, the design of a μ-classification ($\mu > 2$) is more difficult than that of a binary classification. Accordingly, the accuracy of an MV-PC oracle can be worse than a binary oracle, which leads to an increase in the number of traces required to realize a *reliable* PC oracle. If this increase is larger than the decrease in the number of oracle accesses, an attack using an MV-PC oracle has no advantage compared with conventional attacks using a binary PC oracle. To avoid such a large increase, we introduce a μ-valued NN to efficiently implement an accurate MV-PC oracle. Note that our proposal includes how to learn an NN model that can be used to implement the proposed key-recovery attack using the MV-PC oracle. We then evaluate the accuracy enhancement of an MV-PC oracle realized using multiple traces and discuss its information-theoretic aspects. Our proposed

Table 1: Number of attack traces required for successful key recovery for NIST PQC KEM standard/candidates with NIST security level 1 using NLL based distinguisher

		Ueno et al. [UXT+21]	Qin et al. [QCZ+21]	Revi et al. [RRD+22]	**This work**	Maximum reduction rate
Lattice	Kyber	3,072	2,622	190 / 950†	**576**	81% / 78% / 61%
	Saber	6,144	2,952	N/A	**1,920**	69% / 35.0%
	FrodoKEM	51,200	36,720	N/A	**9,600**	81% / 77%
	NTRU	5,608	N/A	N/A	5,608	0%
	NTRU LPRime	3,406	N/A	N/A	**642**	81%
Code	HQC	36,222	N/A	N/A	36,222	0%
	BIKE	6 M	N/A	N/A	6 M	0%
	Classic McEliece	Unknown	Inapplicable	Inapplicable	Inapplicable	Inapplicable
Isogeny	SIKE	548	N/A	N/A	**111**	80%

†Left value is derived from their paper [RRD+22], while right value is due to our reproduced experiment in Section 5 using SHA3 implementation with L2 norm based distinguisher of $N = 8$, as Kyber's re-encryption uses SHA3.

attack achieves a significant reduction in the number of side-channel traces required for successful key recovery compared to conventional attacks using a binary PC oracle and binary-classification NN.

For experimental validation, we applied the proposed attack to the NIST PQC standard and third-round candidates for KEMs. Table 1 summarizes the least numbers of attack traces required for successful key recovery in our experiments using both real devices and open-source implementations.

For comparison, Table 1 also shows the numbers of attack traces required for conventional attacks in [UXT+21] (worst case), [QCZ+21] (average case)[1], and [RRD+22] as state-of-the-art counterparts, where "N/A" indicates that their attack is not described. In addition, Table 1 evaluates their corresponding reduction rates by our attack. For each key-recovery attack, we derived the number of attack traces from the accuracy of the PC oracle implemented with an NN in our experiment (See Section 5).

Table 1 confirms that the proposed attack reduces the number of attack traces by a maximum of 81% for lattice-based PQ KEMs except for NTRU although it does not reduce those for NTRU and code-based KEMs due to the difficulty in the key-recovery attack using an MV-PC oracle. To the best of the authors' knowledge, the proposed attack can recover the secret keys of Kyber, Saber, FrodoKEM, NTRU LPRime of NTRU Prime, and SIKE[2] with the smallest numbers of traces among the general power/EM side-channel attacks on re-encryption under the condition that real devices and measurements are used.

1.3 Paper organization

The remainder of this paper is organized as follows. In Section 2, we review KEMs that are based on the FO transform and previous SCAs on KEMs that focus on FO transforms. We also describe the previous most-generalized attack which was presented in [UXT+21]. In Section 3, we describe the proposed attack and its application to the NIST PQC standard and third-round candidates with theoretical evaluations. In Section 4, we present the neural side-channel distinguisher (i.e., MV-PC oracle) design for the proposed attack

[1]The previous study in [QCZ+21, Table 6] presented the expected numbers of queries used in KR-PCAs against lattice-based PQ KEMs. The expected numbers of traces in our table was computed by multiplying the expected number of queries with the number of traces to implement a PC oracle.

[2]SIKE is considered insecure currently as practical attacks on SIKE have been reported [CD23, MM22, Rob23, MMP+23]. However, we present and evaluate the application of the proposed attack to SIKE in order to validate the generality and applicability of the proposed attack. In addition, researchers are developing variants of SIKE to counter the Castryck–Decru attack (e.g., [FMP23]). The evaluation of the proposed attack on SIKE would contribute to developing secure implementation of isogeny-based KEMs.

Algorithm 1 CCA-secure KEM based on FO transform (KeyGen, Encaps, Decaps)

KeyGen

Input: 1^λ
Output: $\mathsf{sk}, \mathsf{pk}, s$
1: **Function** KeyGen(1^λ)
2: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$;
3: $s \leftarrow_\$ \mathcal{M}$;
4: **return** $(\mathsf{sk}, \mathsf{pk}, s)$;

Encaps

Input: pk
Output: c, k
1: **Function** Encaps(pk)
2: $m \leftarrow_\$ \mathcal{M}$;
3: $r \leftarrow \mathsf{G}(m[, \mathsf{pk}])$;
4: $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m; r)$;
5: $k \leftarrow \mathsf{H}(m, c)$;
6: **return** (c, k);

Decaps

Input: $c, \mathsf{sk}, \mathsf{pk}, s$
Output: k
1: **Function** Decaps($c, \mathsf{sk}, \mathsf{pk}, s$)
2: $m' \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, c)$;
3: $r' \leftarrow \mathsf{G}(m'[, \mathsf{pk}])$;
4: $c' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m'; r')$;
5: **if** $c = c'$ **then**
6: **return** $\mathsf{H}(m', c)$;
7: **else**
8: **return** $\mathsf{H}_{\mathsf{prf}}(s, c)$;

and discuss its information-theoretic aspects. In Section 5, we present our experimental validation using real devices and open-source implementation of symmetric primitives compatible with PQCs. Finally, Section 6 concludes this paper.

2 Related Works

2.1 IND–CCA secure KEM based on the FO transform

A KEM is a public-key cryptographic primitive used to transmit a secret key securely. A KEM consists of three probabilistic polynomial-time algorithms: key generation (KeyGen), encapsulation (Encaps), and decapsulation (Decaps). Most post-quantum KEMs are proven to be CCA secure since they adopt an FO-like transform. Here, we refer to FO transforms and their variants [HHK17, SXY18, BHH+19] as FO-like transforms. Algorithm 1 illustrates a typical post-quantum KEM equipped with an underlying PKE scheme and an FO-like transform, assuming that the PKE is CPA secure and is given as three probabilistic polynomial-time algorithms: key generation Gen, encryption Enc and decryption Dec. Such KEMs employ random oracles (ROs) denoted by G, H, and $\mathsf{H}_{\mathsf{prf}}$ in Algorithm 1, which are frequently realized using a cryptographic hash function (or other symmetric primitive), and are frequently instantiated with SHA-3 or SHAKE.

The attacks considered herein focus on the decapsulation KEM.Decaps, which computes the shared secret as a result of H at Line 6 from an input ciphertext c, a private key sk and a public key pk (if the input ciphertext is valid). KEM.Decaps first applies the PKE decryption PKE.Dec to compute the corresponding plaintext m' from the ciphertext and then performs a re-encryption to validate the computed m', that is, computes PKE.Enc in the same manner as KEM.Encaps to check whether the re-encrypted ciphertext c' equals to the input ciphertext c. If $c = c'$, the input ciphertext is considered valid, and the shared secret $\mathsf{H}(m', c)$ is then calculated and output. Otherwise (i.e., $c \neq c'$), the ciphertext is invalid, and a pseudorandom value computed using $\mathsf{H}_{\mathsf{prf}}$ (or a rejection symbol) is output at Line 8. The FO-like transform performs the ciphertext verification to detect any invalid ciphertexts and prevent any CCA that queries invalid ciphertexts and exploits their decryption results.

2.2 Existing side-channel attacks on FO-like transforms

Side-channel attacks on the FO-like transforms were initially presented in the pioneering works by Guo et al. [GTN20] and Ravi et al. [RRCB20]. Guo et al. presented a timing attack exploiting the equality check (as at Line 5 in Algorithm 1). Ciphertexts of post-quantum KEMs are treated as a long vector in CPUs/microcontrollers. A comparison using an usual operation (e.g., `memcmp`) takes a (relatively) long time if two ciphertexts are very similar to each other; otherwise, the comparison terminates shortly. They exploited

this timing difference to implement a PC oracle for lattice- and code-based KEMs, and presented key-recovery attacks for lattice-based and code-based KEMs where the equality check was not implemented in constant time. Ravi et al. reported the first power/EM attack on an FO-like transform. The attack implemented a PC oracle or decryption failure oracle by exploiting the side-channel leakage during computation of RO (i.e., hash function) in re-encryption with a t-test-based template. Their attack achieved key recovery of six lattice-based KEMs: Kyber, Saber, FrodoKEM, Round5, NewHope and LAC.

Bhasin et al. [BDH+21] presented an attack on masked polynomial comparison schemes for Kyber, Saber, and FrodoKEM [OSPG18, BPO+20] by exploiting ciphertext equality checks in lattice-based KEMs and demonstrated its application to Kyber. They implemented a PC oracle using a distinguisher based on the t-test. More recently, Ueno et al. demonstrated a generalization of power/EM attacks on FO-like transforms [UXT+21]. They illustrated that the side-channel leakage during re-encryption can be *generally* exploited if a KR-PCA on the underlying PKE is known. They demonstrated that their attack could achieve key recovery of eight out of nine KEMs of NIST PQC third-round candidates.

One major research direction of attacks of this kind is to improve the attack efficiency (i.e., reduce the number of required traces) for a precise evaluation of the cost of an optimal attack. Other side-channel-assisted CCAs have also been extensively studied in this way [XPR+22, RBRC22, SKL+20, REB+21, NDGJ21], especially for lattice-based KEMs. More recently, in [QCZ+21], Qin et al. showed an improvement of CCAs on lattice-based KEMs using a binary key-mismatch oracle with adaptive queries, which reduced the number of oracle accesses/queries compared to that of the CCA used in [UXT+21]. In [SCZ+23], Shen et al. presented a side-channel-assisted CCA using a binary PC oracle with an error correction of the PC oracle outputs, implemented by side-channel information. They showed that the error tolerance reduced the number of traces required to implement a PC oracle, which reduced the total number of traces required for key recovery. They also applied their attack to Kyber, and achieved up to 55.4% reduction of the total number of traces required for key recovery compared to that in [UXT+21].

2.3 PC oracle implementation using neural side-channel distinguisher

First we formally describe a PC oracle. Let $(\mathsf{sk}, \mathsf{pk})$ denote a key pair of a KEM and let c be a valid ciphertext corresponding to a plaintext m. Here, "plaintext m" denotes the input to the PKE.Enc in KEM.Encaps (which corresponds the output of PKE.Dec in KEM.Decaps). We call m and c the reference plaintext and ciphertext, respectively. Let $\tilde{c}$ be an invalid ciphertext, which is a modification of c made by an attacker. When the attacker queries $\tilde{c}$, a PC oracle states whether $\tilde{c}$ is decrypted to m or not. Note that the attacker cannot obtain the decryption result of PKE.Dec (i.e., m' in Algorithm 1); they can only obtain the binary information. Formally, the PC oracle is defined as

$$\mathcal{O}(\tilde{c}; m) = \begin{cases} 1 & \text{if } \mathsf{PKE.Dec}_{\mathsf{sk}}(\tilde{c}) = m, \\ 0 & \text{otherwise.} \end{cases}$$

We refer to a key-recovery attack using the PC oracle as KR-PCA.

We now outline the side-channel-assisted KR-PCA of Ueno et al. [UXT+21], which is enhanced in this paper. Ueno et al. experimentally showed that an attacker can recover the secret key of many post-quantum KEMs vulnerable to KR-PCA if the attacker can implement the above PC oracle by utilizing the side-channel leakage. They also used the likelihood ratio test to realize a distinguisher to check whether $m' = m$ from the side-channel traces (i.e., a side-channel distinguisher). Their PC oracle is formally given in the following.

Unless defined otherwise, an uppercase character (e.g., X) denotes a random variable/vector of a set denoted by the calligraphic character (e.g., $\mathcal{X}$), and a lowercase character (e.g., x) denotes an element of that set (i.e., $x \in \mathcal{X}$). Let Pr be the probability measure and let p be the probability mass or density function. The conditional probability of $Y = y$ given $X = x$ is defined as $p_{Y|X}(y \mid x) = p_{Y,X}(y, x)/p_X(x)$. Let $\mathbb{E}$ denote the expectation. A side-channel trace is defined as $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^{\omega}$, where ω denotes the number of sample points.

Let B be a random variable that represents the oracle output (i.e., $B = \mathcal{O}(\tilde{C}; m)$ and $b \in \mathcal{B} = \{0, 1\}$). Let $p_{B|\boldsymbol{X}}$ be the true conditional probability distribution of the PC oracle output B given the side-channel trace $\boldsymbol{X}$, that is, $p_{B|\boldsymbol{X}}(1 \mid \boldsymbol{X}) = \Pr(M' = m \mid \boldsymbol{X})$ and $p_{B|\boldsymbol{X}}(0 \mid \boldsymbol{X}) = \Pr(M' \neq m \mid \boldsymbol{X})$. According to the Neyman–Pearson lemma [NP33], if the attacker knows the true distribution $p_{B|\boldsymbol{X}}$, the attacker can perform the most powerful test for $B = 1$ or 0 (i.e., $M = m$ or $M \neq m$) given (multiple copies of) $\boldsymbol{X}$. Let t be the number of traces available for one PC oracle implementation. The attacker queries an invalid ciphertext $\tilde{C}$ repeatedly t times to obtain t side-channel traces $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_i, \ldots, \boldsymbol{X}_{t-1}$. Ueno et al. proposed determining the PC oracle output $\hat{B}$ as follows:

$$\hat{B} = \operatorname*{arg\,max}_{b \in \{0,1\}} \log \prod_{i=0}^{t-1} p_{B|\boldsymbol{X}}(b \mid \boldsymbol{X}_i) = \operatorname*{arg\,max}_{b \in \{0,1\}} \sum_{i=0}^{t-1} \log p_{B|\boldsymbol{X}}(b \mid \boldsymbol{X}_i). \tag{1}$$

The Neyman–Pearson lemma guarantees that this is the most powerful test, as Equation (1) is equivalent to the likelihood test ratio[3] in accordance with Bayes' theorem, supposing that $p_B(0) = p_B(1) = 1/2$ and $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{t-1}$ are independent and identically distributed.

