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] NTT Social Informatics Laboratories, Nippon Telegraph and Telephone 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. This paper presents 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 CCA security. It has been shown that the side-channel leakage of re-encryption can be exploited for mounting a key-recovery plaintext-checking attack (KR-PCA), even if the CPA secure decryption constructing the KEM is securely implemented. In this paper, we propose an efficient side-channel-assisted KR-PCA on post-quantum KEMs, which achieves a key recovery with significantly fewer attack traces than the existing one. The basic ideas of the proposed attack are to present a new KR-PCA based on a multiple-valued (MV-)PC oracle and to utilize a dedicated multi-classification neural network (NN) to implement an MV-PC oracle. This paper also presents how to realize a sufficiently reliable MV-PC oracle from not completely accurate NN model outputs, and analyzes the tradeoff between the key recovery success rate and the number of attack traces, with its application to NIST PQC selected algorithm Kyber and similar lattice-based Saber, FrodoKEM and NTRU Prime, as well as SIKE, a candidate for the fourth round. Furthermore, the feasibility of the proposed attack is assessed through attack experiments on three typical PRF implementations (i.e., SHAKE, SHA3, and AES software). In consequence, we confirm that 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. We also confirm that the proposed attack can 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

Public-key cryptosystems have been essential for information systems to realize; for example, secure communication, authentication, and digital signatures are indispensable for secure information communication. Since RSA and elliptic curve cryptography (ECC), which have been representative and mainstream of public-key cryptography, were known to be broken in a quantum polynomial time employing Shor's algorithm, post-quantum cryptography (PQC) has been actively studied. For constructing PQC, a public key encryption scheme (PKE) with weak security (i.e., chosen-plaintext attack (CPA) security) is first developed, and then a key encapsulation mechanism with strong security (i.e., chosen-ciphertext attack (CCA) security) is obtained by combining the PKE with the Fujisaki–Okamoto (FO) transform [FO99] or its variants. Thus, re-encryption in decapsulation, an essence of the FO(-like) transform, plays an essential role in post-quantum security.

For the practical use of PQC, security evaluation against implementation attacks such as side-channel attacks is inevitable, in addition to the security evaluation against mathematical cryptanalysis attacks using only the input and output of the cryptographic algorithm (i.e., plaintexts, ciphertexts, and public key). Initially, the side-channel attacks focusing on the PKE decryption to exploit the secret key directly have been intensively studied, like the side-channel attacks on the modular exponentiation/scalar multiplication in RSA/ECC. Recently, another attack approach has been reported (e.g., [GTN20, RRCB20, UXT$^+$21]), where the attacker focuses on the leakage of re-encryption to implement a *decryption oracle* which enables the attacker to mount a chosen-ciphertext attack on the underlying CPA-secure PKE. Note that a decryption oracle does not necessarily mean a full decryption oracle but includes, for example, a plaintext-checking oracle and decryption-failure oracle. These attacks disclose that we need to protect not only the PKE decryption but also the whole procedures of KEM decapsulation, including re-encryption and equality/validity check of re-encrypted ciphertext, against side-channel attacks. Ueno et al. [UXT$^+$21] showed that such an attack is generally applicable to post-quantum KEMs owing to the wide deployment of FO-like transform and demonstrated that their attack can recover eight of the nine KEM candidates in the third round of the NIST PQC standardization. Thus, the potential and limitation of such attacks (e.g., the lower bound of attack traces for a successful key recovery) should be investigated to develop secure KEM implementations, as they help to develop countermeasures and design cryptographic protocols, including the determination of key pair lifetime and its update timing.

1.2 Our contribution

This paper presents an improvement of the above attacks (especially focusing on [UXT$^+$21]) in the aspects of key-recovery plaintext-checking attack (KR-PCA) and the corresponding side-channel distinguisher design based on deep learning. The number of traces required for the attack success is determined by two factors: (1) the number of decryption oracle accesses required for key recovery and (2) the number of traces to realize a reliable decryption oracle access. Therefore, a tight evaluation of the factors, that is, an efficient key-recovery algorithm with a decryption oracle sufficiently and efficiently realizable from side-channel traces, would contribute to understanding the lower bound of attack cost (i.e., the number of attack traces for successful key recovery). The basic ideas of the proposed attack are twofold. The first is to present key-recovery attacks using a multiple-valued plaintext-checking (MV-PC) oracle, which is a generalization of a PC oracle used in, for example, [RRCB20, BDH$^+$21, UXT$^+$21]; the second is implementing the MV-PC oracle using a multi-classification neural network (NN) from side-channel traces. Intuitively, we show that a 2^N-valued PC oracle ($N \in \mathbb{N}$) provides an N-bit information of secret key per access for lattice-based KEMs, whereas the conventional (binary) PC oracle in the previous literature provides one bit per access.

Meanwhile, a μ-classification ($\mu > 2$) is a more difficult task than a binary classification in general. That is, the accuracy of an MV-PC oracle can be worse than a binary oracle, which yields an increase of (2) the number of traces to realize a *reliable* PC oracle. If its

Table 1: Numbers of attack traces for successful key recovery for NIST PQC KEM standard/candidates with NIST security level 1

		Ueno et al. [UXT+21]	Qin et al. [QCZ+21]	**This work**	Maximum reduction rate
Lattice	Kyber	13,824	11,799	**1,728**	87% / 85.4%
	Saber	27,648	13,284	**3,456**	87% / 74.0%
	FrodoKEM	230,400	165,240	**46,080**	80% / 72.1%
	NTRU	14,020	N/A	14,020	0% (No improvement)
	NTRU Prime	15,327	N/A	**1,926**	87% / N/A
Code	HQC	90,555	N/A	36,222	0% (No improvement)
	BIKE	30 M	N/A	30 M	0% (No improvement)
	Classic McEliece	Unknown	Not applicable	Not applicable	Not applicable
Isogeny	SIKE	2,610	N/A	**406**	85% / N/A

increase is more significant than the reduction of (1) the number of oracle accesses, the attack using an MV-PC oracle makes no sense concerning the conventional attack using a binary PC oracle. To address this problem, we also propose the usage of μ-valued NN to implement an MV-PC oracle based on deep learning efficiently. Note that our proposal includes how to learn an NN model that can be used for implementing the proposed key-recovery attack using the MV-PC oracle. In addition, we also show and evaluate the accuracy enhancement of MV-PC oracle realized using multiple traces and discuss its information-theoretic aspects. As a result, the proposed attack achieves a significant reduction in the number of attack traces required for a successful attack compared to the conventional attacks using a binary PC oracle and binary-classification NN.

