Side-Channel Analysis on Lattice-Based KEM using Multi-feature Recognition - The Case Study of Kyber

Yuan Ma[1,2], Xinyue Yang[1,2], An Wang[3], Congming Wei[3(✉)], Tianyu Chen[1], and Haotong Xu[3]

[1] SKLOIS, Institute of Information Engineering, CAS, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{mayuan,yangxinyue,chentianyu}@iie.ac.cn
[3] School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China
{wanganl,weicm,3120231257}@bit.edu.cn

Abstract. Kyber, selected as the next-generation standard for key encapsulation mechanism in the third round of the NIST post-quantum cryptography standardization process, has naturally raised concerns regarding its resilience against side-channel analysis and other physical attacks. In this paper, we propose a method for profiling the secret key using multiple features extracted based on a binary plaintext-checking oracle. In addition, we incorporate deep learning into the power analysis attack and propose a convolutional neural network suitable for multi-feature recognition. The experimental results demonstrate that our approach achieves an average key recovery success rate of 64.15% by establishing secret key templates. Compared to single-feature recovery, our approach bypasses the intermediate value recovery process and directly reconstructs the representation of the secret key. Our approach improves the correct key guess rate by 54% compared to single-feature recovery and is robust against invalid attacks caused by errors in single-feature recovery. Our approach was performed against the Kyber768 implementation from `pqm4` running on STM32F429 M4-cortex CPU.

Keywords: Lattice-Based cryptography · Side-channel analysis · Plaintext-checking oracle · Kyber · Convolutional neural network.

1 Introduction

Classical public key cryptosystems rely on the intractability of certain mathematical problems. However, the rapid development of quantum algorithms and quantum computers poses a grave threat to these cryptographic schemes in use today. Integer factorization and discrete logarithm problems can be solved in polynomial time using Shor's algorithm [17]. Furthermore, a recent study estimated the possibility of factoring a 2048-bit RSA integer in 8 hours using "20 million noisy qubits" [5]. Therefore, it is necessary to develop novel, post-quantum secure cryptographic primitives for long-term security.

In 2016, the National Institute of Standards and Technology (NIST) initiated a process [12] to select the best post-quantum cryptography (PQC) primitives for standardization. In July 2022, NIST announced the first group of winners from its six-year competition [2]. Lattice-based cryptography prevailed, with 3 out of 4 winners, demonstrating their foundational role in PQC standards. Among them, Kyber [16], the KEM part of the Cryptographic Suite for Algebraic Cipher Suite (CRYSTALS), was chosen by NIST as the only public key encryption or key encapsulation mechanism (KEM) algorithm for standardization [2]. Shortly after, the National Security Agency included Kyber in the suite of encryption algorithms recommended for national security systems [1]. Currently, the NIST PQC process has entered the fourth round.

In addition to other desired security properties, NIST has prioritized the resilience against side-channel attacks (SCAs), before deploying these PQC algorithms in real-world applications, particularly in scenarios where an attacker could physically access an embedded device.

SCAs were first introduced by Kocher in 1996 [9]. Research has shown that power consumption, electromagnetic emanations (EM), thermal signatures, or other physical phenomena are often correlated with encrypt and decrypt operations occurring on a device [10]. Thus enabling attackers to extract sensitive information such as the long-term secret key. Based on this approach, several SCAs against lattice-based KEMs in the NIST PQC standardization process have been proposed, such as [3,6,15,18–21]. Most of them are chosen-ciphertext attacks (CCAs) due to the fact that NIST PQC KEMs are always targeting CCA security.

The recovery goals of these CCAs can be categorized into two groups: one for decrypted messages recovery [18,20] and the other for key recovery [3,6,15,19,21]. Since key recovery is more powerful than message recovery, we focus our study on key recovery SCAs. Guo et al. in [6] first proposed an oracle based on decryption-failure and instantiated the attack model to complete a timing attack on Frodo KEM. Xu et al. presented a full-decryption-based oracle in [21]. They proved that an attacker only needs 8 traces to recover a specific implementation of Kyber512 compiled at the optimization level -O0. D'Anvers et al. [3] exploited the variable runtime information of its non-constant-time decapsulation implementation on the LAC and successfully recovered its long-term secret key. This key recovery attack, named plaintext-checking (PC) oracle in [14] which was defined as a message-recovery-type attack, finds a link between the long-term secret key and specifically chosen messages and recovers the key by recovering the message. Ravi et al. [15] continue this attack conception by exploiting the leaked side information in Fujisaki-Okamoto (FO) transformation [4] or error correcting codes to propose a generic EM chosen-ciphertext SCA. Qin et al. [13] optimized the approach of [15] in terms of query efficiency. Ueno et al. in [19] further investigated the attack methods against adversaries. More appealing is that they implemented a deep-learning-based distinguisher to assist PC oracle attacks.