The above true distribution is usually unavailable to an attacker/evaluator. Accordingly, Ueno et al. proposed the use of deep learning (DL) to imitate $p_{B|\boldsymbol{X}}$. Let $q_\theta(b \mid \boldsymbol{x}) = q_{B|\boldsymbol{X}}(b \mid \boldsymbol{x}; \theta)$ be the conditional probability distribution represented by an NN with a parameter θ. In typical DL, an NN q_θ is trained such that the cross entropy (CE) is minimized. The goal of DL is to find an optimal parameter $\hat{\theta}$ using a dataset containing labeled side-channel traces. The CE is defined as

$$\mathrm{CE}(q_\theta) = -\mathbb{E} \log q_{B|\boldsymbol{X}}(B \mid \boldsymbol{X}; \theta) = -\int \sum_{b \in \{0,1\}} p_{B,\boldsymbol{X}}(b, \boldsymbol{x}) \log q_{B|\boldsymbol{X}}(b \mid \boldsymbol{x}; \theta)\, d\boldsymbol{x},$$

which is minimized if and only if $p = q_\theta$ (although it is not guaranteed that there exists such a θ according to $p_{B|\boldsymbol{X}}$ and the hyperparameters). As the CE is usually incomputable in practice due to the expectation (i.e., integral), it is approximated by the negative log likelihood (NLL) with a finite number of traces, defined as follows:

$$L(q_\theta) = -\frac{1}{s} \sum_{j=0}^{z-1} \log q_{B|\boldsymbol{X}}(B_j \mid \boldsymbol{X}_j; \theta),$$

where z denotes the number of traces in the dataset, and B_j denotes the j-th PC oracle output (i.e., label) corresponding to the j-th trace $\boldsymbol{X}_j$ in the dataset.

Ueno et al. [UXT+21] showed that such an NN achieved a sufficiently high accuracy to implement a PC oracle, even for a masked software implementation [git21]. Moreover,

[3] Let $\boldsymbol{X}^t$ denote $(\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{t-1})$. The likelihood ratio test in this case originally estimates whether the parameter is 1 or 0 (i.e., $b = 1$ or $b = 0$) by comparing $p_{\boldsymbol{X}^t|B}(\boldsymbol{X}^t \mid 1)$ and $p_{\boldsymbol{X}^t|B}(\boldsymbol{X}^t \mid 0)$. By supposing that $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{t-1}$ are independent and identically distributed, it holds that $p_{\boldsymbol{X}^t|B}(\boldsymbol{X}^t \mid b) = \prod_{i=0}^{t-1} p_{\boldsymbol{X}|B}(\boldsymbol{X}_i \mid b) = \prod_{i=0}^{t-1} p_{B|\boldsymbol{X}}(b \mid \boldsymbol{X}_i) p_{\boldsymbol{X}}(\boldsymbol{X}_i)/p_B(b)$ in accordance with Bayes' theorem. By supposing that $p_B(0) = p_B(1) = 1/2$, this equation is followed by $\arg\max_b p_{\boldsymbol{X}^t|B}(\boldsymbol{X}^t \mid b) = \arg\max_b \prod_{i=0}^{t-1} p_{B|\boldsymbol{X}}(b \mid \boldsymbol{X}_i) = \arg\max_b \log \sum_{i=0}^{t-1} p_{B|\boldsymbol{X}}(b \mid \boldsymbol{X}_i)$.

they experimentally demonstrated that the likelihood ratio test using a trained NN could achieve a higher success rate and fewer traces compared to a majority voting. This led to a key recovery of post-quantum KEMs with a practical total number of traces.

3 Proposed Attack

3.1 Multiple-valued plaintext-checking oracle

The major drawback of CCAs with the PC oracle outlined in Section 2.3 is that the attacker obtains no more than one-bit information per oracle access[4], which results in a large number of traces being required for key recovery. In this study, we propose an oracle named the *multiple-valued plaintext-checking (MV-PC) oracle* to extract more bits of information per oracle access.

Let μ be a positive integer. Consider an attacker who knows that a ciphertext $\tilde{c}$ is necessarily decrypted to either of the μ plaintexts $m_0, m_1, \ldots, m_{\mu-1}$. The attacker can recover the secret key of the KEM with repeated and adaptive queries if the attacker knows to which plaintext $\tilde{c}$ is decrypted. Such an oracle is a generalization of the (binary) PC oracle with $\mu = 2$. Note that, without side-channels, the FO-like transform theoretically does not provide any information about which plaintext $\tilde{c}$ corresponds to (although the attacker knows the reference plaintext for valid ciphertext). We name this oracle the MV-PC oracle, and formally define it as

$$\mathcal{O}_\mu(\tilde{c}; m_0, m_1, \ldots, m_{\mu-1}) = v \quad \text{s.t.} \quad \mathsf{PKE.Dec}_{\mathsf{sk}}(\tilde{c}) = m_v.$$

Intuitively, the MV-PC oracle for μ values (called μV-PC) provides up-to $\log_2 \mu$ information to the attacker; therefore, a CCA using the MV-PC oracle would achieve key recovery with fewer oracle accesses roughly by a factor of $1/\log_2 \mu$ than that of the conventional KR-PCA with the binary PC oracle.

3.2 General attack description

The proposed attack exploits an MV-PC oracle through a side-channel leakage and recovers the secret key by side-channel-assisted CCA on the underlying CPA-secure PKE. As well as the previous attack in [UXT+21], the proposed attack focuses on the side-channel leakage during the RO evaluation (i.e., computation of symmetric primitive like hash function) of re-encryption in PKE.Decaps. Using the side-channel leakage, the attacker accesses to an MV-PC oracle; that is, estimates which an invalid ciphertext $\tilde{c}$ is decrypted to plaintext $m_0, m_1, \ldots, m_{\mu-1}$ by PKE.Dec. If the attacker can correctly estimate the plaintext from the side-channel leakage, the attacker can perform a key recovery CCA. The proposed attack achieves a key recovery with fewer side-channel traces, if the MV-PC oracle is efficiently implemented.

As in [UXT+21], the proposed SCA consists of profiling and attack phases. In the profiling phase, the attacker trains a μ-classification NN to estimate the plaintext (i.e., implement an μV-PC oracle) from side-channel traces during the RO evaluation. In the attack phase, the attacker queries invalid ciphertexts and performs a CCA using the μV-PC oracle implemented by the trained NN. Although the attack requires a profiling, we do not require any profiling device which the attacker can know/control the secret key, in contrast to major DL-based side-channel attacks on symmetric primitive. This is because the profiling can be carried out without secret key, as in previous SCAs on lattice-based KEMs [RRCB20, XPR+22, RBRC20, SKL+20, NDGJ21, UXT+21]. The usage of NN makes the proposed attack general and efficient, because it has been shown that

[4]For lattice-based KEMs, the expected amount of information is far less than one bit, depending on its encoding method.

DL-based side-channel attacks are applicable to various implementations (including ones with countermeasures such as masking and random delay) without detailed knowledge of implementation nor a specific condition/assumption about leakage. Moreover, the proposed attack can be carried out by the neural distinguisher from side-channel leakage during the RO evaluation, and the key recovery capability depends solely on the quality of the neural distinguisher.

3.3 KR-MV-PCA algorithms for Kyber

We here consider Kyber as it is the only KEM selected from the finalists in Round 3. See Appendix A for other candidate KEMs in Round 3. Fist, we briefly review Kyber, and then extend the existing KR-PCA to KR-MV-PCA. Since we are interested in the decryption mechanism herein, we describe on the form of ciphertext of Kyber and its decryption.

Review of Kyber: Kyber involves several parameters, namely, $n = 256$, k, $q = 3329$, η_1, d_U, and d_V. The parameter sets of Kyber in Round 3 are summarized in Table 2.

Table 2: Parameter sets of Kyber in Round 3

parameter sets	n	k	q	η_1	d_U	d_V
Kyber512	256	2	3329	3	10	4
Kyber768	256	3	3329	2	10	4
Kyber1024	256	4	3329	2	11	5

For an odd positive integer α, $r' = r \mod \alpha$ is represented as the unique element r' in the range $-(\alpha-1)/2 \le r' \le (\alpha-1)/2$ satisfying $r' \equiv r \pmod{\alpha}$. For an even positive integer α, we define $r' = r \bmod^+ \alpha$ to be the unique element r' in the range $0 \le r' < \alpha$ satisfying $r' \equiv r \pmod{\alpha}$. Let $\mathcal{R} = \mathbb{Z}[x]/(x^{256}+1)$. Let $\mathcal{R}_q = \mathbb{Z}[x]/(x^{256}+1, q)$, where each coefficient is in $[-(q-1)/2, (q-1)/2]$ for odd q and $[-q/2+1, q/2]$ for even q. For $a \in \mathbb{Z}^+$, let $\mathcal{S}_a = \{f \in \mathcal{R} : f_i \in \{-a, -a+1, \ldots, a-1, a\} \text{ for } i = 0, \ldots, 255\}$, where f_i is the i-th coefficient of $f \in \mathcal{R}$. Let $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0) \in \mathcal{R}^k$ denote the i-th unit vector of $\mathcal{R}^k$ (i.e., a vector where the i-th element is 1 and other elements are 0). For a real number r, $\lceil r \rfloor$ denotes its nearest integer where ties are rounded up, that is, $\lceil r \rfloor = \lfloor r + 1/2 \rfloor$. We extend this notation to a polynomial and a vector of polynomials, in which each coefficient of the polynomial(s) are rounded to its nearest integer. For two vectors $\mathbf{a}$ and $\mathbf{b} \in \mathcal{R}^k$, let $\langle \mathbf{a}, \mathbf{b} \rangle$ denote their inner product, that is, $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{k} a_i b_i \in \mathcal{R}$.

A ciphertext of Kyber is given by $(\mathbf{c}_1, c_2) \in \mathcal{R}^k \times \mathcal{R}$, where coefficients of $\mathbf{c}_1$ are in the range $[0, 2^{d_U})$ and those of c_2 are in the range $[0, 2^{d_V})$. A secret key is denoted by $\mathbf{s} = (s_1, \ldots, s_k) \in \mathcal{S}_{\eta_1}^k$. The decapsulation algorithm first computes $M = \lceil (q/2^{d_V}) c_2 \rfloor - \langle \lceil (q/2^{d_U}) \mathbf{c}_1 \rfloor, \mathbf{s} \rangle \bmod q \in \mathcal{R}_q$ and obtains the plaintext $m' = \lceil (2M/q) M \rfloor \bmod 2 \in \mathcal{R}_2$.

Review of KR-PCA: We briefly review the KR-PCA against Kyber in Round 3 [HV20, XIU+21]. To identify the j-th coefficient of the i-th element s_i of $\mathbf{s}$, the attacker fixes $\mathbf{c}_1$ depending on i, modifies the j-th coefficient of c_2 as its invalid version $\tilde{c}_2$, and makes $\log_2(2\eta_1 + 1)$ queries to the PC oracle to check whether the decrypted plaintext is all zero vector $(0, \ldots, 0)$.

Suppose that we consider Kyber512 and want to determine $s_{i,j}$ (i.e., j-th coefficient of s_i for $j \in \{0, \ldots, 255\}$). We then consider a ciphertext

$$(\mathbf{c}_1, \tilde{c}_2) = \left(U \cdot \mathbf{e}_i, T(t) \cdot x^i \right),$$

where $U = \lceil (2^{d_U}/q) \cdot 276 \rfloor \bmod^+ 2^{d_U}$ and $T(t) = \lceil (2^{d_V}/q) \cdot 208 \cdot t \rfloor \bmod^+ 2^{d_V}$ with $t \in \{-3, -2, \ldots, 3\}$. We have the following lemma for this ciphertext. (We extend this result later.)

Table 3: KR-PCA: The behavior of m'_j of $m' = \mathsf{Dec}(\mathbf{s}, (\mathbf{c}_1, \tilde{c}_2))$ on a ciphertext $(\mathbf{c}_1, \tilde{c}_2)$ with $\mathbf{c}_1 = U \cdot \mathbf{e}_i$ and $\tilde{c}_2 = T(t) \cdot x^j$.

(a) Kyber512

$s_{i,j}$ \ t	-3	-2	-1	0	1	2	3
-3	1	1	1	0	0	0	0
-2	1	1	0	0	0	0	0
-1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
$+1$	0	0	0	0	0	0	1
$+2$	0	0	0	0	0	1	1
$+3$	0	0	0	0	1	1	1

(b) Kyber768 and Kyber1024

$s_{i,j}$ \ t	-3	-2	-1	0	1	2	3
-2	1	1	0	0	0	0	0
-1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
$+1$	0	0	0	0	0	0	1
$+2$	0	0	0	0	0	1	1

Lemma 1 ([HV20, XIU+21]). *For the ciphertext $(\mathbf{c}_1, \tilde{c}_2)$ defined in the above, let $m' = \mathsf{Dec}(\mathbf{s}, (\mathbf{c}_1, \tilde{c}_2))$. We have $m'_k = 0$ for $k \in \{0, \ldots, 255\} \setminus \{j\}$ and*

$$m'_j = 0 \iff |276 \cdot s_{i,j} + 208 \cdot t| \leq 832.$$

See the pattern of m'_j in Table 3a for $s_{i,j} \in \{-3, \ldots, 3\}$ and $t \in \{-3, \ldots, 3\}$. Thus, by querying to the PC oracle three times with appropriately crafted ciphertexts, we can determine $s_{i,j}$ by a binary search.

A similar result holds for Kyber768 and Kyber1024 for the same parameter setting. See the pattern in Table 3b for $s_{i,j} \in \{-2, \ldots, 2\}$ and $t \in \{-3, \ldots, 3\}$. Again, querying the PC oracle three times, we can determine $s_{i,j}$ by a binary search.

KR-2^{N}V-PCA: Intuitively, we run the binary search of N *coefficients in parallel* by using the 2^NV-PC oracle. For the sake of the ease of implementation of the 2^NV-PC oracle, we designed a query invalid ciphertext such that the decrypted plaintext varies in the first N positions of the plaintexts and is fixed as 0 in the other remaining positions by modifying the existing KR-PCA.

For $a \in \{0, N, \ldots, 256 - N\}$, suppose that we want to determine $s_{i,a+j}$ for all $j = 0, \ldots, N-1$, which means N sequential coefficients of s_i starting from its a-th coefficient. We here consider a ciphertext

$$(\mathbf{c}_1, \tilde{c}_2) = \left((U \cdot x^{256-a})\mathbf{e}_i, \sum_{\ell=0}^{N-1} T(t_\ell) \cdot x^\ell \right), \tag{2}$$

where U and T are the same value and function as KR-PCA and $t_0, \ldots, t_{N-1} \in \{-3, -2, \ldots, 3\}$. The decryption algorithm first computes $M = \lceil (q/2^{d_V})\tilde{c}_2 \rfloor - \langle \lceil (q/2^{d_U})\mathbf{c}_1 \rfloor, \mathbf{s} \rangle \bmod q \in \mathcal{R}_q$. Thus, we have

$$\begin{aligned} M &= \lceil (q/2^{d_V})\tilde{c}_2 \rfloor - \langle \lceil (q/2^{d_U})\mathbf{c}_1 \rfloor, \mathbf{s} \rangle \bmod q \\ &= \left\lceil (q/2^{d_V}) \sum_{\ell=0}^{N-1} T(t_\ell) \cdot x^\ell \right\rfloor - \langle \lceil (q/2^{d_U})(U \cdot x^{256-a})\mathbf{e}_i \rfloor, \mathbf{s} \rangle \bmod q \\ &= \sum_{\ell=0}^{N-1} \lceil (q/2^{d_V})T(t_\ell) \rfloor \cdot x^\ell - \lceil (q/2^{d_U})U \rfloor \cdot x^{256-a} s_i \bmod q. \end{aligned}$$

Recall that $U = \lceil (2^{d_U}/q) \cdot 276 \rfloor \bmod^+ 2^{d_U}$ and $T(t) = \lceil (2^{d_V}/q) \cdot 208 \cdot t \rfloor \bmod^+ 2^{d_V}$ with $t \in \{-3, -2, \ldots, 3\}$.