For experimental validation, we apply the proposed attack to the NIST PQC standard and fourth/third-round candidates for KEMs. Table 1 summarizes the experimental results using real devices and open-source implementations: the minimum number of attack traces required for a successful key recovery in our experiment. For a comparison, Table 1 also displays the numbers for the conventional attacks in [UXT+21] (worst case) and [QCZ+21] (average case)[1] and the corresponding reduction rate. For each key-recovery attack, we derived the number of attack traces according to the accuracy of PC oracle implementation using the NN in our experiment and majority vote (See Section 5). From Table 1, we confirm that the proposed attack reduces the number of attack traces by at most 87% (i.e., approximately 10 times more efficient), although the proposed attack is not improved for NTRU and code-based KEMs due to the difficulty in the key-recovery attack using MV-PC oracle. To the best of the authors' knowledge, the proposed attack can recover the secret key of Kyber, Saber, FrodoKEM, NTRU Prime, and SIKE with the least number of traces among the power/EM side-channel attacks focusing on re-encryption, which is validated using real devices and measurement.

1.3 Paper organization

The remainder of this paper is organized as follows: Section 2 reviews the KEM based on FO transform and the previous SCAs on KEMs focusing on FO transform. In particular, this section focuses on the previous most-generalized attack in [UXT+21]. Section 3 describes the proposed attack with its application to the NIST PQC standard and fourth/third-round candidates with theoretical evaluations. Section 4 presents the neural side-channel distinguisher design for mounting the proposed attack on practical implementation and discusses its information-theoretic aspects. Section 5 conducts an experimental validation using real devices and open-source PRF implementation compatible with PQCs. Finally, Section 6 concludes this paper.

[1] [QCZ+21, Table 6] gave the expected numbers of queries used in KR-PCAs against lattice-based PQ KEMs. We compute the expected numbers of traces by multiplying the expected numbers of queries with the number of traces implementing a PC oracle.

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

```
KeyGen
Input: $1^\lambda$
Output: $\mathsf{sk}, \mathsf{pk}, s$
1: Function KeyGen($1^\lambda$)
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: $\mathsf{pk}$
Output: $c, k$
1: Function Encaps($\mathsf{pk}$)
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 $\mathsf{H}_{\mathsf{prf}}(s, c)$;
```

2 Related Works

2.1 IND–CCA secure KEM based on the FO transform

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), key encapsulation (Encaps), and key decapsulation (Decaps). Most post-quantum KEMs are proven to be CCA secure owing to adopting FO-like transform. Here, we refer to FO transform and its variants such as [HHK17, SXY18, BHH+19] as FO-like transform. Algorithm 1 illustrates post-quantum KEMs constructed using an FO-like transform with underlying public-key encryption (PKE) scheme, where we suppose that PKE is CPA secure and consists of three probabilistic polynomial-time algorithms: key generation Gen, encryption Enc and decryption Dec, whereas there are some FO-like transform variants adaptable to other types of PKEs. Such KEMs employ pseudorandom function (PRF), pseudorandom generator (PRG), and/or cryptographic hash function denoted by G, H, and $\mathsf{H}_{\mathsf{prf}}$ in Algorithm 1, which are frequently instantiated with SHA-3 or SHAKE.

The main focus of this paper is devoted to 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 for validating 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 key $\mathsf{H}(m', c)$ 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 calculated and output at Line 8. Intuitively, the FO-like transforms perform the ciphertext verification (like fault analysis countermeasures) to detect any invalid ciphertext and to stop its output, as CCA for CPA secure PKEs generally queries invalid ciphertexts and exploits its decryption result.

2.2 Existing side-channel attacks on FO-like transform

Side-channel attacks on the FO-like transforms were initially found 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 (corresponding to Line 5 in Algorithm 1). Ciphertexts of post-quantum KEMs are treated as a long vector in CPUs/microcontrollers. A comparison using a usual operation (e.g., `memcmp`) takes a (relatively) long time if two ciphertexts are very similar to each other; otherwise, the comparison terminates immediately. They exploited this timing difference to implement a PC oracle for lattice- and code-based KEMs, and gave key-recovery attacks for lattice-based and code-based KEMs where the equivalence determination is not implemented in constant time.

Ravi et al. showed the first power/EM attack on FO-like transform. Revi et al. implemented a PC oracle or decryption failure oracle by exploiting the side-channel leakage during PRF computation in re-encryption with a t-test-based template. They demonstrated that their attack achieved the key recovery of six lattice-based KEMs, namely, Kyber, Saber, FrodoKEM, Round5, NewHope and LAC.

In [BDH+21], Bhasin et al. showed attacks on masked polynomial comparison schemes for Kyber, Saber, and FrodoKEM [OSPG18, BPO+20] by exploiting ciphertext equality check in lattice-based KEMs and demonstrated its application to Kyber. They implemented a PC oracle using a distinguisher based on the t-test. Recently, Ueno et al. showed a generalization of power/EM attacks on FO-like transforms [UXT+21]. They showed 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 the key recovery of eight out of nine KEMs of NIST PQC third-round candidates. The literature reveals that we should consider the implementation security of not only PKE.Dec but also the whole KEM computation, including re-encryption and equality/validity check when applying side-channel attack countermeasures.

In one direction in this research field, many researchers have recently been devoted to improving attack efficiency (i.e., reducing the number of attack traces) for a tight evaluation of lower bound of attack cost. For example, not limited to the attacks focusing on re-encryption, side-channel-assisted CCA approaches have been extensively studied in e.g. [XPRO20, RBRC22, SKL+20, REB+22, NDGJ21], especially for lattice-based KEMs, although some studies focus on specific parts of the underlying PKE (e.g., message encoding/decoding and number theoretic transform (NTT)) rather than FO-like transform to achieve higher efficiency. More recently, in [QCZ+21], Qin et al. showed an improvement of CCA on lattice-based KEMs using a binary key-mismatch oracle with adaptive queries, which reduces the number of oracle accesses/queries compared to CCA used in [UXT+21]. Furthermore, in [SCZ+22], Shen et al. showed a side-channel-assisted CCA using a binary PC oracle with a method to correct errors included in PC oracle outputs implemented by a side-channel. They showed that the error tolerance reduces the number of traces to implement a PC oracle, which reduces the total number of traces for the key recovery. They also demonstrated its application to Kyber, which revealed that their attack could achieve up to 55.4% reduction of the total number of traces for the key recovery compared to [UXT+21].

2.3 Side-channel-assisted KR-PCA and neural side-channel distinguisher

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 the set (i.e., $x \in \mathcal{X}$), if they are defined otherwise. The conditional probability of $Y = y$ given $X = x$ is defined as $p_{Y|X}(y|x) = p_{Y,X}(y,x)/p_X(x)$. Let Pr be the probability measure and p be the density or mass function. Let $\mathbb{E}$ denote the expectation operator. A side-channel trace is defined as $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^{\ell}$, where ℓ denotes the number of sample points.