Our contributions. In this paper, we proposed a novel multi-feature-based side-channel attack (Multi-feature-based SCA) by extracting profiling information from multi-features. Multi-feature-based SCA constructs templates of each secret key value based on a convolutional neural network (CNN) and successfully recovers the secret key of Kyber768. In addition to improving the success rate of recovering the secret key, our approach also eliminates the occurrence of invalid attacks. In summary, we make the following contributions:

- We propose a new profiling approach named Multi-feature-based SCA, which uses multiple features to build templates for the secret key. Our approach eliminates invalid attacks and can directly recover the secret key values, bypassing the intermediate step of recovering the decrypted message.
- We build a CNN to recognize secret keys. The experimental results prove the huge advantages of CNN in constructing templates, and its recognition accuracy reached around 90%.
- Furthermore, we instantiate the described attack framework on Kyber768 and show the details in each step of the new procedure. Compared to Ueno et al.'s [19] method, our approach demonstrates an average success rate enhancement of 27.45%. Additionally, when contrasted with Ravi et al.'s [15] method, our approach exhibits an average attack success rate improvement of 53.69%.

Outline. The remainder of this paper is organized as follows. In Sect. 2, we examine the details of Kyber and the conception of binary PC oracle. Then we enumerate some previous SCAs on it. Sect. 3 outlines the basic idea of our approach, Multi-feature-based SCA. In Sect. 4, we detail our experimental setup and illustrate our attack method and the CNN construction we used. We further demonstrate the effect of our approach on improving the probability of attack success. Lastly, Sect. 5 concludes our work.

2 Background

2.1 Kyber and the binary PC oracle

KEM is a public key cryptographic primitive that encapsulates a secret key. Kyber is a chosen-ciphertext secure (CCA-secure) KEM based on the Module-learning with error (M-LWE) problem. The M-LWE problem evolves from the Ring-LWE (R-LWE) problem, with their theoretical basis being to add noise to the $\mathbf{b} = \mathbf{As}$ problem, making it difficult to recover $\mathbf{b} = \mathbf{As} + \mathbf{e}$. However, in R-LWE problem, $\mathbf{s}$ and each column of $\mathbf{A}$ are chosen from a polynomial ring, while in M-LWE, $\mathbf{s}$ and each column of $\mathbf{A}$ are selected from a module. Therefore, the M-LWE problem offers more flexibility and computational efficiency.

In Kyber, define a polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$, where modulus $q = 3329$ and $n = 256$. For every polynomial $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1} \in \mathbb{R}_q$, each coefficient $a_i \in \mathbb{Z}_q$ $(0 \leq i \leq n-1)$, represents a ring with

all elements are integers modulo q. Additions, subtractions, and multiplications of polynomials all require modulus $x^n + 1$. We use bolded uppercase letters for matrices and bolded lowercase letters for polynomial vectors. Matrix $\mathbf{A} \in \mathcal{R}_q^{k \times k}$, where its vector $(\mathbf{A}[0], \cdots, \mathbf{A}[k-1])$ represent a polynomial. $\mathbf{s}, \mathbf{e} \in \mathcal{B}_\eta^k$, where $\mathcal{B}_\eta$ represents the centered binomial distribution with parameter η, and can be generated by $\sum_{i=1}^{\eta}(a_i - b_i)$. In Kyber, a_i and b_i are uniformly random samples independently selected from $\{0, 1\}$.

Based on the above, Kyber provides three security levels with Kyber512 (NIST Security Level 1), Kyber768 (Level 3) and Kyber1024 (Level 5) with dimension $k = 2$, 3 and 4 respectively. In this paper, we focus on the implementation of Kyber768, but our approaches can also be applied to the other two sets. Parameters in Kyber768 are shown in Table 1. $k = 3$ means secret key $\mathbf{sk}$ has 3 polynomials. $(\eta_1, \eta_2) = (2, 2)$ means the coefficients in $\mathbf{sk}$ belong an integer between $[-2, 2]$. (d_u, d_v) were used in `Compress` and `Decompress` fuction.

Table 1. Parameters used in Kyber768

	Parameters				
	n	q	k	(η_1, η_2)	(d_u, d_v)
values	256	3329	3	$(2, 2)$	$(10, 4)$

Generally, a KEM consists of key generation, encapsulation, and decapsulation. But PC-based SCA is only against the decapsulation part. Thus, in Algorithm 1 and Algorithm 2, we only introduce the main parts of encapsulation and decapsulation of Kyber, ignoring details such as the Number Theoretic Transform (NTT).