Table 4: KR-2^NV-PCA: Behavior of m'_j for $j = 0, \ldots, N-1$ of $m' = \mathsf{Dec}(\mathbf{s}, (\mathbf{c}_1, \tilde{c}_2))$ on a ciphertext $(\mathbf{c}_1, \tilde{c}_2) = ((U \cdot x^{256-a}) \cdot \mathbf{e}_i, \sum_{\ell=0}^{N-1} T(t_\ell) x^\ell)$.

(a) Kyber512

$s_{i,a+j}$ \ t_j	−3	−2	−1	0	1	2	3
−3	1	1	1	0	0	0	0
−2	1	1	0	0	0	0	0
−1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
+1	0	0	0	0	0	0	1
+2	0	0	0	0	0	1	1
+3	0	0	0	0	1	1	1

(b) Kyber768 and Kyber1024

$s_{i,a+j}$ \ t_j	−3	−2	−1	0	1	2	3
−2	1	1	0	0	0	0	0
−1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
+1	0	0	0	0	0	0	1
+2	0	0	0	0	0	1	1

Let us consider the first term: For $q = 3329$ and $d_V \in \{4, 5\}$ in the parameter sets of Kyber, we also have, for $t \in \{-3, -2, \ldots, 3\}$,

$$\left\lceil (q/2^{d_V}) T(t) \right\rfloor = \left\lceil (q/2^{d_V}) \cdot \left(\left\lceil (2^{d_V}/q) \cdot 208 \cdot t \right\rfloor \bmod^+ 2^{d_V} \right) \right\rfloor = 208 \cdot t.$$

Hence, we have

$$\sum_{\ell=0}^{N-1} \left\lceil (q/2^{d_V}) T(t_\ell) \right\rfloor \cdot x^\ell = \sum_{\ell=0}^{N-1} 208 t_\ell \cdot x^\ell.$$

We next consider the second term: For $q = 3329$ and $d_U \in \{10, 11\}$ in the parameter sets of Kyber, we have

$$\left\lceil (q/2^{d_U}) U \right\rfloor = \left\lceil (q/2^{d_U}) \cdot (\left\lceil (2^{d_U}/q) \cdot 276 \right\rfloor \bmod^+ 2^{d_U}) \right\rfloor = 276.$$

We then have

$$\begin{aligned} \left\lceil (q/2^{d_U}) U \right\rfloor x^{256-a} s_i = 276 \cdot x^{256-a} s_i = 276 \cdot x^{256-a} \sum_{\ell=0}^{255} s_{i,\ell} x^\ell = 276 \sum_{\ell=0}^{255} s_{i,\ell} x^{256+\ell-a} \\ = -276 \sum_{\ell=0}^{255-a} s_{i,\ell+a \bmod^+ 256}\, x^\ell + 276 \sum_{\ell=255-a+1}^{255} s_{i,\ell+a \bmod^+ 256}\, x^\ell, \end{aligned}$$

followed by $x^{256+b} = -x^b$ for $b \in \{0, \ldots, 255\}$ since $\mathcal{R} = \mathbb{Z}[x]/(x^{256}+1)$ and $\mathcal{R}_q = \mathbb{Z}[x]/(x^{256}+1, q)$.

Combining those two calculations, for $a \in \{0, N, \ldots, 256 - N\}$, we obtain

$$M_j = \begin{cases} 276 \cdot s_{i,a+j \bmod^+ 256} + 208 \cdot t_j & (j = 0, \ldots, N-1) \\ \pm 276 \cdot s_{i,a+j \bmod^+ 256} & (\text{otherwise}). \end{cases}$$

The decryption algorithm then computes $m' = \lceil (2/q) M \rfloor \bmod 2$. For $j = 0, \ldots, N-1$, M_j is decoded into 0 if and only if $|M_j| \le 832 \approx q/4$, that is, $|276 s_{i,a+j} + 208 t_j| \le 832$. For $j = N, \ldots, 256$, M_j is decoded into 0 since $|(2/q) \cdot 276 \cdot b| < 1/2$ for $b \in \{-\eta_1, \ldots, \eta_1\}$, where $\eta_1 = 2$ or 3. The pattern of m'_j is summarized in Table 4.

Hence, we can run a binary search on $s_{i,j} \in \{-\eta_1, \ldots, \eta_1\}$ *for N coefficients in parallel* and reduce the number of queries by a factor of approximately $1/N$. We estimate the upper bound of the number of oracle accesses to be $\lceil \log_2(2\eta_1 + 1) \rceil \cdot \lceil 256/N \rceil \cdot k = 3 \cdot \lceil 256/N \rceil \cdot k$ for all parameter sets of Kyber. The whole key-recovery algorithm for Kyber768 and Kyber1024 is summarized in Algorithm 2. Intuitively, a query with t_j tells that $s_{i,a+j} \ge t_j$ if $m'_j = 1$; otherwise, $s_{i,a+j} \le t_j - 1$. The key-recovery algorithm for Kyber512 is also obtained by modifying Threshold and Update appropriately.

Algorithm 2 KR-2^NV-PCA against Kyber768 and Kyber1024

Output: $\mathbf{s} = (s_1, \ldots, s_k) \in \mathcal{S}_{\eta_1}^k$

Function $\mathsf{KRA}(1^\lambda)$
  for $i = 0, \ldots, k-1$ **do**
    for $\ell = 0, \ldots, \lceil 256/N \rceil - 1$ **do** ▷ Find $s_{i,\ell N}, \ldots, s_{i,\ell N+N-1}$
      $\mathbf{c}_1 \leftarrow U \cdot x^{256-\ell N} \cdot \mathbf{e}_i$;
      $R_0, \ldots, R_{N-1} \leftarrow \{-\eta_1, \ldots, \eta_1\}$;
      for $i' = 0, \ldots, 2$ **do** ▷ Binary Search in Parallel
        for $j = 0, \ldots, N-1$ **do**
          $t_j \leftarrow \mathsf{Threshold}(R_j)$;
        $\tilde{c}_2 \leftarrow \sum_{j=0}^{N-1} T(t_j) x^j$;
        $v \leftarrow \mathcal{O}_{2^N}((\mathbf{c}_1, \tilde{c}_2); 0^N \| 0^{256-N}, \ldots, 1^N \| 0^{256-N})$; ▷ Let $m_v = v_0 \| \ldots \| v_{N-1} \| 0^{256-N}$
        Parse $v = (v_0, v_1, \ldots, v_{N-1}) \in \{0,1\}^N$;
        for $j = 0, \ldots, N-1$ **do**
          $R_j \leftarrow \mathsf{Update}(R_j, v_j)$;
      for $j = 0, \ldots, N-1$ **do**
        $s_{i,\ell N+j} \leftarrow r_j$ such that $R_j = \{r_j\}$;
  return $\mathbf{s}$;

Function $\mathsf{Threshold}(R)$
  if $R = \{-2,-1,0,+1,+2\}$ **then**
    return -3;
  else if $R = \{-2,-1\}$ **then**
    return -2;
  else if $R = \{0,+1,+2\}$ **then**
    return 3;
  else if $R = \{+1,+2\}$ **then**
    return 2;
  else if $|R| = 1$ **then**
    return 0;

Function $\mathsf{Update}(R, v)$
  if $R = \{-2,-1,0,+1,+2\}$ **then**
    return $\{-2,-1\}$ **if** $v = 1$ **else** $\{0,+1,+2\}$;
  else if $R = \{-2,-1\}$ **then**
    return $\{-2\}$ **if** $v = 1$ **else** $\{-1\}$;
  else if $R = \{0,+1,+2\}$ **then**
    return $\{+1,+2\}$ **if** $v = 1$ **else** $\{0\}$;
  else if $R = \{+1,+2\}$ **then**
    return $\{+2\}$ **if** $v = 1$ **else** $\{+1\}$;
  else if $|R| = 1$ **then**
    return R;

Example of KR-2^N-PCA: We show an example of KR-2^N-PCA on Kyber768 with $N = 4$. We here try to recover a 4-bit secret key $(s_{i,0}, s_{i,1}, s_{i,2}, s_{i,3}) = (-2, 0, 2, 1)$ using $\mathcal{O}_{2^N}$. For $j = 0, 1, 2, 3$, let $R_j \coloneqq \{-2, -1, 0, 1, 2\}$ be candidate sets of $s_{i,j}$. According to Equation (2) and $\mathsf{Threshold}$, we first generate a ciphertext $(\mathbf{c}_1, \tilde{c}_2)$ with $(t_0, t_1, t_2, t_3) = (-3, -3, -3, -3)$. Querying $\mathcal{O}_{2^N}$ with it, we know $(m'_0, m'_1, m'_2, m'_3) = (1, 0, 0, 0)$ for this case. From this result, we can reduce the possible range of secret key coefficients as $s_{i,0} \leq -1$ and $s_{i,1}, s_{i,2}, s_{i,3} \leq 0$ and we have $R_0 = \{-2, -1\}$ and $R_1 = R_2 = R_3 = \{0, +1, +2\}$. Then, to further reduce the range of $s_{i,j}$, we query $\mathcal{O}_{2^N}$ with another ciphertext where $(t_0, t_1, t_2, t_3) = (-2, 3, 3, 3)$, and it tells that $(m'_0, m'_1, m'_2, m'_3) = (1, 0, 1, 1)$. This indicates that $s_{i,0} \leq -2$ and $s_{i,1} \geq 0$ as well as $s_{i,2}, s_{i,3} \leq 1$, and we have that $R_0 = \{+2\}$, $R_1 = \{0\}$, and $R_2 = R_3 = \{+1, +2\}$. Thus, we know $s_{i,0} = -2$ and $s_{i,1} = 0$, yet we cannot identify other coefficients. Finally, we query $\mathcal{O}_{2^N}$ with third ciphertext where $(t_0, t_1, t_2, t_3) = (0, 0, 2, 2)$, and it tells that $(m'_0, m'_1, m'_2, m'_3) = (0, 0, 1, 0)$. This indicates that $s_{i,2} \geq 2$ and $s_{i,3} \leq 1$ and we have that $R_2 = \{+2\}$ and $R_3 = \{+1\}$. In consequence, we identify the secret key as $(s_{i,0}, s_{i,1}, s_{i,2}, s_{i,3}) = (-2, 0, 2, 1)$ from the inequalities. Thus, we can identify N coefficients in parallel with 3 queries, while the conventional binary KR-PCA only identifies one coefficient with 3 queries. We achieve a full-key recovery by repeating this procedure for all i and j.

3.4 Complexity analysis

Table 5 details the numbers of oracle accesses required for the KR-μV-PCA on lattice-based KEMs and SIKE, and Table 6 lists the concrete values when $N = 1, 2, \ldots, 8$. These tables show the values for schemes with the security equivalent of AES128 and AES256 (i.e., NIST security levels 1 and 5, respectively). These tables show that an increase in N (i.e., the usage of the MV-PC oracle) significantly contributes to decreasing the

Table 5: Formulas for deriving the number of oracle accesses required for KR-μV-PCA ($\mu = 2^N$ for lattice-based KEMs and $\mu = 3^N$ for SIKE)

KEM type	Scheme	Instance	# Oracle accesses
Lattice	Kyber	Kyber-512	$3 \times \lceil 256/N \rceil \times 2$
		Kyber-1024	$3 \times \lceil 256/N \rceil \times 4$
	Saber	LightSaber-KEM	$N = 1$: at most $4 \times 256 \times 2 + 2 \times 256 \times 2$ $N \geq 2$: at most $8 \times \lceil 256/N \rceil \times 2 + 2 \times \lceil 256/N \rceil \times 2$
		FireSaber-KEM	$3 \times \lceil 256/N \rceil \times 4$
	FrodoKEM	FrodoKEM-640	$5 \times \lceil 640 \times 8/N \rceil$
		FrodoKEM-1344	$4 \times \lceil 1344 \times 8/N \rceil$
	NTRU Prime	ntrulpr653	$2 + 3 \times \lceil (653 - N)/N \rceil$
		ntrulpr1277	$2 + 3 \times \lceil (1277 - N)/N \rceil$
Isogeny	SIKE	SIKEp434	$\lceil 274 \times 2/(3N) \rceil$
		SIKEp751	$\lceil 478 \times 2/(3N) \rceil$

Table 6: Number of oracle accesses required for KR-MV-PCA when $N = 1, 2, \ldots, 8$ ($\mu = 2^N$ for lattice-based KEMs and $\mu = 3^N$ for SIKE)

KEM type	Scheme	Instance	# Oracle accesses							
		$N =$	1	2	3	4	5	6	7	8
Lattice	Kyber	Kyber-512	1,536	768	516	384	312	258	222	192
		Kyber-1024	3,072	1,536	1,032	768	624	516	444	384
	Saber	LightSaber-KEM	3,072	2,560	1,720	1,280	1,040	860	740	640
		FireSaber-KEM	3,072	1,536	1,032	768	624	516	444	384
	FrodoKEM	FrodoKEM-640	25,600	12,800	8,535	6,400	5,120	4,270	3,660	3,200
		FrodoKEM-1344	43,008	21,504	14,336	10,752	8,604	7,168	6,144	5,376
	NTRU	ntruhrss701	2,804				N/A			
		ntruhps2048509	1,018				N/A			
		ntruhps4096821	1,642				N/A			
	NTRU Prime	ntrulpr653	1,703	853	571	428	344	287	245	214
		ntrulpr1277	3,575	1,789	1,195	896	719	599	512	448
		sntrup653	2,712				N/A			
		sntrup1277	5,175				N/A			
Isogeny	SIKE	SIKEp434	290	145	97	73	58	49	42	37
		SIKEp751	319	160	107	80	64	54	46	40

number of oracle accesses. Note that this value corresponds to the number of attack traces if we can implement the oracle with 100% accuracy using one side-channel trace; in practice, we need multiple traces to implement a reliable oracle. In other words, if we can implement the MV-PC oracle for greater N, we can perform key recovery on these KEMs very efficiently, as demonstrated in Section 5.

4 Neural Side-Channel Distinguisher for MV-PC Oracle

4.1 Basic concept

In this study, we propose the use of DL to implement an MV-PC oracle. Namely, we train a μ-classification NN to estimate which plaintext, $m_0, m_1, \ldots$, or $m_{\mu-1}$, corresponds to $\tilde{c}$ from the power/EM trace(s) during an RO evaluation. Note that a profiling dataset can be acquired using the target device without knowing the secret key, which is meaningful for the practicality of the proposed attack in a real scenario.