This paper focuses on the generalized attack utilizing a PC oracle and neural distinguisher reported by Ueno et al. [UXT+21]. 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 $\hat{c}$ be an invalid ciphertext, which is a modification of c made by the attacker. When the attacker queries $\hat{c}$, a PC oracle tells whether $\hat{c}$ is decrypted to m or not. Note that the attacker cannot obtain the decryption result of PKE.Dec (i.e., m' in Algorithm 1), but can obtain only the binary information. Formally, the PC oracle is defined as

$$\mathcal{O}(c'; m) = \begin{cases} 1 & \text{if } \mathsf{PKE.Dec_{sk}}(c') = m, \\ 0 & \text{othernwise.} \end{cases}$$

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

As it is known that many post-quantum KEMs are vulnerable to KR-PCA, Ueno et al. experimentally showed that the attacker can recover the secret key of such KEMs if the attacker can implement the PC oracle by utilizing the side-channel leakage. Ueno et al. also presented the usage of the likelihood ratio test to realize a distinguisher checking if $m' = m$ or not from side-channel traces (i.e., side-channel distinguisher). Let B be a random variable that represents the oracle output (i.e., $B = \mathcal{O}(C'; m)$ and $b \in \mathcal{B} = \{0, 1\}$). Let $p_{B|\boldsymbol{X}}$ be the true conditional probability distribution of PC oracle output B given 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–Peason lemma [NP33], if the attacker knows the true distribution $p_{M|\boldsymbol{X}}$, 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. Here, the attacker queries an invalid ciphertext C' repeatedly t times to obtain t side-channe traces $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_i, \ldots, \boldsymbol{X}_{t-1}$. Ueno et al. proposed to determine 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–Peason lemma guarantees that this is the most powerful test, as Equation (1) is equivalent to the likelihood test ratio[2] 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.

However, such a true distribution is usually unavailable to an attacker/evaluator. Accordingly, Ueno et al. proposed the usage of deep learning to imitate $p_{B|\boldsymbol{X}}$. Let $q_\theta(b|\boldsymbol{x}) = q_{B|\boldsymbol{X}}(b \mid \boldsymbol{x}; \theta)$ be the conditional probability distribution represented by an NN with a parameter θ. In a typical DL, an NN q_θ is trained such that the cross entropy (CE) is minimized, and 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 θ according to $p_{B|\boldsymbol{X}}$ and hyperparameter). As the CE is usually incomputable due to the expectation (i.e., integral), in practice, it is approximated by the negative log likelihood (NLL) with the finite number of traces defined as follows:

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

[2]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 $p_{\boldsymbol{X}^t|B}(\boldsymbol{X}^t \mid b) = \prod_{v=0}^{t-1} p_{\boldsymbol{X}|B}(\boldsymbol{X}_i \mid b) = \prod_{i=0}^{t-1} p_{B|\boldsymbol{X}}(b \mid \boldsymbol{X}_i) p_{\boldsymbol{X}}(\boldsymbol{X}_i)/p_B(b)$ in accordance with Bayes' theorem. By supposing that $p_B(0) = p_B(1) = 1/2$, this equation is followed by $\operatorname{arg\,max}_b p_{\boldsymbol{X}^t|B}(\boldsymbol{X}^t \mid b) = \operatorname{arg\,max}_b \prod_{i=0}^{t-1} p_{B|\boldsymbol{X}}(b \mid \boldsymbol{X}_i) = \operatorname{arg\,max}_b \log \sum_{i=0}^{t-1} p_{B|\boldsymbol{X}}(b \mid \boldsymbol{X}_i)$.

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. In [UXT+21], Ueno et al. showed that such an NN can achieve a sufficiently high accuracy to implement a PC oracle, even for a masked software implementation [git21]. Moreover, Ueno et al. also experimentally demonstrated that the likelihood ratio test using a trained NN can achieve a higher success rate and fewer traces compared to a majority voting. This yields a key recovery of post-quantum KEMs with the practical number of traces in total.

3 Proposed Attack

3.1 Multiple-valued plaintext-checking (MV-PC) oracle

Although the PC oracle described in Section 2.3 is promising for CCA on many post-quantum KEMs, its major drawback for CCA is that the attacker can obtain no more than one-bit information per oracle access[3], which yields the large number of traces required for key recovery. The proposed attack utilizes another oracle named 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 c' is necessarily decrypted to either of 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 can know which plaintext c' is decrypted to. Such oracle is a generalization of the (binary) PC oracle with $\mu = 2$. Note that, with no side-channel, the FO-like transform theoretically does not give any information about which plaintext c' corresponds to unless the adversary generates c' from the actual encapsulation. We name this oracle *mutiple-valued plaintext-checking (MV-PC) oracle*, which is formally defined as

$$\mathcal{O}_\mu(c'; m_0, m_1, \ldots, m_{\mu-1}) = v \text{ s.t. } \mathsf{PKE.Dec}_{\mathsf{sk}}(c') = m_v.$$

Intuitively, μV-PC oracle provides at most $\log_2 \mu$ information to the attacker; therefore, CCA using μV-PC oracle would achieve the key recovery with fewer oracle accesses by a factor of $1/\log_2 \mu$ than the conventional KR-PCA with the binary PC oracle.

3.2 KR-MV-PCA algorithms for NIST PQC third-round KEM candidates

3.2.1 Kyber

We briefly review Kyber and extend the existing KR-PCA into KR-MV-PCA.

Review of Kyber: The parameter sets of Kyber in Round 3 are summarized in Table 2. 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 $[-(q-1)/2, (q-1)/2]$ for odd q and $[-q/2+1, q/2]$ for even q. For $a \in \mathbb{Z}^+$, we let $\mathcal{S}_a = \{f \in \mathcal{R} : f_i \in \{-a, -a+1, \ldots, a-1, a\} \text{ for } i = 0, \ldots, 255\}$. Let $\mathbf{e}_i = (0^{i-1}, 1, 0^{k-i}) \in \mathcal{R}^k$ be the i-th unit vector of $\mathcal{R}^k$.

A ciphertext of Kyber is denoted by $(\mathbf{c}_1, c_2) \in \mathcal{R}^k \times \mathcal{R}$ and 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 $\lceil (2/q) M \rfloor \bmod 2 \in \mathcal{R}_2$.

KR-PCA: We briefly review the KR-PCA against Kyber in Round 3 [HV20, XIU+21]. In order to determine $s_{i,j}$, the j-th coefficient of s_i, the attacker fixes $\mathbf{c}_1$ determined by i, and modifies c_2's j-th coefficient and makes $\log_2(2\eta_1 + 1)$ queries to the PC oracle to check if the decrypted plaintext is 0^{256} or not.

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

Table 2: Parameter sets of Kyber in Round 3

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

Table 3: The behavior of m_j for $j = 0, \ldots, N-1$ of $m = \mathsf{Dec}(\mathbf{s}, (\mathbf{c}_1, c_2))$ on a ciphertext $(\mathbf{c}_1, c_2) = ((U \cdot x^{256-a}) \cdot \mathbf{e}_i, \sum_{\ell=0}^{N-1} T_{a+\ell} x^\ell)$ with $U = \lceil (2^{d_U}/q) \cdot 276 \rfloor \bmod 2^{d_U}$ and $T_{a+\ell} = \lceil (2^{d_V}/q) \cdot 208 \cdot t_{a+\ell} \rfloor \bmod 2^{d_V}$.

(a) Kyber512

$s_{i,a+j}$ \ t_{a+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_{a+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

KR-2^{N}V-PCA: For the sake of easiness of implementing PC oracle, we design a query ciphertext to the decrypted plaintext varies in the first N positions of plaintexts and fix the rest as 0^{256-N} by modifying the existing KR-PCA.