Let $\lceil x \rfloor$ denotes the nearest integer to x. In the following, we first define two functions, $\texttt{Compress}_q(x, d)$ and $\texttt{Decompress}_q(x, d)$.

Definition 1. *The Compression function is defined as:* $\mathbb{Z}_q \to \mathbb{Z}_{2^d}$

$$\texttt{Compress}_q(x, d) = \left\lceil \frac{2^d}{q} \cdot x \right\rfloor \pmod{2^d}. \tag{1}$$

Definition 2. *The Decompression function is defined as:* $\mathbb{Z}_{2^d} \to \mathbb{Z}_q$

$$\texttt{Decompress}_q(x, d) = \left\lceil \frac{q}{2^d} \cdot x \right\rfloor. \tag{2}$$

We can get in [16], $\texttt{Compress}_q(x, d)$ and $\texttt{Decompress}_q(x, d)$ need polynomials for their inputs. The above operation is separately done on each coefficient in the input polynomial. Kyber uses a version of the FO transformation to achieve its stated security goals, i.e., for the chosen-plaintext secure (CPA-secure) to CCA-secure. In the following two algorithms, $\mathcal{G}$ represents a hash operation to

Algorithm 1 CCA-secure Kyber KEM based on FO transformation (Encaps)

Input: Public key $\mathbf{pk}$
Output: Ciphertext $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$, session key k
$\mathbf{m} \leftarrow \{0,1\}^{256}$
$(\bar{K}, r) = \mathcal{G}(\mathbf{m}||\mathcal{H}(\mathbf{pk}))$
$\triangleright$ $c = \text{CPA.Encrypt}(\mathbf{pk}, \mathbf{m}, r)$
  $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times k}$
  $\mathbf{r} \leftarrow \mathcal{B}_{\eta_1}^k,\ \mathbf{e}_1, \mathbf{e}_2 \leftarrow \mathcal{B}_{\eta_2}^k$
  $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$
  $\mathbf{v} = \mathbf{pk}^T\mathbf{r} + \mathbf{e}_2 + \texttt{Decompress}_q(\mathbf{m}, 1)$
  $\mathbf{c}_1 = \texttt{Compress}_q(\mathbf{u}, d_u)$
  $\mathbf{c}_2 = \texttt{Compress}_q(\mathbf{v}, d_v)$
$k = \text{KDF}(\bar{K}||\mathcal{H}(\mathbf{c}))$
return $\mathbf{c}, k$

get a 64-byte variant meanwhile, $\mathcal{H}$ represents a hash operation to get a 32-byte variant.

During Algorithm 1, the message generates a 32-byte $\mathbf{m}$ from the 0,1 space. By $\mathbf{m}$ and $\mathcal{H}(\mathbf{pk})$, we can get the pre-shared secret $\bar{K}$ and a random coin r. In the encapsulation, a CPA-secure encryption operation is used to output $\mathbf{c}_1$ and $\mathbf{c}_2$. Then, the shared secret k is calculated from $\bar{K}$ and $\mathcal{H}(\mathbf{c})$ through the key-derivation function (KDF).

Algorithm 2 CCA-secure Kyber KEM based on FO transformation(Decaps)

Input: Ciphertext $\mathbf{c}$, secret key $\mathbf{sk}$
Output: Session key k
$\mathbf{pk}, \mathcal{H}(\mathbf{pk}), z \leftarrow \text{UnpackSK}(\mathbf{sk})$
$\triangleright$ $\mathbf{m}' \leftarrow \text{CPA.Decrypt}(\mathbf{sk}, \mathbf{c})$
  $\mathbf{u}' = \texttt{Decompress}_q(\mathbf{c}_1, d_u)$
  $\mathbf{v}' = \texttt{Decompress}_q(\mathbf{c}_2, d_v)$
  $\mathbf{m}' = \texttt{Compress}_q(\mathbf{v}' - \mathbf{sk}^T\mathbf{u}', 1)$
$(\bar{K}', r') = \mathcal{G}(\mathbf{m}'||\mathcal{H}(\mathbf{pk}))$ /* Attack loaction */
$\mathbf{c}' \leftarrow \text{CPA.Encrypt}(\mathbf{pk}, \mathbf{m}', r')$
if $\mathbf{c} = \mathbf{c}'$ **then**
  return $k \leftarrow \text{KDF}(\bar{K}', \mathbf{c})$
else
  return $k \leftarrow \text{KDF}(z, \mathbf{c})$
end if