Let B_μ be a random variable representing a μ-valued PC oracle output (i.e., $B_\mu = \mathcal{O}_\mu(\tilde{C}; m_0, m_1, \ldots, m_{\mu-1})$ and $b \in \mathcal{B}_\mu = \{0, 1, \ldots, \mu - 1\}$). Let $p_{B_\mu|\boldsymbol{X}}$ be the true conditional probability distribution of B_μ, given the side-channel trace $\boldsymbol{X}$, that is, $p_{B_\mu|\boldsymbol{X}}(b \mid \boldsymbol{X}) = \Pr(M' = m_b \mid \boldsymbol{X})$. The goal of DL in the proposed attack is to imitate the true conditional distribution of B_μ given a side-channel trace $\boldsymbol{X}$ using an NN $q_\theta = q_{B_\mu|\boldsymbol{X}}(B_\mu \mid \boldsymbol{X}; \theta)$. Therefore, in the profiling phase, we train the NN to minimize $\mathrm{CE}(q_\theta)$ using a dataset acquired from the target device. To acquire a dataset containing t labeled side-channel traces $(B_\mu, \boldsymbol{X})$, the attacker computes a valid ciphertext corresponding to each

of $m_0, m_1, \ldots,$ and $m_{\mu-1}$ using KEM.Encaps and the public key, and queries them to the target device to measure its side-channel trace. Thus, NN training can be conducted using this dataset in the same manner as DL-based side-channel attacks on symmetric ciphers [BPS+20, KPH+19, PCP20, ZBHV20, WAGP20, ISUH21].

We now describe how to realize a reliable μ-class PC oracle using multiple traces, as it is quite difficult for a classification (even using NN) to achieve an accuracy as high as 100%. Let α_t be the resulting accuracy using multiple NN inferences for t side-channel traces. The number of traces for one MV-PC oracle implementation should be determined according to $\sigma \leq {\alpha_t}^u$, where u denotes the number of oracle accesses required for key recovery. As the most efficient method for a high α_t, we can use the likelihood ratio test as shown in [UXT+21]; we determine the oracle output $\hat{B}_\mu$ as

$$\hat{B}_\mu = \arg\max_{b \in \mathcal{B}_\mu} \log \prod_{i=0}^{t-1} q_{B_\mu|\boldsymbol{X}}(b \mid \boldsymbol{X}_i; \theta) = \arg\max_{b \in \mathcal{B}_\mu} \sum_{i=0}^{t-1} \log q_{B_\mu|\boldsymbol{X}}(b \mid \boldsymbol{X}_i; \theta). \tag{3}$$

As proven in Section 4.2, if $q_\theta = p$, then this method is optimal, which means that this method theoretically maximizes the success rate $\alpha_t = \Pr(\hat{B}_\mu = B_\mu)$. In Equation (3), $-\sum_{i=0}^{t-1} \log q_{B_\mu|\boldsymbol{X}}(b \mid \boldsymbol{X}_i; \theta)$ is the negative-log likelihood (NLL), which converges in probability to the cross entropy (CE) as $t \to \infty$.

One major drawback in using the NLL is its difficulty in evaluating the resulting accuracy in an analytical manner. As an easily available alternative, our method can employ a majority voting of multiple inference results to evaluate the lower bound of the number of attack traces required for key recovery. Using an NN with an accuracy of a, we can readily evaluate the resulting accuracy α_t by

$$\alpha_t \geq 1 - \sum_{s=0}^{\lceil t/2 \rceil} \binom{t}{s} a^s (1-a)^{t-s}, \tag{4}$$

because the majority voting result is always correct if more than $\lfloor t/2 \rfloor$ NN inference results are correct. Note that its converse does not necessarily hold for the majority voting of μ-classification if $\mu > 2$, and therefore α_t is evaluated by an inequality. To evaluate the value of α_t as an equality, we need to consider multinomial coefficients for the probabilities that all incorrect labels are inferred as correct; however, such an analysis is difficult in multi-class classification. Therefore, in addition to the NLL, we also use the inequality in Equation (4) for an analytical evaluation in this paper, as it can be readily evaluated using an NN accuracy, which is a common metric for NN.

4.2 Information-theoretic aspects of side-channel distinguisher

In this section, we first show the optimality of the likelihood ratio test with the true conditional probability distribution $p_{B_\mu|\boldsymbol{X}}$ in Equation (3) as Theorem 1. The theorem theoretically validates the usage of DL to implement an MV-PC oracle, as the goal of DL is to imitate the true conditional probability distribution.

Theorem 1 (Optimal distinguish rule for MV-PC oracle implementation)**.** *Let $p_{B_\mu|\boldsymbol{X}}$ be the true conditional probability distribution of a μ-valued PC oracle B_μ, given side-channel traces $\boldsymbol{X}$. Let $\hat{B}_\mu$ denote the attacker's guess of B_μ. Suppose that t side-channel traces $\boldsymbol{X}^t = (\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{t-1})$ are independent and identically distributed (i.i.d). A distinguish rule defined as*

$$\hat{B}_\mu = \arg\max_{b \in \mathcal{B}_\mu} \sum_{i=0}^{t-1} \log p_{B_\mu|\boldsymbol{X}}(b \mid \boldsymbol{X}_i),$$

is optimal, that is, maximizes the success rate of the guess $\Pr(\hat{B}_\mu = B_\mu)$.

Proof. Let $\ell(\hat{b}, b) = \mathbb{1}_{\{\hat{b} \neq b\}}$ be a loss function, where $\mathbb{1}$ denotes the indicator function. The optimal distinguish rule is defined as a rule that maximizes the success rate in guessing the correct b. Note that $\Pr(\hat{B}_\mu = B_\mu) = \mathbb{E}\mathbb{1}_{\{\hat{B}_\mu = B_\mu\}} = 1 - \mathbb{E}\ell(\hat{B}_\mu, B_\mu)$ holds. Therefore, we can maximize the success rate by minimizing the loss function $\mathbb{E}\ell(\hat{B}_\mu, B_\mu)$. To obtain the optimal distinguish rule, we rewrite the loss function as:

$$\mathbb{E}\ell(\hat{B}_\mu, B_\mu) = \mathbb{E}\mathbb{1}_{\{\hat{B}_\mu \neq B_\mu\}} = \mathbb{E}\mathbb{E}[1 - \mathbb{1}_{\{\hat{B}_\mu = B_\mu\}} \mid \boldsymbol{X}^t] = \mathbb{E}[1 - \Pr(\hat{B}_\mu = B_\mu \mid \boldsymbol{X}^t)].$$

Therefore, according to the i.i.d assumption,

$$\hat{B}_\mu = \arg\max_b \Pr(B_\mu = b \mid \boldsymbol{X}^t) = \arg\max_b \sum_{i=0}^{t-1} \log p_{B_\mu|\boldsymbol{X}}(b \mid \boldsymbol{X}_i)$$

is the optimal distinguish rule, because it minimizes the loss function. □

Theorem 1 validates the use of DL to imitate $p_{B_\mu|\boldsymbol{X}}$ in the attack, and implies that the NLL-based distinguihser achieves a high performance (i.e., high success rate in distinguishing the input) if NN can sufficiently imitate the true conditional probability distribution. We can evaluate an upper bound of the success rate or this optimal distinguish rule using Theorem 2, similar to the key-recovery side-channel attacks on symmetric ciphers [dCGRP19, IUH22].

Theorem 2 (Upper bound of success rate of MV-PC oracle implementation)**.** *Let $I(B_\mu \mid \boldsymbol{X})$ be the mutual information between a μ-valued PC oracle B_μ and side-channel traces $\boldsymbol{X}$. Assume that t side-channel traces $\boldsymbol{X}^t = (\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{t-1})$ are i.i.d. The optimal success rate of the μ-valued PC oracle implementation using t traces, denoted by α_t, is bounded as*

$$\xi(\alpha_t) \leq tI(B_\mu; \boldsymbol{X}), \tag{5}$$

where ξ is a function defined as

$$\xi(\alpha_t) = H(B_\mu) - (1 - \alpha_t)\log_2(\mu - 1) - H_2(\alpha_t),$$

and H_2 is the binary entropy function.

Proof. This is proven in a similar manner to [dCGRP19] using Fano's inequality [CT06, Theorem 2.10.1]. Let $H(X)$ be the entropy of a random variable X. Suppose that the side-channel attack is represented as a Markov chain of $B_\mu \leftrightarrow \boldsymbol{X} \leftrightarrow \hat{B}_\mu$, similar to [HRG14, dCGRP19]. Due to the definition of mutual information, we have

$$I(B_\mu; \boldsymbol{X}^t) = H(B_\mu) - H(B_\mu \mid \boldsymbol{X}^t).$$

As $\hat{B}_\mu$ is a function of $\boldsymbol{X}^t$, we have $H(B_\mu \mid \boldsymbol{X}^t) = H(B_\mu \mid \boldsymbol{X}^t, \hat{B}_\mu)$, and it follows that

$$\begin{aligned} I(B_\mu; \boldsymbol{X}^t) &= H(B_\mu) - H(B_\mu \mid \boldsymbol{X}^t, \hat{B}_\mu) \\ &\geq H(B_\mu) - H(B_\mu \mid \hat{B}_\mu). \end{aligned} \tag{6}$$

According to Fano's inequality [CT06, Theorem 2.10.1] on the Markov chain $B_\mu \leftrightarrow \boldsymbol{X} \leftrightarrow \hat{B}_\mu$, we have

$$H(B_\mu \mid \hat{B}_\mu) \leq H_2(\alpha_t) + (1 - \alpha_t)\log_2(|\mathcal{B}_\mu| - 1). \tag{7}$$

Combining Equation (6) and (7), we have

$$\xi(\alpha_t) = H(B_\mu) - (1 - \alpha_t)\log_2(\mu - 1) - H_2(\alpha_t) \leq I(B_\mu; \boldsymbol{X}^t).$$

Since it holds $I(B_\mu; \boldsymbol{X}^t) \leq tI(B_\mu; \boldsymbol{X})$ due to the i.i.d assumption, we conclude that it holds $\xi(\alpha_t) \leq tI(B_\mu; \boldsymbol{X})$, as required. □

Corollary 1. *Suppose that $H(B_\mu) = \log_2 \mu$. In the distinguish attack, to achieve a success rate of 1 (i.e., $\alpha_t = \Pr(\hat{B}_\mu = B_\mu) = 1$), the following should hold:*

$$t \geq \frac{\log_2 \mu}{I(B_\mu; \boldsymbol{X})}. \tag{8}$$

Proof. If we substitute $\alpha_t = 1$ and $H(B_\mu) = \log_2 \mu$ into the inequality in Equation (5), the proof follows. □

The upper bound of the success rate conversely represents a lower bound of the number of attack traces required to achieve a given success rate. The inequality in Equation (8) corresponds to a shortcut evaluation formula used in [ABH+22], although no proof was previously provided for this case. The upper bounds of the success rate based on Fano's inequality are meaningfully precise for the cases of key-recovery side-channel attacks [dC-GRP19, IUH22]; accordingly, the inequality in Equation (5) is expected to be helpful for a precise evaluation of the number of attack traces required to implement a reliable MV-PC oracle. The evaluation using Equation (5) is available if $I(B_\mu; \boldsymbol{X})$ is available, for example, if we assume that the noise is Gaussian distributed and additive (implying that $I(B_\mu; \boldsymbol{X})$ is determined by its variance) and if the evaluator performs profiling for the actual device to estimate $I(B_\mu; \boldsymbol{X})$.

As outlined in [IUH22], intuitively, the function ξ converts the success rate to the number of bits required for a successful distinguish. If the attacker requires a μV-PC oracle with a success rate of 1, then $\xi(1) = H(B_\mu) = \log_2 \mu$, which indicates that the attacker requires $\log_2 \mu$-bit information for the classification. In contrast, if the attacker has no advantage in the distinguisher (i.e., $\alpha_t = 1/\mu$), then $\xi(1/\mu) = 0$, which indicates that the attacker has no (i.e., zero-bit) information about the oracle output. Equation (5) states that the attacker should receive more significant bits of information through the side-channel traces (i.e., $tI(B_\mu; \boldsymbol{X})$) than the bits of information for achieving a given success rate.

We now discuss the relationship between the successful implementation of the MV-PC oracle and μ in Equation (8). Given a value of μ, the range of $I(B_\mu; \boldsymbol{X})$ is determined by $[0, \log_2 \mu]$. When μ increases, on the one hand, the coefficient of the right-hand side of the inequality in Equation (8) increases; as the attacker requires more bits to implement a μV-PC oracle for greater μ. This means that the number of traces for successful distinguish potentially increases. On the other hand, the value of $I(B_\mu; \boldsymbol{X})$ is also likely to increase as μ increases. From the information-theoretic viewpoint, this indicates that the number of traces for successful distinguish does not increase as $\log_2 \mu$ increases if $I(B_\mu; \boldsymbol{X})$ sufficiently increases. This situation would frequently occur for some (sufficiently practical) devices with low noise (such as the one used in the experiment in [BS21]). If we design an NN that can sufficiently exploit information about B_μ from $\boldsymbol{X}$, we can implement a μV-PC oracle with only a few traces, even for large μ. This implies that the attacker can receive more bits of information from a trace as $I(B_\mu; \boldsymbol{X})$, which yields a non-trivial decrease in the total number of attack traces required for key recovery.

5 Experimental Validation

5.1 Experimental setup

In this section, we describe the experimental attacks we conducted to validate the feasibility and efficiency of the proposed attack. In the experiment, we employed CUDA 11.4, cuDNN 8.0.5, Tensorflow-gpu 2.4.1, and Keras 2.4.0 on an nIntel Xeon W-2145 3.70 GHz and NVIDIA GeForce RTX 2080 Ti to carry out NN training. The learning rate was

Table 7: NN hyperparameters for μ-classification

	Input	Operator	Output	Activation function	Batch normalization	Pooling	Stride
Conv1	1000×1	conv1d(3)	16	SELU	Yes	Avg (2)	2
Conv2	500×4	conv1d(3)	16	SELU	Yes	Avg (2)	2
Conv3	250×4	conv1d(3)	16	SELU	Yes	Avg (2)	2
Conv4	125×4	conv1d(3)	32	SELU	Yes	Avg (2)	2
Conv5	62×8	conv1d(3)	32	SELU	Yes	Avg (2)	2
Conv6	31×8	conv1d(3)	32	SELU	Yes	Avg (2)	2
FLT	15×8	flatten	120	-	-	-	-
FC1	120	dense	20	SELU	No	No	-
FC2	20	dense	20	SELU	No	No	-
FC3	20	dense	μ	Softmax	No	No	-

Table 8: Experimental conditions for the evaluation of a μ-classification NN

Reference	`pqm4` [KRSS19, pqm21]
Device	ARM Cortex-M4 (STM32F407VGT6U)
Board	STM32F407G-DISC1
Side-channel	EM radiation
Measurement interface	Langer EMV-Technik RF-U T-2 probe
# Training traces	$1,000 \times \mu$
# Validation traces	$500 \times \mu$
# Test traces	$500 \times \mu$

0.0001, the batch size was 64, and the number of epochs was 100. Table 7 lists the hyperparameters of the CNN for traces with 1,000 sample points, where the top and bottom rows denote the input and output layers, respectively, and the remaining hidden layers are connected in ascending order from the input to output. In the "Input" column, $S_1 \times S_2$ denotes the input shape, where S_1 is the trace size and S_2 is the input dimension. In the "Operator" column, the operation at each layer, conv1d(F), denotes a convolution with a filter size of F.

Table 8 lists the experimental conditions used to evaluate the attack feasibility. We used an ARM Cortex-M4, which is a major micro-controller to evaluate PQC implementation including side-channel aspects. We employed software implementation of three symmetric primitives which are frequently used for the RO instantiation in KEM: non-protected AES, SHA3-512, and SHAKE128/256. The input value to the symmetric primitive was given in μ patterns, and the key was fixed. For the test (i.e., the evaluation of the attack phase), we acquired 2,048 traces for each plaintext ($2{,}048 \times \mu$ traces in total). To evaluate the accuracy of the test, we randomly selected $500 \times \mu$ labeled traces for one trial, evaluated the accuracy for each trial, repeated this trial 10 times, and averaged the accuracy. Note that we evaluate the proposed neural distinguisher in this paper, which is sufficient to evaluate the attack feasibility as the whole success rate and number of traces for key recovery are dependant solely on the quality of distinguisher.