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

$$(\mathbf{c}_1, c_2) = \Big((U \cdot x^{256-a})\mathbf{e}_i, (\overbrace{T_a, T_{a+1}, \ldots, T_{a+N-1}}^{N}, \overbrace{0, \ldots, 0}^{256-N})\Big),$$

where $U = \lceil (2^{d_U}/q) \cdot 276 \rfloor \bmod 2^{d_U}$ and $T_{a+j} = \lceil (2^{d_V}/q) \cdot 208 \cdot t_{a+j} \rfloor \bmod 2^{d_V}$ with $t_{a+j} \in \{-3, -2, \ldots, 3\}$. The decryption algorithm computes $M = \lceil (q/2^{d_V}) c_2 \rfloor - \lceil (q/2^{d_U}) \mathbf{c}_1 \rfloor \cdot \mathbf{s} \bmod q \in \mathcal{R}_q$. Expanding this, we have

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

Recall that the j-th plaintext is $m_j = \lceil (2/q) M_j \rfloor \bmod 2$. M_j is decoded into 0 for $j = N, \ldots, 256$ since $|(2/q) \cdot 276 \cdot b| < 1/2$ for $b \in \{-\eta_1, \ldots, \eta_1\}$. For $j = 0, \ldots, N-1$, M_j is decoded into 0 if and only if $|M_j| \le 832$, that is, $|276 s_{i,a+j} + 208 t_{a+j}| \le 832$. Thus, we have the pattern of m_j summarized in Table 3.

Hence we can run binary search on $s_{i,j} \in \{-\eta_1, \ldots, \eta_1\}$ *in parallel* and the number of queries are reduced by the factor approximately $1/N$. We estimate the upper bound of the number of oracle access as $\log_2(2\eta_1 + 1) \cdot \lceil 256/N \rceil \cdot k$.

3.2.2 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 $[0, q)$. For $a \in \mathbb{Z}^+$, we let $\mathcal{S}_a = \{f \in \mathcal{R} : f_i \in \{-a, -a+1, \ldots, a-1, a\}$ for $i = 0, \ldots, 255\}$.

Saber has three parameter sets, LightSaber, Saber, and FireSaber, summarized in Table 4. 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 decapsulation 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, taking MSBs of $M \in \{0, \ldots, 1023\}^{256}$.

Table 4: 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

Table 5: 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

	$U=-54$		$U=-57$							
sk_i \ t	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

KR-PCA and KR-2^NV-PCA for Saber/FireSaber: For Saber and FireSaber, we can mount a similar attack to the above KR-MV-PCA against Kyber by extending the existing KR-PCA in [OUKT21]. Adapting and summarizing the result of Osumi, Uemura, Kudo, and Takagi [OUKT21], we obtain the table of the behavior of decrypted messages in special ciphertexts $\mathbf{c}_1 = U \cdot x^{256-a} \cdot \mathbf{s}_i$ and $c_2 = t$ to determine $s_{i,a}$ in Table 5. (We can verify this behavior by direct computation.)

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

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

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

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

and decodes M_j into 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 $s_{i,j}$ is one of $\mathcal{I}$ or included in $\{-1, 0, +1\}$. If $c_{2,j} = 0$, then for $\mathbf{c}_1 = U \cdot \mathbf{e}_i$ with any constant $U \in [-196/5, 196/5]$ and $s_{i,j} \in \{-5, \dots, +5\}$ we have $m_j = 0$ if and only if $U s_{i,j} + 196 < 512$. In addition, if $c_{2,j} = 2$, then we have $-128 \cdot 2 + 196 \bmod 1024 = 964$. Thus, for $c \in \{2, 3, 4, 5\}$, we have $m_j = 0$ if and only if $((60/c)s_{i,j} + 964 \bmod 1024) < 512$, that is, $s_{i,j} \geq c$. By a similar computation, for $c \in \{-5, -4, -3, -2\}$, 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 can determines $s_{i,j}$ is one of $\mathcal{I}$ or included in $\{-1, 0, +1\}$ by using $4 = \lceil \log_2(9) \rceil$ queries.

The attacker then determines $s_{i,j} = -1, 0, +1$. 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^NV-PCA for LightSaber: We make the above KR-PCA into KR-2^NV-PCA where the decrypted plaintext varies only in the first N position.

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

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

where $U \in [-196/5, +196/5]$. For $j = N, \dots, 256$, we have $M_j = \pm U s_{i,a+j \bmod 256} + 196$, where the sign $\pm$ is depending on a and j. In any case, we have $M_j < 512$ and $m_j = 0$.

For $j = 0, \dots, N-1$, 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 one of $\mathcal{I}$ or included in $\{-1, 0, 1\}$ *in parallel*.

Afterward, to determine $s_{i,j} \in \{-1, 0, +1\}$, we prepare the following queries: we define $V := \sum_{j=0}^{255} v_j x^j$, where $v_j := \lceil 60 s_j / 128 \rfloor$ for $\{j : s_{i,j} \in \mathcal{I}\}$ and $v_j = 0$ for other j's. Notice that the parameter setting induces $-196 < 60 s_{i,j} - 128 v_j < 196$ for $s_{i,j} \in \mathcal{I}$. We then query two ciphertexts $(-60 \cdot x^{256-a} \cdot \mathbf{e}_i, -V \cdot x^{256-a} + \sum_{\ell=0,\dots,N-1: s_{i,a+\ell} \in \{-1,0,+1\}} 2x^\ell)$ and $(60 \cdot x^{256-a} \cdot \mathbf{e}_i, V \cdot x^{256-a} + \sum_{\ell=0,\dots,N-1: s_{i,a+\ell} \in \{-1,0,+1\}} 2x^\ell)$ to determine $s_{i,a+j} \in \{-1, 0, +1\}$ for $j = 0, \dots, N-1$ in parallel. For the first query, we have

$$M_j = \begin{cases} 60 s_{i,a+j} - 128 \cdot 2 + 196 \bmod 1024 & \text{if } j = 0, \dots, N-1 \text{ and } s_{i,a+j} \in \{-1, 0, +1\} \\ 60 s_{i,a+j} - 128 v_j + 196 \bmod 1024 & \text{if } j = 0, \dots, N-1 \text{ and } s_{i,a+j} \in \mathcal{I} \\ 60 s_{i,a+j} - 128 v_j + 196 \bmod 1024 & \text{if } j = N, \dots, 255-a \\ -(60 s_{i,a+j \bmod 256} - 128 v_j) + 196 \bmod 1024 & \text{if } j = 255-a, \dots, 255. \end{cases}$$

Thus, for $j = 0, \dots, N-1$ 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 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 $j = 0, \dots, N-1$ and $s_{i,a+j} = 0, 1$. As summary, We can determine $s_{i,a+j} \in \{-1, 0, +1\}$ for $j = 0, \dots, N-1$ by using those two queries.

Thus, the number of oracle access is upper-bounded by $8 \cdot \lceil 256/N \rceil \cdot 2$ plus $2 \cdot \lceil 256/N \rceil \cdot 2$.