CCA.Decaps first performs the CPA decryption. In CPA-secure decryption, from $\mathbf{c}_1$ and $\mathbf{c}_2$ using `Compress`, we obtained the plaintext $\mathbf{m}'$. Then, similar

to CCA.Encaps, CCA.Decaps generates r' and $\bar{K}'$, and evaluates CPA.Encrypt $(\mathbf{pk}, \mathbf{m}', r')$. This procedure is called re-encryption. At Algorithm 2 line 8, the algorithm executes equality checking, namely, examines whether the re-encryption result $\mathbf{c}'$ is equal to the ciphertext $\mathbf{c}$. If equals, the CCA.Decaps algorithm returns the shared secret k as the ciphertext is valid; otherwise, the algorithm returns a pseudorandom number of $\mathrm{KDF}(z, \mathbf{c})$ (instead of $\perp$) as the ciphertext is invalid. Thus, the KEM scheme gives any active attacker no information about the PKE decryption result for invalid ciphertext.

The CPA-secure KEMs are vulnerable to chosen-ciphertext attacks when the secret key is reused. These attacks are generally operated in a key-mismatch or PC Oracle. The working principle of PC oracle is to recover one coefficient of the secret key polynomial at a time. Algorithm 3 depicts the PC oracle, in which the adversary sends ciphertext $\mathbf{c}$ and a reference message $\mathbf{m}$ to the oracle. The oracle tells whether $\mathbf{m}$ equals the CPA decryption result $\mathbf{m}'$ or not.

Algorithm 3 PC oracle

Input: Ciphertext $\mathbf{c}$, message $\mathbf{m}$
Output: 0 or 1
1: $\mathbf{m}' \leftarrow \mathrm{CPA.Decrypt}(\mathbf{sk}, \mathbf{c})$
2: **if** $\mathbf{m} = \mathbf{m}'$ **then**
3: **return** 1
4: **else**
5: **return** 0
6: **end if**

The key recovery process is based on the recovery of message $\mathbf{m}'$ in Algorithm 3. By constructing the selected ciphertext, we can combine every possible coefficient value in Kyber with a set of oracle response sequences. With multiple queries, we are able to recover this coefficient value. Using the rotation property of the polynomial ring, we are then able to recover the complete secret key polynomial of Kyber.

2.2 PC oracle-based SCA attacks

The LWE-based KEM in the CPA model can be upgraded to a CCA-secure KEM through FO transformation. As we described in Section 2.1, using FO transformation, the attacker cannot obtain any prompt information about the decapsulation failure when decapsulating. This theoretically provides a strong security guarantee for CPA security KEM, which can prevent selected ciphertext attacks.

However, with the help of side information, such as analyzing the power or electromagnetic waveforms of certain operations during the decapsulation process, an attacker can directly discover the CPA-secure operations inside the CCA-secure model and launch the same attack.

At CHES 2020, Ravi et al. launched a PC oracle-based SCA attack against NIST KEM by utilizing side information leaked from the re-encryption process in the FO transform [15]. Taking the attack against Kyber as an example, the attacker only needs to control $\mathbf{m}'$ to be $\mathrm{O} = (0, 0, 0, 0, \cdots)$ or $\mathrm{X} = (1, 0, 0, 0, 0, \cdots)$. In this way, they build a PC Oracle with a side-channel waveform distinguisher. In [15], Ravi et al. used simple Euclidean distances to create a recognizer with profiled waveform templates. More specifically, they first collected two sets of re-encrypted waveforms with $\mathbf{m}' = \mathrm{O}$ and $\mathbf{m}' = \mathrm{X}$. Then, they performed a Test Vector Leakage Assessment (TVLA) between the two sets to select the Point of Interest (PoI). In the attack phase, they achieve binary classification by computing the Euclidean distance between the collected PoI waveforms and the two waveform templates. If each PC oracle query is correct, then Ravi et al. need 5 queries to recover a coefficient. In total, they need $256 \times 2 \times 5 = 2560$ queries to recover Kyber512.

After that, Qin et al. improved the query efficiency by using an optimal binary tree similar to Hoffman tree encoding to reduce the average number of queries to recover Kyber512 to 1312, which can be found in [13].

We call all the above recovery methods *single-feature recovery*, and if the value of the private key cannot be found based on the private key identifier obtained from a set of oracle queries, we call this case an invalid attack.

This type of key recovery approach designed by them cannot always tell the truth due to the influence of ambient noise and the accuracy of the side channel distinguisher itself. And since we cannot determine the location of the error, the complexity of brute force cracking is quite high. Therefore, additional techniques are needed to enhance the recovery procedure or tolerate the error. One commonly used technique is majority voting, which was also used in the Ravi et al. attack. With multiple votes, we can obtain a more accurate Oracle.