5.2 Evaluation result of neural distinguisher

Table 9 shows the accuracy of the trained NNs on the test sets for $\mu = 2^N$ ($N = 1, 2, \ldots, 8$). This table shows that the trained NNs achieved a very high accuracy compared to DL-based key-recovery side-channel attacks on symmetric ciphers, even when the number of classes increased. Even for large μ values, such as $N = 8$, the NN could estimate B_μ with far higher accuracy compared with the case of key-recovery side-channel attacks (e.g., [PHJ+18]). This could be because the NNs in the proposed attack scenario could exploit a whole operation rather than a partial operation (e.g., S-box). The high accuracy for large μ values validates the efficiency of the proposed attack when using the MV-PC oracle in practice. We note here that a number of oracle accesses are still required even with such a high accuracy for key recovery; therefore, we should consider further

Table 9: NN accuracy to distinguish PRF input

	Accuracy for 2^N classification							
	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8
AES	0.9995	0.9997	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
SHA3-512	0.9993	0.9998	0.9994	0.9993	0.9993	0.9993	0.9992	0.9992
SHAKE128	0.9995	0.9997	0.9994	0.9995	0.9982	0.9982	0.9897	0.9746

enhancements of the MV-PC oracle accuracy using multiple inference results.

We then evaluated the numbers of traces required for a reliable MV-PC oracle access by the NLL-based and majority-voting-based distinguishers, respectively. Table 10 reports the numbers of traces required for reliable PC oracle implementation. For a comparison, Table 10 also lists the results for an existing (non-neural) distinguisher based on the L2 norm using a t-test-based template in [RRD+22], called L2-norm-based distinguisher in this paper. Here, we empirically evaluated the distinguish success rate using t traces by means of *resampling from samples* 10,000 times (for each t), as in many existing studies on DL-based side-channel attack, including [UXT+21]. The NLL was computed by Equation (3). For the NLL-based and L2-norm-based distinguishers, the numbers of required traces t to realise a single μ-class PC oracle were finally determined when the PC oracle output matched the correct label (i.e., actual plaintext) for all 10,000 trials. For the majority-voting-based distinguisher, the number of traces was determined as the minimum value satisfying $\alpha_t \geq \tau = 0.999999$.

From Table 10, we confirm the effectiveness of the NLL-based distinguisher, which achieves the fewest number of traces (i.e., highest efficiency) for all situations in this experiment As proven in Theorem 1, we can achieve an optimal distinguisher if we have the true conditional probability distribution. This result implies that the NN training would be appropriately performed, and the resulting NN would sufficiently imitate the true conditional probability distribution. The majority-voting-based distinguisher requires more traces than the NLL-based one. This would be because the number of traces for the majority-voting-based distinguisher was evaluated using an inequality, which would not be very tight/precise for the evaluation of actual success rate, though it can be easily evaluated from the accuracy of NN, and therefore used for an analytical evaluation, as shown in Section 5.3. The L2-norm-based distinguisher in [RRD+22] also had an efficiency comparable to the NLL-based distinguisher in some parts of Table 10, though it sometimes required a lot of traces for an empirical accuracy of 100%, and it could not achieve a reliable MV-PC oracle for SHAKE128 in our experiment. As a result, the generality and applicability of L2-norm-based distinguisher was unclear in this experiment. At least, the L2-norm-based distinguisher was not robust to the translation of traces (due to jitter or random delay countermeasures) and cannot be applied to masked implementations against side-channel attacks. In contrast, our neural distinguishers were more robust to such a translation and would be effectively applied to masked implementations as evaluated in previous studies on DL-based side-channel attacks (using ASCAD [BPS+20]).

5.3 Evaluation of number of traces for successful key recovery

As a preliminary proof-of-concept, we first evaluate the number of attack traces using a majority-vote-based distinguisher as an analytical lower bound for the success rate given an NN. Figure 1 shows the number of attack traces required for the key recovery using the majority-vote-based distinguisher and an NN evaluated in Section 5.2. Table 11 shows the minimum number of attack traces in Figure 1, the number of classes at that time, and the reduction rate from the conventional 2-classification. Note that for the attack on SIKE, we employed a 3^N-valued PC oracle and evaluated $2^{N'}$-classification NNs, which includes a 3^N-classification such that $2^{N'} > 3^N$. Note also that $N = 1$ is equivalent to the conventional attack using a binary PC oracle, as in [UXT+21], except for the case

Table 10: Number of traces required for available one μ-class PC oracle

Implementation	Evaluation method	Accuracy for 2^N classification							
	$N =$	1	2	3	4	5	6	7	8
AES	Majority vote†	7	5	5	5	5	5	5	5
	NLL	2	2	2	2	2	2	2	2
	L2 norm‡	2	2	2	2	2	3	4	5
SHA3-512	Majority vote†	5	5	5	7	7	7	9	9
	NLL	2	3	3	3	3	3	3	3
	L2 norm‡	2	4	5	4	4	4	4	5
SHAKE128	Majority vote†	5	5	7	7	7	11	13	9
	NLL	2	2	2	2	4	3	3	4
	L2 norm‡	100<	100<	150<	180<	180<	200<	200<	200<

† For majority vote, the number of traces was determined as the minimum value satisfying $\alpha_t \geq \tau = 0.999999$.
‡ L2 norm indicates the approach in [RRD+22]: point-of-interest selected using t-test, template created as their average vector for each value of B_μ, and b_μ determined according to an L2 norm between the template vector and a vector of point-of-interest of the attack trace.

Table 11: Minimum number of attack traces required to achieve specific success rates (SRs) using majority-voting-based distinguisher

Scheme	Instance	# Minimum traces for attack phase (μ, reduction ratio from [UXT+21])				
	SR $\geq$	$1-0.1^6$	$1-0.1^7$	$1-0.1^8$	$1-0.1^9$	$1-0.1^{10}$
Kyber	Kyber-512	1,728 (256, 87.5%)	1,728 (256, 87.5%)	1,728 (256, 90.0%)	2,112 (256, 87.5%)	2,112 (256, 89.4%)
	Kyber-1024	3,456 (256, 87.5%)	3,456 (256, 87.5%)	4,224 (256, 87.5%)	4,224 (256, 87.5%)	4,224 (256, 87.5%)
Saber	LightSaber-KEM	5,760 (256, 79.2%)	5,760 (256, 79.2%)	7,040 (256, 79.2%)	7,040 (256, 79.2%)	7,040 (256, 82.4%)
	FireSaber-KEM	3,456 (256, 87.5%)	3,456 (256, 87.5%)	4,224 (256, 87.5%)	4,224 (256, 87.5%)	4,224 (256, 89.4%)
FrodoKEM	FrodoKEM-640	46,080 (64, 80.0%)	46,080 (64, 83.6%)	55,510 (32, 80.3%)	55,510 (32, 80.3%)	56,320 (32, 83.1%)
	FrodoKEM-1344	77,436 (64, 80.0%)	77,436 (64, 83.6%)	93,184 (32, 80.3%)	93,184 (32, 80.3%)	107,520 (64, 80.8%)
NTRU Prime	ntrulpr653	2,205 (256, 87.5%)	2,205 (256, 87.5%)	2,695 (256, 87.5%)	2,695 (256, 87.5%)	2,695 (256, 89.4%)
	ntrulpr1277	4,311 (256, 87.5%)	4,311 (256, 89.8%)	5,269 (256, 87.5%)	5,269 (256, 87.5%)	5,269 (256, 89.4%)
SIKE	SIKEp434	406 (256, 84.4%)	522 (256, 80.0%)	522 (256, 80.0%)	639 (256, 80.0%)	639 (256, 80.0%)
	SIKEp751	448 (256, 84.4%)	576 (256, 80.0%)	576 (256, 80.0%)	704 (256, 80.0%)	704 (256, 80.0%)

of SIKE. Each number of attack traces for one μ-valued PC oracle (i.e., t) in Table 11 was determined when ${\alpha_t}^u$ was greater than a threshold τ, where u denotes the number of oracle accesses for key recovery in Table 6 and ${\alpha_t}^u$ represents the success rate of key recovery (not of one μ-valued PC oracle access). In this experiment, we varied τ from $1-0.1^6$ to $1-0.1^{10}$. Full detailed data on the number of attack traces required for each class are presented in Appendix B.

The number of attack traces was finally derived as $t \times u$ as the above calculation. These results show that the proposed attack significantly reduces the number of attack traces required for key recovery compared to the existing attack in [UXT+21] (i.e., $\mu = 2$ and $N = 1$). When $N = 5$–8, the proposed attack achieved 80–87% reductions of the cost of profiling (i.e., NN training). In addition, the number of attack traces for the highest key-recovery success rate (i.e., $\tau = 1-0.1^{10}$) was not much larger than that for $\tau = 1-0.1^6$; thus, the majority-voting-based distinguisher could achieve a high success rate with significantly fewer traces. It should be noted that the success rates of the majority-voting-based distinguisher are easy-to-evaluate and analytically guaranteed as these values are derived from an analytical inequality for a given NN accuracy, although the success rate of the NLL-based distinguisher (evaluated below) is evaluated only empirically. Thus, Table 11 shows the effectiveness of the proposed attack in terms of the number of traces for a successful attack from the theoretical perspectives.

Table 12 reports the number of traces required for a successful key recovery using the NLL-based distinguisher. Here, we used the evaluation result of NN in Section 5.2 and the numbers of traces were derived by $t \times u$ as Table 11. We consider the accuracy of the NLL-based distinguisher as 100% for a given t if the empirical evaluation in Section 5.2 achieved 100% accuracy. Although the majority vote is useful for analytical evaluation, the NLL-based distinguisher achieved the fewest number of traces, which yields efficient

Table 12: Number of attack traces required for successful key recovery using NLL-based distinguisher

KEM type	Scheme	Instance	# Traces for attack phase							
		$N =$	1	2	3	4	5	6	7	8
Lattice	Kyber	Kyber-512	3,072	2,304	1,548	1,152	936	774	666	**576**
		Kyber-1024	6,144	4,608	3,096	2,304	1,872	1548	1,332	**1,152**
	Saber	LightSaber-KEM	6,144	7,680	5,160	3,840	3,120	2,580	2,220	**1,920**
		FireSaber-KEM	6,144	4,608	3,096	2,304	1,872	1548	1,332	**1,152**
	FrodoKEM	FrodoKEM-640	51,200	25,600	17,070	12,800	20,480	12,810	**10,980**	12,800
		FrodoKEM-1344	86,016	43,008	28,672	21,504	34,416	21,504	**18,432**	21,504
	NTRU Prime	ntrulpr653	3,916	2,940	1,959	1,473	1,176	978	843	**735**
		ntrulpr1277	7,660	5,748	3,831	2,877	2,301	1,914	1,644	**1,437**
Isogeny	SIKE	SIKEp434	870	435	291	219	**174**		N/A	
		SIKEp751	957	480	321	240	**192**		N/A	

key recovery as confirmed from Table 11 and Table 12. Owing to the high accuracy of the NN, the NLL-based distinguisher achieved a sufficiently reliable μ-valued PC oracle access with only a few traces. Thus, the NLL-based distinguisher achieved the fewest number of traces for a successful key recovery.

Although some improvements of the CCA on KEMs with a binary key-mismatch or PC oracle have also been developed (e.g., in [QCZ+21]), the attack proposed here, with the MV-PC oracle, achieved a smaller number of oracle accesses and attack traces, as shown in Table 1. In addition, Shen et al. [SCZ+23] recently presented an efficient side-channel-assisted CCA using the binary PC oracle with a method to correct the errors in the NN inference, which achieved a 45.9–55.4% reduction of the number of attack traces required for key recovery for Kyber; however, the proposed attack achieved an 87% reduction. Thus, the proposed attack achieved the highest efficiency and required the least number of attack traces for key recovery compared with existing state-of-the-art CCAs and side-channel attacks.

5.4 Comparison to a state-of-the-art attack on Kyber in [RRD+22]

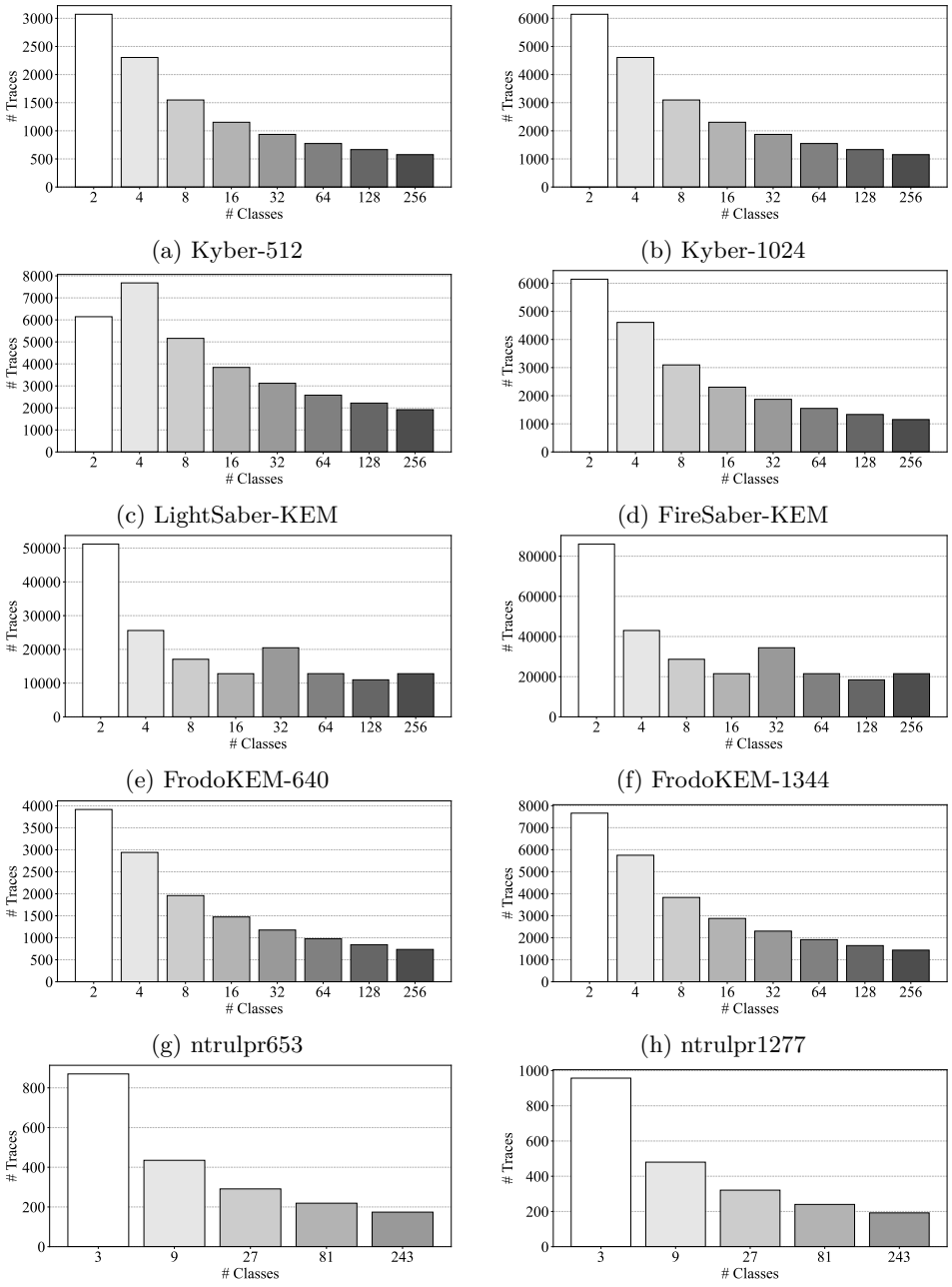
Recently, Ravi et al. presented a side-channel attack on Kyber in [RRD+22], which is very similar to our attack as it utilizes and realizes *parallel* PC oracle. In addition to the proposal of parallel PC oracle, they presented an optimal strategy to recover the secret coefficients, which yields further efficiency in terms of the number of queries. In fact, their attack requires only 190 queries[5] if $N = 8$, while our attack makes 192 queries.