3.2.3 FrodoKEM

In FrodoKEM, one of the lattice-based KEMs, is denoted the ciphertext 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 by $S \in \{-s, -s+1, \dots, s\}^{n \times \bar{n}}$. During the decapsulation, this ciphertext and the 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 this time, up to $\bar{m} \times \bar{n}$ $S_{i,j}$ can be determined in parallel at the same time. For N, when the 2^N class PC oracle is accessible, it is possible to reduce the number of queries to $1/N$.

3.2.4 NTRU

Ding et al. [DDS+19] gave a KR-PCA against NTRU-HPS of NTRU and Zhang et al. [ZCD21] gave that against NTRU-HRSS of NTRU. In their attacks, they seek a part of a secret key $g \in \{-1, 0, +1\}^n$. They first search the longest chain of $+1$ or -1 in g by querying malformed ciphertexts to the PC oracle with a guess 0^n. They then determine the remaining coefficients of g by querying crafted ciphertexts to the PC oracle with a guess 0^n. In those procedures, the decision is made by checking if the decrypted plaintext is $(r_{\text{guess}}, 0)$ or not. (For detail, see [UXT+21, Section 4.1.4].)

Ravi et al. [REB+22] gave SCA-assisted KR-PCA against NTRU (and Streamlined NTRU Prime of NTRU Prime) by extending the chosen-ciphertext attack against old-school NTRU [JJ00]. In their attack, we test if a decrypted plaintext is $(r_{\text{guess}}, m_{\text{guess}})$ or not and determine a part of the secret key.

In both attacks, we currently do not know how to reduce the number of queries using the MV-PC oracle. We leave exploiting the MV-PC oracle as an interesting open problem.

3.2.5 Streamlined NTRU Prime in NTRU Prime

Ravi et al. [REB+22] gave SCA-assisted KR-PCA against Streamlined NTRU Prime of NTRU Prime as we already referred. In their attack, they implement the PC oracle by using SCA, and they check if a decrypted plaintext is their guess $(r_{\text{guess}}, m_{\text{guess}})$ or not and determine a part of the secret key.

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

3.2.6 NTRU LPRime in NTRU Prime

NTRU LPRime is similar to Kyber and Saber, and we can mount 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 $\mathcal{S}_d = \{f \in \mathcal{R} : f_i \in \{-d, -d+1, \dots, d-1, d\}$ for $i = 0, \dots, p-1\}$. A secret key is denoted by $s \in \mathcal{S}_1$.

We briefly review the KR-PCA in [XIU+21] with a small adaption. They first determine the first N coefficients of s with non-adaptive two queries by checking if the decrypted plaintext is of the form $(m, 1, dots, 1)$, where $m \in \{0,1\}^N$. For the rest $p - N$ coefficients, they sequentially determine N coefficients of s with adaptive three queries by checking if the decrypted plaintext is of the form $(m, 1, \dots, 1)$, where $m \in \{0,1\}^N$. (They determine $s_j + s_{j+1} \in \{-2, -1, 0, +1, +2\}$ and compute $s_j \in \{-1, 0, +1\}$.) Thus the number of queries to the MV-PC oracle is at most $2 + 3 \cdot \lceil (p - N)/N \rceil$.

3.3 Code-based KEMs (Classic McEliece, BIKE, HQC)

MV-PC oracle does not improve the number of oracle accesses for key recovery. We have 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 based on the GJS attack [GJS16], in which the attacker sends many ciphertexts of invalid plaintexts with special patterns and estimated decryption failure rates for each pattern. Thus, MV-PC oracle is not useful to mount such 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].[4] 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 $M = c_2 \oplus [c_1 \otimes s]_{i=0,\dots,n_1 n_2 - 1} \in \mathbb{F}_2^{n_1 n_2}$;
    2. decode each $M_i \in \mathbb{F}_2^{n_2}$ into $\tilde{m}_i \in \mathbb{F}_2^8 \simeq \mathbb{F}_{2^8}$ by using a decoder $\mathsf{decode}_{\mathsf{dRM}}$ of the duplicated Reed-Muller code with parameter $[n_2, 8]_2$;

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

3. decode $\tilde{m} \in \mathbb{F}_{2^8}^{n_1}$ into $m \in \mathbb{F}_{2^8}^{k} \simeq \mathbb{F}_2^{8k}$ by using a decoder $\mathsf{decode}_{\mathsf{RS}}$ of the Reed-Solomon code with parameters $[n_1, k_1]_{2^8}$.

Intuitively speaking, the KR-PCAs uses a close-to-0 oracle, which checks if the input is decoded into 0 or not by $\mathsf{decode}_{\mathsf{dRM}}$. Since we only have access to the plaintext decoded by *both* decoders, we need to take account of $\mathsf{decode}_{\mathsf{RS}}$. In order 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 $m = 0^{8k}$ by using the property of the Reed-Solomon code.

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

3.4 Isogeny-based KEM (SIKE)

We extend the KR-PCA on SIKE [GPST16, UXT+21] to KR-MV-PCA. We use $\mu = 3^N$-valued PC oracle for the key recovery of SIKE. In SIKE, given $\tilde{P_A}$ and $\tilde{Q_A}$ (i.e., the points consisting 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 $\mathsf{sk}_3 \in \{1, 2, \dots, 3^{e_B} - 1\}$ 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} as well as 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 here consider a case that the attacker has already recovered up-to the $(lN-1)$-th ternary digit (i.e., $\beta_0, \beta_1, \dots, \beta_{lN-1}$), and attempts the recovery of the (lN-th ternary digits (i.e., $\beta_{lN}, \dots, \beta_{l(N+1)-1}$), where l is a natural number. Note that $l = 0$ indicates that the attacker has recovered no digit yet and starts recovering $\beta_0, \dots, \beta_{N-1}$. Let $K_l = 3^0\beta_0 + 3^1\beta_1 + \cdots + 3^{lN-1}\beta_{lN-1}$ $(K_0 = 0)$ be the recovered part of the secret key upto the $(lN-1)$-th digit. Given $\tilde{P_A}$ and $\tilde{Q_A}$, the attacker computes two points on Alice's curve as

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

generates an invalid ciphertext $(c_0^{(l)}, c_1)$ with $\tilde{P}_A^{(l)}$ and $\tilde{Q}_A^{(l)}$, and queries it. For the 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 $E_A/\langle R_{AB}^{(\tau,i)}\rangle$. Here, it holds

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

because the order of $\tilde{Q_A}$ is 3^{e_B}. Therefore, $R_{AB}^{(l)}$ takes either of 3^N values depending on $\beta_{lN}, \beta_{N+1}, \dots, \beta_{l(N+1)-1}$, which determines the value of plaintext m'. Thus, the attacker