2.3 Convolutional neural network in SCAs

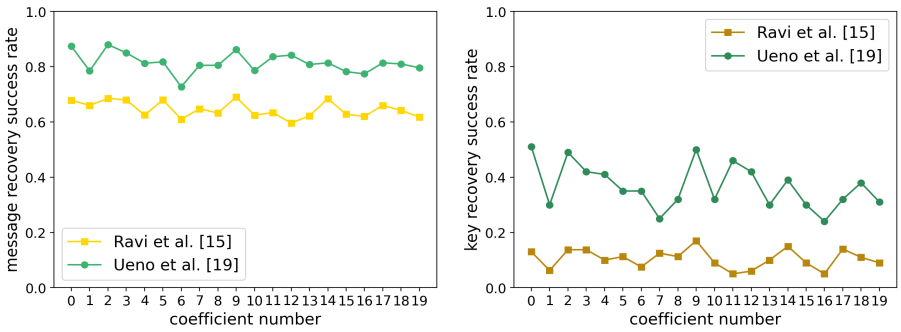
Convolutional neural networks are a powerful class of neural networks designed for processing image data. It has achieved widespread success across domains, including side-channel analysis. It is not surprising, as deep learning excels at identifying patterns and relationships, which aids in extracting information from power consumption time series. This is especially useful for template attacks.

In [11], Maghrebi et al. first applied deep learning in a side-channel context. They found that against unprotected cryptographic algorithm implementations, DL-based attacks are more effective than machine learning-based and traditional template attacks. Notably, their experimental results show that the feature extraction-based model performed very well on both datasets. This could be explained by the fact that CNN applies a nice features extraction technique based on filters allowing dealing with the most informative samples from the processed traces. The work of [8] also proves this.

At CHES 2022, Ueno et al. used CNN to design a side-channel distinguisher and achieve a similar binary classification [19]. With the CNN distinguisher, they can get higher accuracy of single-feature recognition.

2.4 Open problem

We reproduce the method of Ravi and Ueno in [15] and [19] using energy analysis. As an example, 20 coefficient values are recovered, as shown in Fig. 1, the average success rate for recovering a single-feature (i.e., message $\mathbf{m}'$) using Ravi's method is 64.58%. However, for recovering the complete label, the entire attack fails even if one-bit feature is incorrectly recovered. Hence, the average success rate of secret key recovery using the method in [15] is only 10.46%.



(a) recover message success rate. (b) recover key success rate.

Fig. 1. The success rates of using Ravi [15] and Ueno [19] methods in recovering message bit and a certain secret key coefficient respectively.

As illustrated in Fig. 1, using Ueno's method in [19], the CNN model leads to significant performance gains, with the average success rate of recovering message $\mathbf{m}'$ directly improved from 64.58% to 81.4%. However, the success rate of secret key recovery using the method in [19] remains only 36.7%.

We also noticed that with both methods in [15] and [19], this attack approach of recovering the secret key value bit-by-bit according to the single-feature of $\mathbf{m}'$ has a very large invalid attack space. That is, the recovered binary label string may represent neither the correct secret key value nor the wrong secret key value, but rather a meaningless label string. Shockingly, the average occurrence probability of invalid attacks at 75.17% in [15], shown in Fig. 2. Although using CNN in [19] reduces the occurrence of this event, the proportion of invalid attacks still reaches over 50%.

So how to improve the success rate of attacks and avoid such invalid attacks?

3 Multi-feature-based SCA on Kyber

In this section, we elucidate in detail the methodology for constructing multi-features of secret key and use it to recover Kyber768 using power analysis attacks. Using this approach, we eliminate the occurrence of invalid attacks.

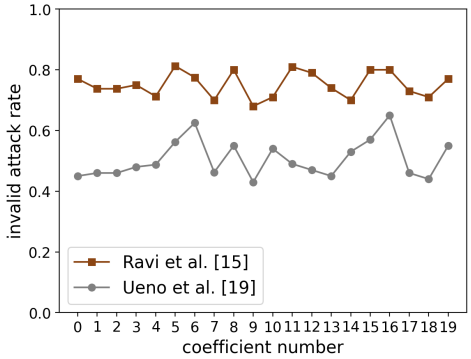


Fig. 2. Invalid attack rate in Ravi [15] and Ueno [19].

3.1 Construction of multiple features

In this part, we describe the full-key recovery framework of the new attack.

All previous attack methods take recovering $\mathbf{m}'$ as an intermediate step (including [15] and [19]), with the decrypted message value $\mathbf{m}'$ as the profiling target. In contrast, our approach bypasses this intermediate process and directly builds templates for the key. The comparison between the two approaches is illustrated in Fig. 3.