Moreover, they presented another side-channel distinguisher based on t-test-based template and its L2 norm from attack traces. However, note that their distinguisher is not shown to be theoretically optimal in terms of success rate (*i.e.*, the number of traces to achieve an SR) unlike ours. The optimality of our distinguisher is proven in Section 4.2. In fact, our experimental results showed that our neural distinguisher requires fewer traces for a reliable MV-PC oracle than [RRD+22].

It should be noted that our distinguisher would require a huge *offline* computational cost to achieve an optimal attack because we need to train an NN for the approximation of $p_{B_\mu|\boldsymbol{X}}$ (compared to [RRD+22]). However, such an offline computational cost is not critical and would be trivial in many attack scenarios. In addition, a major goal for the security evaluation of KEM schemes is to analyze a theoretical potential of side-channel attack[6].

[5] We derived the value of 190 by visually reading the graph in the paper.

[6] For example, if we want to determine a rekeying frequency with regard to side-channel attack, we should consider the online computational cost (*i.e.*, the number of required attack traces) and the offline cost would not be important.



(a) Kyber-512 (b) Kyber-1024

(c) LightSaber-KEM (d) FireSaber-KEM

(e) FrodoKEM-640 (f) FrodoKEM-1344

(g) ntrulpr653 (h) ntrulpr1277

(i) SIKEp434 (j) SIKEp751

Figure 1: Number of attack traces required for successful key recovery

6 Conclusion

In this paper, we presented an efficient power/EM side-channel attack on KEMs with FO-like transforms by utilizing the MV-PC oracle. The proposed SCA is an extension/generalization of the side-channel-assisted CCA, in which the re-encryption leakage is used to implement a decryption oracle. We also designed a DL-based side-channel distinguisher using a multi-class classification NN and discussed its information-theoretic aspects. We demonstrated a set of experimental attacks using typical implementations of

Table 13: Parameter sets of Saber in Round 3

parameter sets	k	q	p	T	η	h_2
LightSaber	2	8192	1024	8	5	196
Saber	3	8192	1024	16	4	228
FireSaber	4	8192	1024	64	3	252

symmetric primitive for an RO. The results demonstrated that the proposed SCA can perform key recovery against major lattice-based and isogeny-based KEM implementations with significantly fewer attack traces than existing state-of-the-art (side-channel-assisted) CCAs. To the best of the authors' knowledge, the proposed attack achieved the smallest number of attack traces for the key recovery among the existing attacks.

The proposed attack is more efficient for larger numbers of classifications (i.e., μ). This yields attack efficiency, but also incurs a higher cost in NN training (i.e., profiling). In practice, the cost of (offline) profiling would be trivial compared to that of an online attack phase. Developing a more efficient learning method for the proposed attack would be an important area for future research. In addition, further evaluation and validation of the proposed attack for other implementations, especially masked ones, would be an interesting future research area.

Moreover, the explainability of NN in our attack is also an important topic to be addressed. On the one hand, we have provided a theorem that states a true conditional probability distribution provides an optimal attack, which motivates us to use an NN training. On the other hand, it is still unclear, for example, why and how the NN exploits leakage, what the NN actually learns, and when the NN training is successful. Although we did not discussed the NN explainability in this paper, studying it would contribute to the theoretical evaluation of attack feasibility and development of effective countermeasures against the proposed side-channel attack.

Acknowledgments

We would like to thank Dr. Sebastian Berndt for the shepherding care. This work has been supported by JSPS Kakanhi Grant No. 17H00729 No. 19H21526, and JST CREST No. JPMJCR19K5.

Appendices

A Description of MV-PCA on other KEMs

A.1 Saber

Let $\mathcal{R} = \mathbb{Z}[x]/(x^{256}+1)$ and $\mathcal{R}_q = \mathbb{Z}[x]/(x^{256}+1, q)$, where each coefficient is in the range $[0, q)$. For $a \in \mathbb{Z}^+$, we let $\mathcal{S}_a = \{f \in \mathcal{R} : f_i \in \{-a, -a+1, \ldots, a-1, a\} \text{ for } i = 0, \ldots, 255\}$.

Saber has three parameter sets, LightSaber, Saber, and FireSaber, summarized in Table 13. A ciphertext of Saber is denoted by $(\mathbf{c}_1, c_2) \in \mathcal{R}_p^k \times \mathcal{R}_T$ and a secret key is denoted by $\mathbf{s} \in \mathcal{S}_\eta^k$. The decryption algorithm first computes $M = \langle \mathbf{c}_1, \mathbf{s} \rangle - (p/T)c_2 + \sum_{i=0}^{255} h_2 x^i \in \mathcal{R}_p$ where $h_2 = p/4 - p/2T + q/2p$ and obtains the plaintext $m' = (M \gg (\log_2(p) - 1)) \in \mathcal{R}_2$, that is, takes MSBs of $M \in \{0, \ldots, 1023\}^{256}$.

KR-PCA and KR-2^{N}V-PCA for Saber/FireSaber: We can mount an attack against Saber and FireSaber similar to the aforementioned KR-MV-PCA against Kyber by extending the existing KR-PCA against them in [OUKT21]. Adapting and summarizing the

Table 14: Saber and FireSaber: the behavior of m'_0 of $m' = \mathsf{Dec}(\mathsf{sk}, (\mathbf{c}_1, c_2))$ on a ciphertext $(\mathbf{c}_1, c_2)$ with $\mathbf{c}_1 = U \cdot x^{256-a} \cdot \mathbf{e}_i$ and $c_2 = t$

(a) Saber with $U = -54$ and -57

sk_i \ t	$U=-54$		$U=-57$							
	0	1	0	1	2	3	4	5	6	7
-4	0	1	0	1	1	1	1	1	1	1
-3	0	0	0	1	1	1	1	1	1	1
-2	0	0	0	0	1	1	1	1	1	1
-1	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1
$+1$	0	0	0	0	0	0	0	1	1	1
$+2$	0	0	0	0	0	0	0	0	1	1
$+3$	0	0	0	0	0	0	0	0	0	1
$+4$	0	0	0	0	0	0	0	0	0	0

(b) FireSaber with $U = -15$

sk_i \ t	0	13	14	15	16	17	18
-3	0	1	1	1	1	1	1
-2	0	0	1	1	1	1	1
-1	0	0	0	1	1	1	1
0	0	0	0	0	1	1	1
$+1$	0	0	0	0	0	1	1
$+2$	0	0	0	0	0	0	1
$+3$	0	0	0	0	0	0	0

result of Osumi et al. [OUKT21], we obtain Table 14: a table of the behavior of decrypted messages in special ciphertexts $\mathbf{c}_1 = U \cdot x^{256-a} \cdot \mathbf{e}_i$ and $c_2 = t$ to determine $s_{i,a}$ (we can verify this behavior by direct computation).

For Saber, we first check whether a coefficient is -4 or not with $U = -54$ and then determine the other 8 cases with three adaptive queries with $U = -57$. As in the case for Kyber, we can run this test *in parallel*, and we estimate the upper bound of the number of oracle accesses to be $4 \cdot \lceil 256/N \rceil \cdot 3$.

For FireSaber, we can run a binary search on $s_{i,j} \in \{-3, \ldots, +3\}$ *in parallel* and we estimate the upper bound of the number of oracle accesses to be $3 \cdot \lceil 256/N \rceil \cdot 4$.

KR-PCA for LightSaber: We now review the KR-PCA against LightSaber in Round 3 [HV20], which consists of two phases. Following the computation of [HV20], for the input $(\mathbf{c}_1, c_2)$, the decryption algorithm computes

$$M_j = (\langle \mathbf{c}_1, \mathbf{s} \rangle)_j - 128 \cdot c_{2,j} + 196 \bmod^+ 1024 \in \{0, 1, \ldots, 1023\}$$

and decodes M_j into $m'_j = 0$ if $M_j \leq 512$ and into 1 otherwise. Let $\mathcal{I} := \{-5, -4, -3, -2, +2, +3, +4, +5\}$.

The attacker first determines whether $s_{i,j}$ is in $\mathcal{I}$ or $\{-1, 0, +1\}$. Suppose that $\mathbf{c}_1 = U \cdot \mathbf{e}_i$ for some constant $U \in [-196/5, 196/5]$.

- We first consider the case for $c_{2,j} = 0$. For any $s_{i,j} \in \{-5, \ldots, +5\}$, we have $m_j = 0$ since $0 \leq U s_{i,j} + 196 < 512$.

- Next, we consider the case for $c_{2,j} = 2$. Note that we have $(-128 \cdot 2 + 196) \bmod^+ 1024 = 964$. For $c \in \{2, 3, 4, 5\}$, we let $U = 60/c$. We have $m'_j = 0$ if and only if $((60/c)s_{i,j} + 964) \bmod^+ 1024 < 512$, that is, $s_{i,j} \geq c$. For $c \in \{-5, -4, -3, -2\}$, we again let $U = 60/c$. By a similar computation, we have $m'_j = 0$ if and only if $((60/c)s_{i,j} + 964) \bmod^+ 1024 < 512$, that is, $s_{i,j} \leq c$.

Thus, the attacker will query $(U \cdot \mathbf{e}_i, 2x^j)$ for $U = 60/c$ with $c \in \mathcal{I}$ to the PC oracle. It can determines whether $s_{i,j}$ is in $\mathcal{I}$ or $\{-1, 0, +1\}$ using $4 = \lceil \log_2(9) \rceil$ queries.

The attacker then determines whether $s_{i,j} = -1, 0, +1$. We define $V^+ = \sum_{j:s_{i,j}=4 \text{ or } 5} 5x^j$ and $V^- = \sum_{j:s_{i,j}=-4 \text{ or } -5} 5x^j$. The attacker then makes two ciphertexts queries, $(60 \cdot \mathbf{e}_i, 2x^j + V^+)$ and $(-60 \cdot \mathbf{e}_i, 2x^j + V^-)$, to determine $s_{i,j} = -1, 0, +1$.

KR-2^{N}V-PCA for LightSaber: We now make the above KR-PCA into a KR-2^{N}V-PCA, where the decrypted plaintext only varies in the first N coefficients.

Suppose that we want to determine N sequential coefficients of s_i from its a-th coefficient for $a \in \{0, \ldots, 256 - N\}$: $s_{i,a+j}$ for $j = 0, \ldots, N-1$. In this case, we query a ciphertext

$$(\mathbf{c}_1, c_2) = \Big((-U \cdot x^{256-a})\mathbf{e}_i, \sum_{\ell=0}^{N-1} 2x^\ell\Big),$$

where $U \in [-196/5, +196/5]$. For $j = N, \ldots, 256$, we have $M_j = \pm U s_{i,a+j \bmod^+ 256} + 196$, where the sign $\pm$ depends on a and j. For all cases, we have $M_j < 512$ and $m_j = 0$.

For $j = 0, \ldots, N-1$, we have $M_j = U s_{i,a+j} - 128 \cdot 2 + 196 \bmod^+ 1024$. Thus, for $c \in \{2, 3, 4, 5\}$, we have $m'_j = 0$ if and only if $((60/c)s_{i,a+j} + 964) \bmod^+ 1024 < 512$, that is, $s_{i,a+j} \geq c$. By a similar computation, for $c \in \{-5, -4, -3, -2\}$, we have $m'_j = 0$ if and only if $((60/c)s_{i,a+j} + 964) \bmod^+ 1024 < 512$, that is, $s_{i,a+j} \leq c$. Unfortunately, we cannot make adaptive queries here. Thus, we use *eight* queries to determine whether $s_{i,a+j}$ is in $\mathcal{I}$ or $\{-1, 0, 1\}$ *in parallel*.

Then, to determine $s_{i,j} \in \{-1, 0, +1\}$, we prepare the following queries: we define $V \coloneqq \sum_{j=0}^{255} v_j x^j$. Notice that the parameter setting induces $-196 < 60 s_{i,j} - 128 v_j < 196$ for $s_{i,j} \in \mathcal{I}$. We then make a query with two ciphertexts:

$$(-60 \cdot x^{256-a} \cdot \mathbf{e}_i, -V \cdot x^{256-a} + \sum_{\ell=0,\ldots,N-1 : s_{i,a+\ell} \in \{-1,0,+1\}} 2x^\ell),$$
$$(60 \cdot x^{256-a} \cdot \mathbf{e}_i, V \cdot x^{256-a} + \sum_{\ell=0,\ldots,N-1 : s_{i,a+\ell} \in \{-1,0,+1\}} 2x^\ell),$$

to determine $s_{i,a+j} \in \{-1, 0, +1\}$ for $j = 0, \ldots, N-1$ in parallel. For the first query, we have

$$M_j = \begin{cases} 60 s_{i,a+j} - 256 + 196 \bmod^+ 1024 & \text{if } j = 0, \ldots, N-1 \text{ and } s_{i,a+j} \in \{-1, 0, +1\} \\ 60 s_{i,a+j} - 128 v_j + 196 \bmod^+ 1024 & \text{if } j = 0, \ldots, N-1 \text{ and } s_{i,a+j} \in \mathcal{I} \\ 60 s_{i,a+j} - 128 v_j + 196 \bmod^+ 1024 & \text{if } j = N, \ldots, 255 - a \\ -(60 s_{i,a+j \bmod^+ 256} - 128 v_j) & \\ \qquad + 196 \bmod^+ 1024 & \text{if } j = 255 - a, \ldots, 255. \end{cases}$$

Thus, for $j = 0, \ldots, N-1$ and $s_{i,a+j} \in \{-1, 0, +1\}$, if $s_{i,a+j} = 1$, then we have $M_j = 0$ and $m'_j = 0$; otherwise, we have $M_j = 964$ or 904 and $m'_j = 1$. For all other cases, we have $0 < M_j < 392$ and $m'_j = 0$.

For the second query, by similar computation, we have $m'_j = 1$ if and only if $j = 0, \ldots, N-1$ and $s_{i,a+j} = 0, 1$. In summary, we can determine $s_{i,a+j} \in \{-1, 0, +1\}$ for $j = 0, \ldots, N-1$ using these two queries.