Algorithm 2 Key-recovery 3^N-valued plaintext-checking attack on SIKE

```
Input: Reference ciphertext $(c_0, c_1)$ and candidate plaintexts $m^{(0,0,\dots,0)}, m^{(1,0,\dots,0)}, \dots, m^{(2,2,\dots,2)}$
Output: Secret key $\mathsf{sk}_3$
1: Function AttackOnSIKE($(c_0, c_1), m^{(0,0,\dots,0)}, m^{(1,0,\dots,0)}, \dots, m^{(2,2,\dots,2)}$)
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:         $K_{l+1} \leftarrow K_l + 3^{lN} \times \mathcal{O}_{3^N}((c_0^{(l)}, c_1); m^{(0,0,\dots,0)}, m^{(1,0,\dots,0)}, \dots, m^{(2,2,\dots,2)})$;
8:     return $K_{\lceil (e_B - 1)/N \rceil + 1}$;
```

can recover the secret key digits $\beta_{lN}, \beta_{N+1}, \dots, \beta_{l(N+1)-1}$ if the attacker can know the plaintext corresponds to c'. Let $m^{(b_0,b_1,\dots,b_{N-1})}$ be a plaintext corresponding to

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

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

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

is sufficient for the key recovery. Algorithm 2 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 by a factor of $1/2N$ than the KR-PCA in [UXT+21]. Note that we require only one PC oracle access to recover a digit if $N = 1$, whereas the conventional binary PC oracle for reference plaintext in [UXT+21] requires at least two accesses, which yields a higher efficiency of the proposed attack. 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].

3.5 Complexity analysis

Table 6 describes the numbers of oracle accesses for KR-μV-PCA on lattice-based KEMs and SIKE, and Table 7 reports the concrete values when $N = 1, 2, \dots, 8$. These tables show the values for schemes with the security equivalent to AES128 and AES256 (i.e., NIST security levels 1 and 5, respectively). From the tables, we can confirm that the increase of N (i.e., the usage of MV-PC oracle) significantly contributes to reducing the 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 the key recovery of these KEMs very efficiently, as demonstrated in Section 5.

4 Neural side-channel distinguisher for MV-PC oracle

4.1 Basic concept

In this paper, we propose the usage of DL to implement an MV-PC oracle as well as the existing study in [UXT+21]. Namely, we train a μ-classification NN to estimate which a plaintext $m_0, m_1, \dots,$ or $m_{\mu-1}$ corresponds to c' from the power/EM trace(s) during

Table 6: Formulas for deriving 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 7: 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

PRF/PRG execution. Note that the profiling dataset can be acquired using the target device without the secret key and the exiting study, which indicates the practicality of the proposed attack in a real scenario.

Let B_μ be the random variable representing a μ-valued PC oracle output (i.e., $B_\mu = \mathcal{O}_\mu(C'; m_0, m_1, \ldots, m_{\mu-1})$ 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 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 the dataset containing t labeled side-channel traces $(B_\mu, \boldsymbol{X})$, the attacker computes a reference ciphertext c corresponding to the reference plaintext m using KEM.Encaps and the public key, computes a valid ciphertext corresponding to each of $m_0, m_1, \ldots,$ and $m_{\mu-1}$ as well, and queries them to the target device to measure its side-channel trace. Thus, the NN training can be conducted using the dataset in the same manner as DL-based side-channel attacks on symmetric ciphers [BPS+18, KPH+19, PCP20, ZBHV20, WAGP20, ISUH21].

Note that some conventional side-channel attacks on the FO-like transform employ a template based on the t-test [RRCB20, BDH+21]. Although their methods achieved sufficient accuracy for a practical key recovery, their extension to MV-PC oracle is unknown, which indicates an advantage of the usage of the proposed neural distinguisher.

We then describe how to realize a reliable μ-class PC oracle using multiple traces, as the accuracy of an NN inference is not usually 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]; namely, we determine the oracle output $\hat{B}_\mu$ as

$$\hat{B}_\mu = \operatorname*{arg\,max}_{b \in \mathcal{B}_\mu} \log \prod_{i=0}^{t-1} q_{B_\mu|\boldsymbol{X}}(b \mid \boldsymbol{X}_i; \theta) = \operatorname*{arg\,max}_{b \in \mathcal{B}_\mu} \sum_{i=0}^{t-1} \log q_{B_\mu|\boldsymbol{X}}(b \mid \boldsymbol{X}_i; \theta). \tag{2}$$

As proven in Section 4.2, if $q_\theta = p$, then this method is optimal, that is, this method can maximize the success rate $\alpha_t = \Pr(\hat{B}_\mu = B_\mu)$.

However, one major drawback of this method is its difficulty in analytically evaluating the resulting accuracy. As an easily evaluable alternative, our method can also employ a majority voting of multiple inference results to evaluate the lower bound of the number of attack traces for the key recovery. Using an NN with an accuracy of a, we can readily evaluate the resulting accuracy α_t by

$$\alpha_t \geq 1 - \sum_{s=0}^{\lceil t/2 \rceil} \binom{t}{s} a^s (1-a)^{t-s}, \tag{3}$$

because the majority voting result is always correct if more than $\lceil t/2 \rceil$ NN inference results are correct. Note that its converse does not necessarily hold for 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 equality, we need to consider multinomial coefficients for the probabilities that all incorrect labels are inferred as correct; but such an analysis is difficult in multi-class classification. Therefore, we use Ineqality (3) as it can be readily evaluated using an NN accuracy, which is also a common metric for NN.

4.2 Information-theoretic aspects of side-channel distinguisher

In this subsection, we first show the optimality of the likelihood ratio test with the true conditional probability distribution $p_{B_\mu|\boldsymbol{X}}$ in Equation (2) as Theorem 1.

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*

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

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. Note that $\Pr(\hat{B}_\mu = B_\mu) = \mathbb{E}\mathbb{1}_{\{\hat{B}_\mu = B_\mu\}} = \mathbb{E}\ell(\hat{B}_\mu, B_\mu)$ holds. Optimal distinguish rule is defined as a rule that maximizes the success rate in guessing the correct b. To obtain the optimal distinguish rule, we rewrite the success rate as follows:

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

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

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

□

Theorem 1 validates the usage of DL to imitate $p_{B_\mu|\boldsymbol{X}}$ in the attack. We can evaluate an upper bound of the success rate or this optimal distinguish rule using Theorem 2, similarly 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 μ-valued PC oracle implementation using t traces, denoted by α_t, is bounded as*

$$\xi(\alpha_t) \leq tI(B_\mu; \boldsymbol{X}), \tag{4}$$

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 the manner similar to [dCGRP19] using Fano's inequality [CT06, Theorem 2.10.1]. See Appendix A. $\square$

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

$$t \geq \frac{\log_2 \mu}{I(B_\mu; \boldsymbol{X})}. \tag{5}$$

Proof. We have it if we substitute $\alpha_t = 1$ and $H(B_\mu) = \log_2 \mu$ to Inequality (4). $\square$

Note that an upper bound of success rate conversely represents a lower bound of the number of attack traces required to achieve a given success rate. Inequality (5) corresponds to a shortcut evaluation formula used in [ABH+22], although there was no proof for this case. It has been shown that the upper bounds of success rate based on Fano's inequality are meaningfully tight for the cases of key-recovery side-channel attacks [dCGRP19, IUH22]; accordingly, Inequality (4) is also expected to be helpful for a tight evaluation of the number of attack traces to implement a reliable MV-PC oracle. The evaluation using Inequality (4) 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 device to estimate $I(B_\mu; \boldsymbol{X})$.