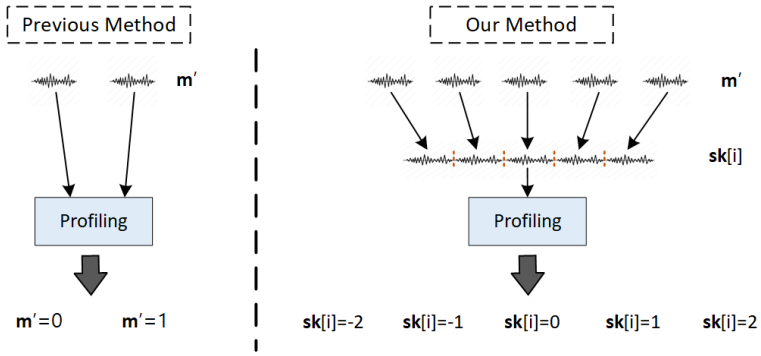


Fig. 3. Our profiling strategy.

In order to eliminate the invalid attack presented above, we propose a new profiling method that builds templates for secret keys from multi-features. We integrate the modeling and matching of $\mathbf{m}'$ and build a template for the secret key instead of the decrypted message $\mathbf{m}'$. Instead of recovering the key's binary label bit by bit, the new key feature construction method stitches single-features

$\mathbf{m}'$ together based on a specific ciphertext query result. Compared to *single-feature recovery*, we absorb the invalid attack space into the guess space for the entire secret key value, avoiding such situation.

3.2 Our attack scenario

We denote the i-th coefficient of the private key polynomial as $\mathbf{sk}[i]$. The overall workflow of the profiling stage and attack stage are shown in Fig. 4 and Fig. 5, respectively. We assume the adversary can manipulate the target device and collect the leaked power traces during cryptographic operations.

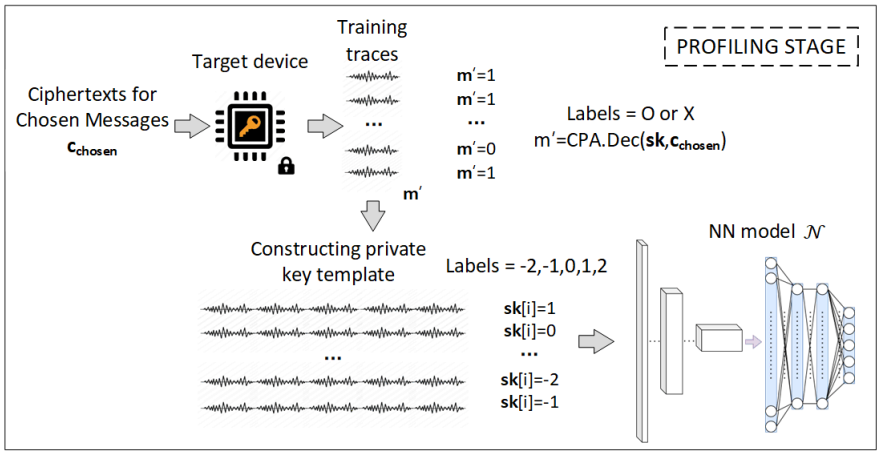


Fig. 4. Profiling stage of the Multi-feature-based SCA of key recovery. The NN model learns to find the combined message bit $\mathbf{m}'$.

By querying the PC oracle with constructed ciphertexts multiple times, the attacker obtains a set of pre-modeled power traces with the decrypted message $\mathbf{m}'$ being 0 or 1. Based on the mapping between the chosen ciphertexts and the private key values, the adversary acquires the multivariate feature labels representing the coefficients of the private key polynomial. Using the multivariate feature identifiers for each private key value, we construct the modeled power traces for $\mathbf{sk}[i]$ and label these traces based on the value of $\mathbf{sk}[i]$. Finally, they are fed into the network for training.

During the attack stage, as shown in Fig. 5, the attacker replaces the ciphertext with five preset chosen ciphertexts and polls the decrypted messages $\mathbf{m}'$ from the target device by decrypting these five chosen ciphertexts. After that, the five obtained traces are concatenated in order and preprocessed into the $\mathbf{sk}[i]$ template style during the modeling stage. Finally, the preprocessed power trace is fed into the trained network, which will directly output the value of this $\mathbf{sk}[j]$.