Thus, the number of oracle accesses is bounded above by $8 \cdot \lceil 256/N \rceil \cdot 2 + 2 \cdot \lceil 256/N \rceil \cdot 2$, that is, $20 \lceil 256/N \rceil$.

A.2 FrodoKEM

In FrodoKEM, a lattice-based KEMs, the ciphertext is denoted by $(C_1, C_2) \in \mathbb{Z}_q^{\bar{m} \times n} \times \mathbb{Z}_q^{\bar{m} \times \bar{n}}$ and the secret key is denoted by $S \in \{-s, -s+1, \ldots, s\}^{n \times \bar{n}}$. During decapsulation, the ciphertext and secret key are used to compute $M = C_2 - C_1 S \in \mathbb{Z}^{\bar{m} \times \bar{n}}$ to obtain the plaintext $\lfloor M \cdot 2^B / q \rceil \bmod 2^B \in \mathbb{Z}_{2^B}^{\bar{m} \times \bar{n}}$.

When determining $S_{i,j}$, the attacker fixes C_1, determined from i, j, makes $S_{i,j}$ appear in the calculation of M and queries $\log_2(2s+1)$ streets of ciphertexts appropriately while changing C_2. In this way, $S_{i,j} \in [-s, +s]$ can be determined. At the same time, up to $\bar{m} \times \bar{n}$ $S_{i,j}$ can be determined in parallel. When the 2^N class PC oracle is accessible, the number of queries can be reduced to $1/N$.

A.3 NTRU

Ding et al. [DDS+19] presented a KR-PCA against NTRU-HPS of NTRU and Zhang et al. [ZCD21] presented the same attack against NTRU-HRSS of NTRU. In their attacks, they sought a part of a secret key $g \in \{-1, 0, +1\}^n$. They first searched the longest chain of $+1$ or -1 in g by querying malformed ciphertexts to the PC oracle with a guess of 0^n. They then determined the remaining coefficients of g by querying crafted ciphertexts to the PC oracle with a guess of 0^n. In these procedures, decisions were made by checking whether the decrypted plaintext was $(r_{\text{guess}}, 0)$ (for details, see [UXT+21, Section 4.1.4]).

Ravi et al. [REB+21] presented an SCA-assisted KR-PCA against NTRU (and Streamlined NTRU Prime of NTRU Prime) by extending the chosen-ciphertext attack against the old-school NTRU [JJ00]. In their attack, they tested whether a decrypted plaintext was $(r_{\text{guess}}, m_{\text{guess}})$ and determined a part of the secret key.

We currently do not know how to reduce the number of queries for these attacks using the MV-PC oracle. We leave exploiting the MV-PC oracle as an interesting open problem for future work.

A.4 Streamlined NTRU Prime in NTRU Prime

Ravi et al. [REB+21] also presented an SCA-assisted KR-PCA against Streamlined NTRU Prime of NTRU Prime, as already referred. In their attack, they implemented the PC oracle using an SCA, and checked whether a decrypted plaintext was their guess $(r_{\text{guess}}, m_{\text{guess}})$, and determined a part of the secret key.

Again, we currently do not know how to reduce the number of queries for this attack using the MV-PC oracle. We leave exploiting the MV-PC oracle as an interesting open problem for future work.

A.5 NTRU LPRime in NTRU Prime

NTRU LPRime is similar to Kyber and Saber, and we can easily mount a KR-2^NV-PCA against it.

Let $\mathcal{R} = \mathbb{Z}[x]/(x^p - x - 1)$ and $\mathcal{R}_q = \mathbb{Z}[x]/(x^p - x - 1, q)$. Let $\mathcal{S}_d = \{f \in \mathcal{R} : f_i \in \{-d, -d+1, \ldots, d-1, d\} \text{ for } i = 0, \ldots, p-1\}$. A secret key is denoted by $s \in \mathcal{S}_1$.

We now briefly review the KR-PCA in [XIU+21] with a small adaption. First, Xagawa et al. determined the first N coefficients of s with two non-adaptive queries by checking whether the decrypted plaintext was of the form $(m, 1, \ldots, 1)$, where $m \in \{0,1\}^N$. For the remaining $p - N$ coefficients, they sequentially determined N coefficients of s with three adaptive queries by checking whether the decrypted plaintext was of the form $(m, 1, \ldots, 1)$, where $m \in \{0,1\}^N$. They determined that $s_j + s_{j+1} \in \{-2, -1, 0, +1, +2\}$ and computed $s_j \in \{-1, 0, +1\}$. Thus, the maximum number of queries input to the MV-PC oracle was $2 + 3 \cdot \lceil (p-N)/N \rceil$.

A.6 Code-based KEMs (Classic McEliece, BIKE, HQC)

The MV-PC oracle does not improve the number of oracle accesses for key recovery. We consider three code-based KEMs: Classic McEliece, BIKE, and HQC.

- Classic McEliece: There is no known KR-PCA against Classic McEliece.

- BIKE: The existing KR-PCA (Guo et al. [GHJ+22]) is on the basis of the GJS attack [GJS16], in which the attacker sends many ciphertexts of invalid plaintexts with special patterns and estimates the decryption failure rates for each pattern. Thus, the MV-PC oracle is not useful for mounting such an attack against BIKE.
- HQC: There are two existing KR-PCAs for HQC in Round 3: Guo et al. [GHJ+22] and Schamberger et al. [SHR+22][7]. For a secret key $s \in \mathbb{F}_2^n$ and a ciphertext $(c_1, c_2) \in \mathbb{F}_2^n \times \mathbb{F}_2^{n_1 n_2}$, the decryption of HQC computes a plaintext as follows.
  1. Compute $M = c_2 \oplus [c_1 \otimes s]_{i=0,\ldots,n_1 n_2 - 1} \in \mathbb{F}_2^{n_1 n_2}$;
  2. Decode each $M_i \in \mathbb{F}_2^{n_2}$ into $\tilde{m}_i \in \mathbb{F}_2^8 \simeq \mathbb{F}_{2^8}$ using the decoder $\mathsf{decode}_{\mathsf{dRM}}$ of the duplicated Reed-Muller code with the parameter $[n_2, 8]_2$;
  3. Decode $\tilde{m} \in \mathbb{F}_{2^8}^{n_1}$ into $m \in \mathbb{F}_{2^8}^k \simeq \mathbb{F}_2^{8k}$ using the decoder $\mathsf{decode}_{\mathsf{RS}}$ of the Reed-Solomon code with the parameters $[n_1, k_1]_{2^8}$.

  Intuitively speaking, KR-PCAs use a close-to-0 oracle, which checks whether the input is decoded into 0 by $\mathsf{decode}_{\mathsf{dRM}}$. Since we only have access to the plaintext decoded by *both* decoders, we need to take account of $\mathsf{decode}_{\mathsf{RS}}$. To determine s_i, we can design a query ciphertext (c_1, c_2) such that M_i is decoded into 0^8 by $\mathsf{decode}_{\mathsf{dRM}}$ if and only if $m = 0^{8k}$, using the properties of the Reed-Solomon code.

  To use the MV-PC oracle, we need to consider the behavior of the decrypted plaintext after it is decoded by the inner decoder $\mathsf{decode}_{\mathsf{RS}}$; we are currently unable to design such ciphertexts. We leave designing a KR-MV-PCA against HQC as an interesting open problem for future work.

A.7 Isogeny-based KEM (SIKE)

In 2022, SIKE is shown as insecure, that is, we can compute the secret key from its public key in polynomial time [CD23, MM22, Rob23]. While it is broken, we report our KR-MV-PCA against SIKE to show our extension is applicable to post-quantum KEMs other than lattice-based one.

We extend the KR-PCA in SIKE [GPST16, UXT+21] to KR-MV-PCA. We use a $\mu = 3^N$-valued PC oracle for the key recovery of SIKE. In SIKE, given $\tilde{P_A}$ and $\tilde{Q_A}$ (i.e., the points in the SIKE ciphertext), the decapsulation first calculates Bob's point $R_{AB} = \tilde{P_A} + [\mathsf{sk}_3]\tilde{Q_A}$ on Alice's elliptic curve E_A with the order of 3^{e_B}, where $\mathsf{sk}_3 \in \{1, 2, \ldots, 3^{e_B} - 1\}$ is the secret key, and the j-variant of R_{AB} is used for recovering the plaintext. In KR-3^NV-PCA, the attacker exploits the fact that the order of R_{AB} is 3^{e_B} and the KR-PCA in [GPST16, UXT+21], and recovers the secret key iteratively from the least significant ternary digit to the upper digits in an N-digit-wise manner. Let $\mathsf{sk}_3 = 3^0\beta_0 + 3^1\beta_1 + \cdots + 3^w\beta_w + \cdots + 3^{e_B-1}\beta_{e_B-1}$ ($\beta_w \in \{0, 1, 2\}$) be the ternary expanded secret key. We consider a case in which the attacker has already recovered up to the $(lN-1)$-th ternary digit (i.e., $\beta_0, \beta_1, \ldots, \beta_{Nl-1}$), and attempts to recover the Nl-th ternary digits (i.e., $\beta_{Nl}, \beta_{Nl+1}, \ldots, \beta_{N(l+1)-1}$), where l is a natural number. Note that $l = 0$ indicates that the attacker has not yet recovered any digits and starts recovering $\beta_0, \ldots, \beta_{N-1}$. Let $K_l = 3^0\beta_0 + 3^1\beta_1 + \cdots + 3^{Nl-1}\beta_{Nl-1}$ ($K_0 = 0$) be the recovered part of the secret key, up to the $(Nl-1)$-th digit. Given $\tilde{P_A}$ and $\tilde{Q_A}$, the attacker computes two points on Alice's curve as

$$\tilde{P}_A^{(i)} = \tilde{P_A} - [3^{e_B - N(l+1)}K_l]\tilde{Q_A},$$
$$\tilde{Q}_A^{(i)} = \tilde{Q_A} + [3^{e_B - N(l+1)}]\tilde{Q_A},$$

[7] Schamberger et al. [SHR+22] highlighted that the KR-PCA in [XIU+21] (and [UXT+21]) is incorrect because of the mistreatment of the decoder in the decryption.

Algorithm 3 Key-recovery for 3^N-valued plaintext-checking attack on SIKE

Input: Reference ciphertext (c_0, c_1) and candidate plaintexts $m^{(0,0,\ldots,0)}, m^{(1,0,\ldots,0)}, \ldots, m^{(2,2,\ldots,2)}$
Output: Secret key sk_3
1: **Function** AttackOnSIKE$((c_0, c_1), m^{(0,0,\ldots,0)}, m^{(1,0,\ldots,0)}, \ldots, m^{(2,2,\ldots,2)})$
2: $K_0 \leftarrow 0$;
3: **for** $l = 0$ to $\lceil (e_B - 1)/N \rceil$ **do**
4: $\tilde{P}_A^{(l)} \leftarrow \tilde{P_A} - [3^{e_B - N(l+1)} K_i]\tilde{Q_A}$;
5: $\tilde{Q}_A^{(l)} \leftarrow \tilde{Q_A} + [3^{e_B - N(l+1)}]\tilde{Q_A}$;
6: $(c_0^{(l)}, c_1) \leftarrow ((E_A, \tilde{P}_A^{(l)}, \tilde{Q}_A^{(l)}), c_1)$;
7: $K_{l+1} \leftarrow K_l + 3^{Nl} \times \mathcal{O}_{3^N}(c_0^{(l)}, c_1; m^{(0,0,\ldots,0)}, m^{(1,0,\ldots,0)}, \ldots, m^{(2,2,\ldots,2)})$;
8: **return** $K_{\lceil (e_B - 1)/N \rceil + 1}$;

generates an invalid ciphertext $(c_0^{(l)}, c_1)$ with $\tilde{P}_A^{(l)}$ and $\tilde{Q}_A^{(l)}$, and queries it. In this query, the SIKE decapsulation calculates the generator of the cyclic group as

$$\begin{aligned} R_{AB}^{(l)} &= (\tilde{P_A} - [3^{e_B - N(l+1)} K_l]\tilde{Q_A}) + [\mathsf{sk}_3](\tilde{Q_A} + [3^{e_B - N(l+1)}]\tilde{Q_A}) \\ &= R_{AB} + [3^{e_B - N(l+1)}(\mathsf{sk}_3 - K_l)]\tilde{Q_A}, \end{aligned}$$

instead of R_{AB} for the reference ciphertext (c_0, c_1), and subsequently computes the j-variant of $E_A/\langle R_{AB}^{(i)} \rangle$. Here, we have

$$\begin{aligned} [3^{e_B - N(l+1)}(\mathsf{sk}_3 - K_l)]\tilde{Q_A} &= [3^{e_B - N(l+1)} \sum_{l=i}^{e_B - 1} 3^{Nl} \sum_{v=0}^{N-1} 3^v \beta_{Nl+v}]\tilde{Q_A}, \\ &= [3^{e_B - N} \sum_{v=0}^{N-1} 3^v \beta_{Nl+v}]\tilde{Q_A}, \end{aligned}$$

because the order of $\tilde{Q_A}$ is 3^{e_B}. Therefore, $R_{AB}^{(l)}$ takes any of the 3^N values depending on $\beta_{Nl}, \beta_{Nl+1}, \ldots, \beta_{N(l+1)-1}$, which determines the value of plaintext m'. Thus, the attacker can recover the secret key digits $\beta_{Nl}, \beta_{Nl+1}, \ldots, \beta_{N(l+1)-1}$ if the attacker knows the plaintext corresponding to c'. Let $m^{(b_0, b_1, \ldots, b_{N-1})}$ be a plaintext corresponding to

$$R_{AB}^{(b_0, b1, \ldots, b_{N-1})} = R_{AB} + [3^{e_B - N}(3^0 b_0 + 3^1 b_1 + \cdots + 3^{N-1} b_{N-1})]\tilde{Q_A},$$

where R_{AB} and $\tilde{Q_A}$ are for the reference ciphertext. As the attacker can compute all values of $[3^{e_B - N} \sum_v 3^v b_v]\tilde{Q_A}$ for all $b_v \in \{0, 1, 2\}$ in advance without the secret key, a 3^N-valued PC oracle defined as

$$\begin{aligned} \mathcal{O}_{3^N}(c_0', c_1; m^{(0,0,\ldots,0)}, m^{(1,0,\ldots,0)}, \ldots, m^{(2,2,\ldots,2)}) &= 3^0 b_0 + 3^1 b_1 + \cdots + 3^{N-1} b_{N-1} \\ \text{s.t.} \quad \mathsf{SIKE.Dec}_{\mathsf{sk}_3}(c_0', c_1) &= m^{(b_0, b_1, \ldots, b_{N-1})}, \end{aligned}$$

is sufficient for key recovery. Algorithm 3 illustrates the KR-3^NV-PCA on SIKE, which exploits the 3^N-valued PC oracle at Line 7. The number of iterations is $\lceil e_B - 1/N \rceil$, which is less than that in the KR-PCA in [UXT+21] by a factor of $1/2N$. Note that we only require one PC oracle access to recover a digit if $N = 1$, whereas the conventional binary PC oracle for a reference plaintext in [UXT+21] requires at least two accesses; hence, the proposed attack yields a higher efficiency. To implement the MV-PC oracle, we can employ the hash function and PRF in SIKE.Decaps as a leakage source, as described in [UXT+21].