As mentioned in [IUH22], intuitively speaking, the function ξ converts the success rate to the number of bits required for a successful distinction. If the attacker requires a μ-valued PC oracle with a success rate of 1, then $\xi(1) = H(\mu) = \log_2 \mu$, which indicates that attacker requires $\log_2 \mu$-bit information for the classification. In contrast, the attacker has no advantage in the distinguish (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. Inequality (4) states that the attacker should receive the more significant bits of information through the side-channel traces (i.e., $tI(B_\mu; \boldsymbol{X})$) than the above bits of information for a given success rate.

We then discuss the relation between the successful MV-PC oracle implementation and μ concerning Inequality (5). Given a value of μ, the range of $I(B_\mu; \boldsymbol{X})$ is determined by $[0, \log_2 \mu]$. If increasing μ, on the one hand, the coefficient of the right-hand side of Inequality (5) becomes greater, as the attacker requires more bits to implement a μ-valued PC oracle for greater μ. This indicates that the number of traces for successful distinction potentially increases. On the other hand, the value of $I(B_\mu; \boldsymbol{X})$ is likely to get also greater if we increase μ. This discussion indicates from the information-theoretic viewpoint that

Table 8: 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	256	SELU	No	No	-
FC2	20	dense	256	SELU	No	No	-
FC3	20	dense	μ	Softmax	No	No	-

Table 9: Experimental conditions for evaluation of μ-classification NN

Reference	`pqm4` [KRSS19, pqm21]
Device	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	$500 \times \mu$

the number of traces for successful distinction does not increase by as great as the increase of $\log_2 \mu$ if $I(B_\mu; \boldsymbol{X})$ sufficiently gets greater. This situation would frequently occur for some (sufficiently practical) devices with low noise (such as one used in the experiment of [BS21]). If we design an NN which can sufficiently exploit the information about B_μ from $\boldsymbol{X}$, we can implement a μ-valued PC oracle with a few traces even for large μ. This implies that the attacker receives more bits of information from a trace as $I(B_\mu; \boldsymbol{X})$, which yields a non-trivial reduction of the total number of attack traces for key recovery.

5 Experimental Validation

5.1 Experimental setup

We demonstrate experimental attacks 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 Intel Xeon W-2145 3.70 GHz and NVIDIA GeForce RTX 2080 Ti to carry out the NN training. The learning rate was 0.0001, the batch size was 64, and the number of epochs was 100. Table 8 illustrates the hyperparameters of the CNN for traces with 1,000 sample points, where the top and bottom columns denote the input and output layers, respectively, and the remaining hidden layers are connected in the ascending order from the input to output. In the “Input” row, $S_1 \times S_2$ denotes the input shape, S_1 is the trace size, and S_2 is the input dimension. In the “Operator” row, the operation at each layer, conv1d(F), denotes a convolution with a filter size of F.

Table 9 lists the experimental conditions for evaluating a μ-classification NN. We employed the following three PRF implementations: non-protected AES, SHA3-512, and SHAKE128/256 software. The input value to PRF is given in μ patterns, whereas the key is fixed. For the test (i.e., evaluation of attack phase), we acquired 2,048 traces for each plaintext (2,048 $\times$ μ traces in total). To evaluate the test accuracy, we randomly selected 500 $\times$ μ labeled traces for one trial, evaluated the accuracy for each trial, and repeated this trial 10 times to average the accuracy.

Table 10: NN accuracy to distinguish PRF input

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

Table 11: Minimum number of attack traces to achieve an SR

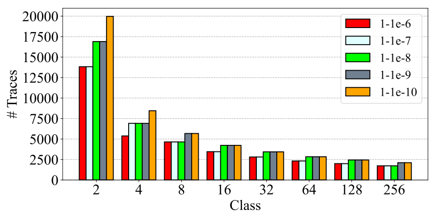
Scheme	Instance	# Minimum traces for attack phase (Class, Reduction rate)				
	SR =	$1-0.1^6$	$1-0.1^7$	$1-0.1^8$	$1-0.1^9$	$1-0.1^{10}$
Kyber	Kyber-512	1,728 (256, 87.5%)	1,728 (256, 87.5%)	1,728 (256, 90.0%)	2,112 (256, 87.5%)	2,112 (256, 89.4%)
	Kyber-1024	3,456 (256, 87.5%)	3,456 (256, 87.5%)	4,224 (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%)

5.2 Evaluation results

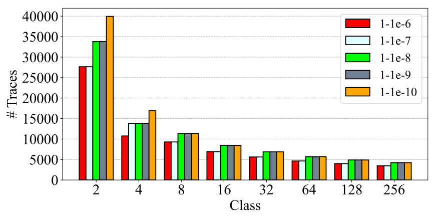
Table 10 reports the accuracy of the trained NN on the test sets for $\mu = 2^N$ ($N = 0, 1, \ldots, 8$). We can confirm from Table 10 that the trained NNs achieved a very high accuracy (compared to DL-based key-recovery side-channel attacks on symmetric ciphers) even when the number of classes increased. Even for large μ such as $N = 8$, the NN can estimate B_μ with far higher accuracy than in the case of key-recovery side-channel attacks (e.g., [PHJ+19]). This is probably because the NN in the proposed attack scenario can exploit a whole operation rather than an S-box operation. High accuracy for large μ validates the efficiency of the proposed attack using MV-PC oracle in practice, as demonstrated in the following subsection. Note that even such a high accuracy is insufficient for key recovery as it requires a lot of oracle accesses; therefore, we should consider the enhancement of MV-PC oracle accuracy using multiple inference results.

In the following, we evaluate the number of attack traces using a majority-vote-based distinguisher as a lower bound of success rate, as we mentioned in Section 4. Figure 1 reports the number of attack traces required for the key recovery with a success rate greater than threshold τ. Table 11 shows the minimum number of attack traces in Figure 1, the number of classes at that time, and the reduction rate from 2-classification. Note that, as the attack on SIKE utilizes a 3^N-valued PC oracle, we evaluated the values using a $2^{N'}$-classification NN 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 in [UXT+21], except for the case of SIKE. We determined the number of attack traces for one μ-valued PC oracle (i.e., t) should be ${\alpha_t}^u$ greater than a threshold τ, where u denotes the number of oracle accesses for key recovery in Table 7 and ${\alpha_t}^u$ represents the success rate of key recovery (not of one μ-valued PC oracle access). In this experiment, we set τ as $1 - 0.1^6$ to $1 - 0.1^{10}$. The number of attack traces is finally derived as $t \times u$. Full data on the number of attack traces for each class is given in Appendix B.