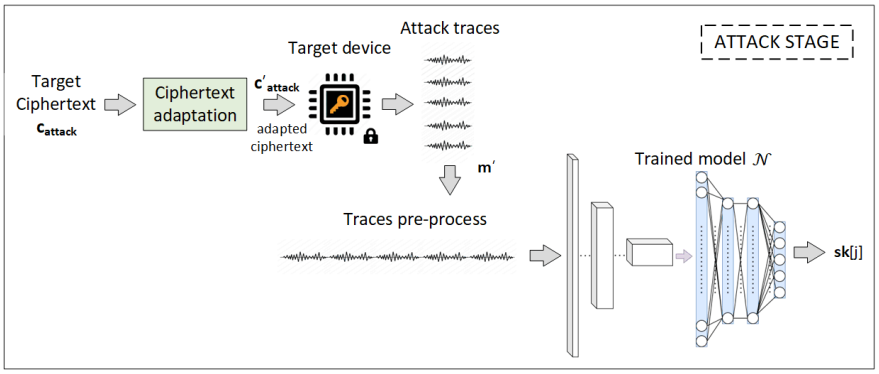


Fig. 5. Attack stage of the Multi-feature-based SCA of key recovery.

3.3 Generate qualified ciphertexts

The process of obtaining these five chosen ciphertexts is as follows:

In CCA.Decaps of Algorithm 2, an attacker can construct the ciphertext $\mathbf{c} = (\mathbf{u}, \mathbf{v})$. And set $\mathbf{u} = k_u \cdot x^0$ and $\mathbf{v} = k_v \cdot x^0$ where $(k_u, k_v) \in \mathbb{Z}_q$.

Let us take the example of recovering $\mathbf{sk}[0]$ (i.e., the lowest coefficient in the first polynomial of $\mathbf{sk}$). We take a long rectangle to represent a polynomial, and each small rectangle in it represents a coefficient. In Kyber768, the polynomial vector has three dimensions, so $\mathbf{sk}$ and $\mathbf{u}'$ each have three long rectangles. We omit certain modules, such as `Compress` operations. The connection between the decrypted $\mathbf{m}'$, the selected ciphertext $\mathbf{c}$ and the secret key is as shown in Fig. 6:

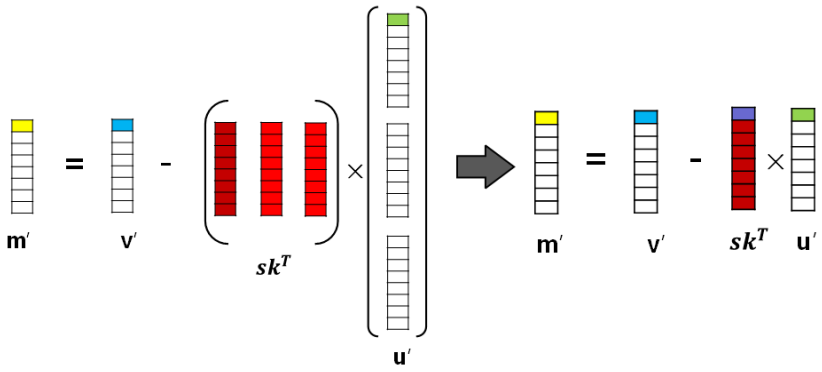


Fig. 6. The abstract compute relation for $\mathbf{m}'$ in line 5 in Algorithm 2. We fill the nonzero coefficients of each polynomial in $\mathbf{m}'$, $\mathbf{sk}$, $\mathbf{u}'$, and $\mathbf{v}'$ with a different color, with a white rectangle indicating that the coefficient is 0.

From Fig. 6, we can see that all coefficients in $\mathbf{m}'$ except for the lowest coefficient, the remaining are all zeros. This allows the attacker to establish a binary distinguishing identity for $\mathbf{sk}[0]$ by controlling $\mathbf{m}' = 0/1$. By instantiating this binary plaintext checking mechanism through the side channel, $\mathbf{sk}[0]$ can be recovered through multiple queries, and the remaining coefficients of $\mathbf{sk}$ can be recovered by exploiting the rotational property of polynomial multiplication in the ring.

Therefore, for the above selected $\mathbf{u}, \mathbf{v}$ (i.e., $\mathbf{u} = k_u \cdot x^0, \mathbf{v} = k_v \cdot x^0$), the lowest bit of the decrypted message $\mathbf{m}'[0]$ can be expressed as:

$$\mathbf{m}'[0] = \begin{cases} k_v - k_u \cdot \mathbf{sk}[0] & \text{if} \quad t = 0 \\ k_v - k_u \cdot -\mathbf{sk}[n-t] & \text{if} \quad 0 < t \leq n-1 \end{cases} \tag{3}$$

By iterating through the positions of t from 0 to $n-1$, we can recover the coefficients of the first polynomial in secret key s in the order of $\mathbf{sk}[0], -\mathbf{sk}[n-1], -\mathbf{sk}[n-2], \ldots, -\mathbf{sk}[1]$.