B Detailed evaluation results of majority-voting-based distinguisher

Table 15 shows the number of attack traces required for key recovery using majority-voting-based distinguisher; the smallest number in each column is highlighted in bold.

References

[ABH+22] Melissa Azouaoui, Olivier Bronchain, Clément Hoffmann, Yulia Kuzovkova, Tobias Schneider, and François-Xavier Standeart. Systematic study of decryption and re-encryption leakage: The case of Kyber. In *COSADE 2022*, pages 236–256, 2022.

[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. Embedded Syst.*, 2021(3):334–359, 2021.

[BHH+19] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In *TCC 2019, Part II*, pages 61–90, 2019.

[BPO+20] Florian Bache, Clara Paglialong, Tobias Oder, Tobias Schneider, and Tim Güneysu. High-speed masking for polynomial comparison in lattice-based KEMs. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2020(3):483–507, 2020.

[BPS+20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *Journal of Cryptographic Engineering*, 10(2):163–188, 2020. See also https://eprint.iacr.org/2018/053.

[BS21] Olivier Bronchain and François-Xavier Standeart. Breaking masked implementations with many shares on 32-bit software platforms: or when the security order does not matter. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2021(3):202–234, 2021.

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In *EUROCRYPT*, 2023. https://eprint.iacr.org/2022/975.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.

[dCGRP19] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful: Mutual information and success rate in side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2019(2):49–79, 2019.

Table 15: Number of attack traces required for successful key recovery using majority-voting-based distinguisher

(a) $\tau = 0.999999\ (= 1 - 0.1^6)$

KEM type	Scheme	Instance	# Traces for attack phase							
		$N =$	1	2	3	4	5	6	7	8
Lattice	Kyber	Kyber-512	13,824	5,376	4,644	3,456	2,808	2,322	1,998	**1,728**
		Kyber-1024	27,648	10,752	9,288	6,912	5,616	4,644	3,996	**3,456**
	Saber	LightSaber-KEM	27,648	23,040	15,480	11,520	9,360	7,740	6,660	**5,760**
		FireSaber-KEM	27,648	10,752	9,288	6,912	5,616	4,644	3,996	**3,456**
	FrodoKEM	FrodoKEM-640	230,400	115,200	76,815	57,600	**46,080**	46,970	54,900	67200
		FrodoKEM-1344	387,072	193536	129,024	96,768	**77,436**	78,848	92,160	11,2896
	NTRU Prime	ntrulpr653	17,622	6,860	5,877	4,419	3,528	2,934	2,529	**2,205**
		ntrulpr1277	34,470	17,244	11,493	8,631	6,903	5,742	4,932	**4,311**
Isogeny	SIKE	SIKEp434	2,610	1,305	679	511	**406**		N/A	
		SIKEp751	2,871	1,440	963	560	**448**		N/A	

(b) $\tau = 0.9999999\ (= 1 - 0.1^7)$

KEM type	Scheme	Instance	# Traces for attack phase							
		$N =$	1	2	3	4	5	6	7	8
Lattice	Kyber	Kyber-512	13,824	6,912	4,644	3,456	2,808	2,322	1,998	**1,728**
		Kyber-1024	27,648	13,824	9,288	6,912	5,616	4,644	3,996	**3,456**
	Saber	LightSaber-KEM	27,648	23,040	15,480	11,520	9,360	7,740	6,660	**5,760**
		FireSaber-KEM	27,648	13,824	9,288	6,912	5,616	4,644	3,996	**3,456**
	FrodoKEM	FrodoKEM-640	230,400	115,200	76,815	57,600	**46,080**	46,970	62,220	67,200
		FrodoKEM-1344	473,088	193,536	157,696	118,272	**77,436**	78,848	104,448	123,648
	NTRU Prime	ntrulpr653	17,622	8,820	5,877	4,419	3,528	2,934	2,529	**2,205**
		ntrulpr1277	42,130	17,244	11,493	8,631	6,903	5,742	4,932	**4311**
Isogeny	SIKE	SIKEp434	2,610	1,305	873	657	**522**		N/A	
		SIKEp751	2,871	1,440	963	720	**576**		N/A	

(c) $\tau = 0.99999999\ (= 1 - 0.1^8)$

KEM type	Scheme	Instance	# Traces for attack phase							
		$N =$	1	2	3	4	5	6	7	8
Lattice	Kyber	Kyber-512	16,896	6,912	4,644	4,224	3,432	2,838	2,442	**1,728**
		Kyber-1024	33,792	13,824	11,352	8,448	6,864	5,676	4,884	**4,224**
	Saber	LightSaber-KEM	33,792	23,040	18,920	14,080	11,440	9,460	8,140	**7,040**
		FireSaber-KEM	33,792	13,824	11,352	8,448	6,864	5,676	4,884	**4,224**
	FrodoKEM	FrodoKEM-640	281,600	140,800	93,885	70,400	56,320	**55,510**	69,540	73,600
		FrodoKEM-1344	473,088	236,544	157,696	118,272	94,644	**93,184**	116,736	134,400
	NTRU Prime	ntrulpr653	21,538	8,820	7,183	5,401	4,312	3,586	3,091	**2,695**
		ntrulpr1277	42,130	17,244	14,047	10,549	8,437	7,018	6,028	**5,269**
Isogeny	SIKE	SIKEp434	2,610	1,305	873	657	**522**		N/A	
		SIKEp751	2,871	1,440	963	720	**576**		N/A	

(d) $\tau = 0.999999999\ (= 1 - 0.1^9)$

KEM type	Scheme	Instance	# Traces for attack phase							
		$N =$	1	2	3	4	5	6	7	8
Lattice	Kyber	Kyber-512	16,896	6,912	5,676	4,224	3,432	2,838	2,442	**2,112**
		Kyber-1024	33,792	13,824	11,352	8,448	6,864	5,676	4,884	**4,224**
	Saber	LightSaber-KEM	33,792	28,160	18,920	14,080	11,440	9,460	8,140	**7,040**
		FireSaber-KEM	33,792	13,824	11,352	8,448	6,864	5,676	4,884	**4,224**
	FrodoKEM	FrodoKEM-640	281,600	140,800	93,885	70,400	56,320	**55,510**	69,540	80,000
		FrodoKEM-1344	473,088	236,544	157,696	118,272	94,644	**93,184**	116,736	145,152
	NTRU Prime	ntrulpr653	21,538	8,820	7,183	5,401	4,312	3,586	3,091	**2,695**
		ntrulpr1277	42,130	17,244	14,047	10,549	8,437	7,018	6,028	**5,269**
Isogeny	SIKE	SIKEp434	3,190	1,595	1,067	803	**639**		N/A	
		SIKEp751	3,509	1,760	1,177	880	**704**		N/A	

(e) $\tau = 0.9999999999\ (= 1 - 0.1^{10})$

KEM type	Scheme	Instance	# Traces for attack phase							
		$N =$	1	2	3	4	5	6	7	8
Lattice	Kyber	Kyber-512	19,968	8,448	5,676	4,224	3,432	2,838	2,442	**2,112**
		Kyber-1024	39,936	16,896	11,352	8,448	6,864	5,676	4,884	**4,224**
	Saber	LightSaber-KEM	39,936	28,160	18,920	16,640	13,520	9,460	8,140	**7,040**
		FireSaber-KEM	39,936	16,896	11,352	8,448	6,864	5,676	4,884	**4,224**
	FrodoKEM	FrodoKEM-640	332,800	140,800	110,955	83,200	**56,320**	64,050	76,860	86,400
		FrodoKEM-1344	559,104	236,544	186,368	139,776	111,852	**107,520**	129,024	155,904
	NTRU Prime	ntrulpr653	25,454	10,780	7,183	5,401	4,312	3,586	3,091	**2,695**
		ntrulpr1277	49,790	21,076	14,047	10,549	8,437	7,018	6,028	**5,269**
Isogeny	SIKE	SIKEp434	3,190	1,595	1,067	803	**639**		N/A	
		SIKEp751	3,509	1,760	1,177	880	**704**		N/A	

[DDS+19] Jintai Ding, Joshua Deaton, Kurt Schmidt, Vishakha, and Zheng Zhang. A simple and efficient key reuse attack on NTRU cryptosystem. IACR ePrint archive: Report 2019/1022, 2019. https://eprint.iacr.org/2019/1022.

[FMP23] Tako Boris Fouotsa, Tomoki Moriya, and Christophe Petit. M-SIDH and MD-SIDH: countering SIDH attacks by masking information. In *EUROCRYPT*, 2023. https://eprint.iacr.org/2023/013.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO 1999*, pages 537–554, 1999.

[GHJ+22] Qian Guo, Clemens Hlauschek, Thomas Johansson, Norman Lahr, Alexander Nilsson, and Robin Leander Schröder. Don't reject this: Key-recovery timing attacks due to rejection-sampling in HQC and BIKE. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2022(3):223–263, 2022.

[git21] Fast, constant-time and masked AES assembly implementations for ARM Cortex-M3 and M4. https://github.com/Ko-/aes-armcortexm, May 2021.

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on MDPC with CCA security using decoding errors. In *ASIACRYPT 2016, Part I*, pages 789–815, 2016.

[GPST16] Steven D. Galbraith, Christophe Petit, Barak Shani, and Bo Yan Ti. On the security of supersingular isogeny cryptosystems. In *ASIACRYPT 2016, Part I*, pages 63–91, 2016.

[GTN20] Qian Guo, Johansson Thomas, and Alexander Nilsson. A key-recovery timing attack on post-quantum primitives using the Fujisaki–Okamoto transformation and its application on FrodoKEM. In *CRYPTO 2020, Part II*, pages 359–386, 2020.

[HHK17] Denis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki–Okamoto transformation. In *TCC 2017, Part I*, pages 341–371, 2017.

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough: Deriving optimal distinguishers from communication theory. In *CHES 2014*, pages 55–74, 2014.

[HV20] Loïs Huguenin-Dumittan and Serge Vaudenay. Classical misuse attacks on NIST round 2 PQC – the power of rank-based schemes. In *ACNS 2020, Part I*, pages 208–227, 2020.

[ISUH21] Akira Ito, Kotaro Saito, Rei Ueno, and Naofumi Homma. Imbalanced data problems in deep learning-based side-channel attacks: Analysis and solution. *IEEE Trans. Inf. Forensics Security*, 16:3790–3802, 2021.

[IUH22] Akira Ito, Rei Ueno, and Naofumi Homma. On the success rate of side-channel attacks on masked implementations: Information-theoretical bounds and their practical usage. In *ACM CCS 2022*, pages 1521–1535, 2022.

[JJ00] Éliane Jaulmes and Antoine Joux. A chosen-ciphertext attack against NTRU. In *CRYPTO 2000*, pages 20–35, 2000.

[KPH+19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2019(3):148–179, 2019.

[KRSS19] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4. IACR ePrint archive: Report 2019/844, 2019. https://eprint.iacr.org/2019/844.

[MM22] Luciano Maino and Chloe Martindale. An attack on SIDH with arbitrary starting curve. Cryptology ePrint Archive, Paper 2022/1026, 2022. https://eprint.iacr.org/2022/1026.

[MMP+23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In *EUROCRYPT*, 2023. https://hal.science/hal-04023441.

[NDGJ21] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johanson. A side-channel attack on a masked IND-CCA secure Saber KEM. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2021(4):676–707, 2021.

[NP33] Jerzy Neyman and Egon Sharpe Peason. IX. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society A*, 231:694–706, 1933.

[OSPG18] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. Practical CCA2–secure and masked ring-LWE implementation. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2018(1):142–174, 2018.

[OUKT21] Yuki Osumi, Shusaku Uemura, Momonari Kudo, and Tsuyoshi Takagi. Key mismatch attack on SABER. In *SCIS 2021*, January 2021. In Japanese.

[PCP20] Guilherme Perin, Łukasz Chmielewski, and Stjepan Picek. Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2020(4):337–364, 2020.

[PHJ+18] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2019(1):209–237, 2018.

[pqm21] Post-quantum crypto library for the ARM Cortex-M4. https://github.com/mupq/pqm4, April 2021.

[QCZ+21] Yue Qin, Chi Cheng, Xiaohan Zhang, Yanbin Pan, and Jintai Ding. A systematic approach and analysis of key mismatch attacks on lattice-based NIST candidate KEMs. In *ASIACRYPT 2021, Part IV*, pages 92–121, 2021.

[RBRC20] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadyay. On exploiting message leakage in (few) NIST PQC candidates for practical message recovery and key recovery attacks. IACR ePrint archive: Report 2020/1559, 2020. https://eprint.iacr.org/2020/1559.

[RBRC22] Prasanna Ravi, Shivam Bhasin, Sujoy Sinha Roy, and Anupam Chattopadhyay. On exploiting message leakage in (few) NIST PQC candidates for practical message recovery attacks. *IEEE Transactions on Information Forensics and Security*, 17:684–699, 2022.

[REB+21] Prasanna Ravi, Martianus Frederic Ezerman, Shivam Bhasin, Anupam Chattopadhyay, and Sujoy Sinha Roy. Will you cross the threshold for me? generic side-channel assisted chosen-ciphertext attacks on NTRU-based KEMs. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2022(1):722–761, 2021.

[Rob23] Damien Robert. Breaking SIDH in polynomial time. In *EUROCRYPT*, 2023. https://eprint.iacr.org/2022/1038.

[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. Embedded Syst.*, 2020(3):307–335, 2020.

[RRD+22] 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. Cryptology ePrint Archive, Paper 2022/931, 2022. https://eprint.iacr.org/2022/931.

[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. Embedded Syst.*, 2023(1):89–112, 2023.

[SHR+22] Thomas Schamberger, Lukas Holzbaur, Julian Renner, Antonia Wachter-Zeh, and Georg Sigl. A power side-channel attack on the Reed-Muller Reed-Solomon version of the HQC cryptosystem. In *PQCrypto 2022*, pages 327–352, 2022.

[SKL+20] Bo-Yeon Sim, Jihoon Kwon, Joohoo Lee, Il-Ju Kim, Tae-Ho Lee, Hyojin Yoon, Jihoon Cho, and Dong-Gak Han. Single-trace attacks on message encoding in lattice-based KEMs. *IEEE Access*, 8:183175–183191, 2020.

[SXY18] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In *EUROCRYPT 2018, Part III*, pages 520–551, 2018.

[UXT+21] 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. Embedded Syst.*, 2022(1):296–332, 2021.

[WAGP20] Lennert Wouters, Victors Arribas, Benedikt Gierlichs, and Bart Praneel. Revisiting a methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2020(3):147–168, 2020.

[XIU+21] Keita Xagawa, Akira Ito, Rei Ueno, Junko Takahashi, and Naofumi Homma. Fault-injection attacks against NIST's post-quantum cryptography round 3 KEM candidates. In *ASIACRYPT 2021, Part II*, pages 33–61, 2021.

[XPR+22] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David Oswald, Wang Yao, and Zhiming Zheng. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of Kyber. *IEEE Trans. on Computers*, 71(9):2163–2176, 2022.

[ZBHV20] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2020(1):1–36, 2020.

[ZCD21] Xiaohan Zhang, Chi Cheng, and Ruoyu Ding. Small leaks sink a great ship: An evaluation of key reuse resilience of PQC third round finalist NTRU-HRSS. In *ICICS 2021*, pages 283–300, 2021.