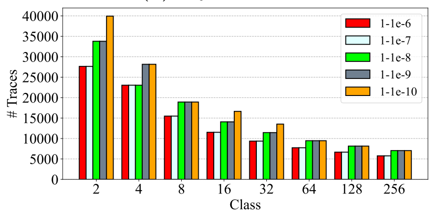
From Table 12, we can confirm that the proposed attack significantly reduces the number of attack traces for key recovery compared to the existing attack in [UXT+21] (i.e., $N = 1$). In particular, when $N = 5$–8, the proposed attack achieves at most 80–87% reduction at the cost of profiling (i.e., training NN). In addition, the number of attack traces for very high key-recovery success rate (e.g., $\tau = 1 - 0.1^{10}$) is not very larger than that for $\tau = 1 - 0.1^6$. This would be because of the high accuracy of the NN, which achieves a sufficiently reliable μ-valued PC oracle access with a few traces. Although some improvements of CCA on these KEMs with a binary key-mismatch or PC oracle have also been developed in [QCZ+21], the proposed attack with MV-PC oracle achieved



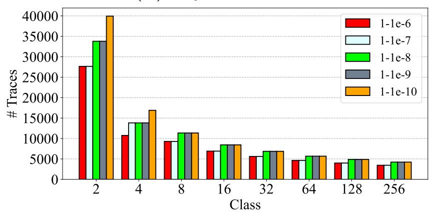
(a) Kyber-512



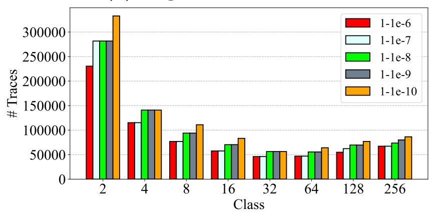
(b) Kyber-1024



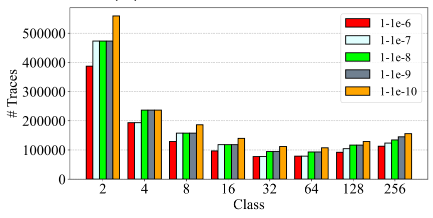
(c) LightSaber-KEM



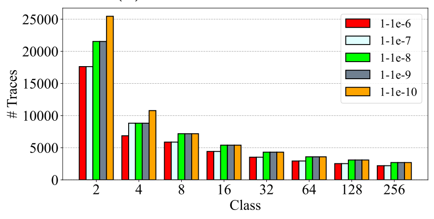
(d) FireSaber-KEM



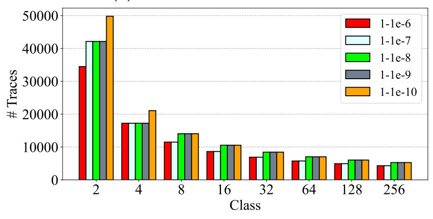
(e) FrodoKEM-640



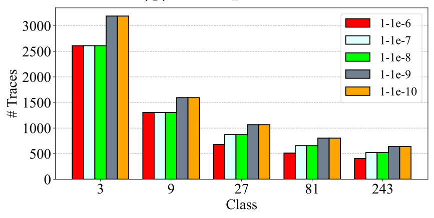
(f) FrodoKEM-1344



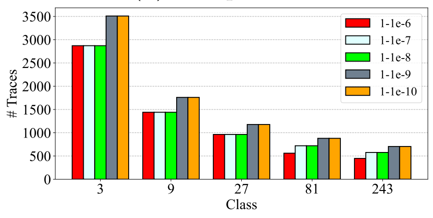
(g) ntrulpr653



(h) ntrulpr1277



(i) SIKEp434



(j) SIKEp751

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

a less number of oracle accesses/attack traces than them, as shown in Table 1. In addition, in [SCZ+22], Shen et al. 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 45.9–55.4% reduction of the number of attack traces for the key recovery of Kyber, whereas the proposed attack achieved 87% reduction. Thus, the proposed attack achieved the highest efficiency and the least number of attack traces for the key recovery with regard to the state-of-the-art CCAs and side-channel attacks.

6 Conclusion

This paper presented an efficient power/EM side-channel attack on KEMs with FO-like transforms based on MV-PC oracle. The proposed SCA is an extension/generalization of side-channel-assisted CCA using the re-encryption leakage to implement a decryption oracle. We also presented the design of 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 PRF implementations. We confirmed from the results that the proposed SCA can perform the key recovery against major lattice-based and isogeny-based KEM implementations with significantly fewer attack traces than the state-of-the-art (side-channel-assisted) CCAs. To the best of the authors' knowledge, the proposed attack achieved the least number of attack traces for the key recovery among the existing attacks.

The proposed attack is more efficient when the larger number of classifications (i.e., μ) is achieved. This yields an attack efficiency but also incurs a higher cost in NN training (i.e., profiling). In practice, the cost of (offline) profiling would be trivial rather than that of the online attack phase. Developing a more efficient learning method for the proposed attack would be an important future subject. In addition, a further evaluation/validation of the proposed attack on other implementations especially masked ones, would be future work.

Appendix A: Proof of Theorem 2

Let $H(X)$ be the entropy of a random variable X. Suppose that the side-channel attack is represented as a Markov chain of $B_\mu \leftrightarrow \boldsymbol{X} \leftrightarrow \hat{B}_\mu$, similarly to [HRG14, dCGRP19]. Due to the definition of mutual information, it holds

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

As $\hat{B}_\mu$ is a function of $\boldsymbol{X}$, it holds $H(B_\mu \mid \boldsymbol{X}^t) = H(B_\mu \mid \boldsymbol{X}^t, \hat{B}_\mu)$, which is followed by

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

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

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

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

Appendix B: Number of attack traces required for key recovery

Table 12 shows the number of attack traces required for the key recovery, where the minimum number for each column is highlighted by bold.

Table 12: Number of attack traces required for successful key recovery

(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	

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 *International Workshop on Constructive Side-Channel Analysis and Secure Design*, volume 13211 of *Lecture Notes in Computer Science*, 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*, 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+18] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. IACR ePrint archive: Report 2018/053, 2018. 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.

[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 Guilly, 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.

[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.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, 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 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*, 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*, pages 359–386, 2020.

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

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough: Deriving optimal distinguishers from communication theory. In *International Workshop on Cryptographic Hardware and Embedded Systems*, 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, 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. Cryptology ePrint Archive, Report 2022/576, 2022. `https://eprint.iacr.org/2022/576`.

[JJ00] Éliane Jaulmes and Antoine Joux. A chosen-ciphertext attack against NTRU. In *CRYPTO*, 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`.

[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 Transactions on Cryptographic Hardware and Embedded Systems*, (4):337–364, 2020.

[PHJ+19] 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.*, (1):209–237, 2019.

[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*, pages 92–121, 2021.

[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+22] 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.*, 1:722–761, 2022.

[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.

[SCZ+22] 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. Cryptology ePrint Archive, Paper 2022/563, 2022. https://eprint.iacr.org/2022/563.

[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. Cryptology ePrint Archive, Paper 2022/724, 2022. https://eprint.iacr.org/2022/724.

[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*, 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 Transactions on Cryptographic Hardware and Embedded Systems*, 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*, pages 33–61, 2021.

[XPRO20] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, and David Oswald. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of Kyber. IACR ePrint archive: Report 2020/912, 2020. https://eprint.iacr.org/2020/912.

[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*, pages 283–300, 2021.