Since the coefficient values of the secret key in Kyber768 are within $[-2, 2]$, we construct Table 2 to enumerate the mapping between the binary string representation of the decrypted message from a chosen ciphertext and the corresponding secret key value. Where X represents the decrypted $\mathbf{m}' = 1$, and O represents $\mathbf{m}' = 0$.

Table 2. Chosen ciphertext pairs

	(k_u, k_v)						
Coeff.	$(0,0)$	$(0,q/2)$	$(110,657)$	$(240,2933)$	$(110,832)$	$(182,2497)$	$(416,1248)$
-2	O	X	X	O	X	O	X
-1	O	X	O	O	X	O	X
0	O	X	O	O	O	O	X
1	O	X	O	O	O	O	O
2	O	X	O	X	O	X	O

Traces pre-process. As obtained above, Table 2 provides a unique binary label string mapping to each secret key value. Our new profiling approach directly builds templates from this label string to the range of secret key values, instead of mapping the profiled $\mathbf{m}' = 1$ and $\mathbf{m}' = 0$ to the binary representation. This expands the original binary message recognition into a 5-class secret key value recognition problem. In the attack phase, we iterate through the five ciphertexts constructed using Table 2 (last five columns), collecting the power traces over the last four rounds of the hash function during decapsulation for each ciphertext. These are concatenated to form the combined multivariate feature information.

4 Experiments

4.1 Equipment setup

Our measurement setup is shown in Fig. 7. It consists of the Laptop, the versatile current amplifier, the STM32F429 target board, and the PicoScope 3403D Oscilloscope. We target the optimized unprotected implementation of Kyber768, taken from the public `pqm4` library [7], a benchmarking and testing framework for PQC schemes on the 32-bit ARM Cortex-M4 microcontroller. In our initialization, the implementation is compiled with **arm-none-eabi-gcc** using the optimization flag "-O1". We set the operating clock frequency of the target board to 16 MHz and utilized the power analysis side-channel for our experiments. For traces acquisition, we set the trigger at pin PC6, and the measurement results were collected on the oscilloscope with a sampling rate of 62.5 MSam/s.

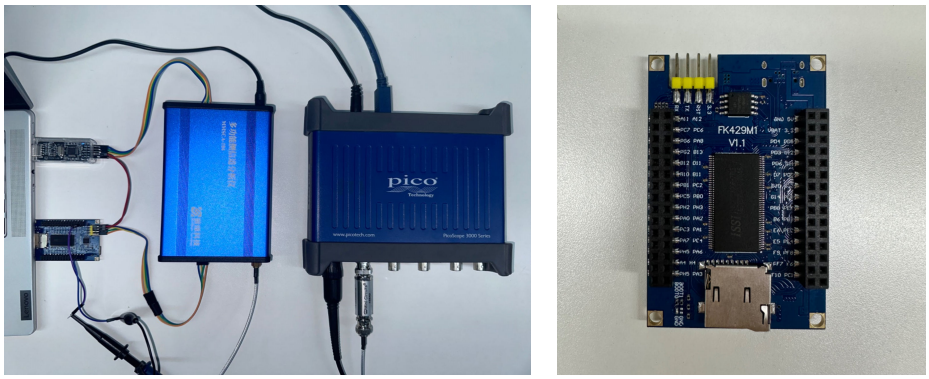


Fig. 7. Equipment for trace acquisition and the board used in the experiment.

4.2 Target operation

The ensuing problem is how to capture this leakage in the side channel. We assume that the attacker has the ability to completely manipulate the target device and is able to measure the power consumption during the execution of a cryptographic algorithm. Then during the inference phase, the adversary aims at recovering the unknown secret key, processed by the same device, by collecting a new set of power consumption traces. To guarantee a fair and realistic attack comparison, we stress the fact that the training and the attack data sets must be different.

Target Operation. Using the key recovery methods in Sect. 3, we find a chosen ciphertext correspondence that is sufficient to distinguish the values of the polynomial coefficients of the secret key. By means of the binary plaintext checking

oracle described above, the attacker constructs a distinction of the decrypted message $\mathbf{m}'$. Exactly through the hash function execution process in the FO transformation, the attacker can amplify the difference of the decrypted message $\mathbf{m}'$ from 1 bit message bit to 256 bits.

The `KeccakF1600_StatePermute` function in $\mathcal{G}$ includes twelve for loops. Therefore, the target option we choose is the last four rounds of the hash operation, as shown in Fig. 8. That is, line 6 in Algorithm 2. The TVLA result of our target operation is as shown in Fig. 9:

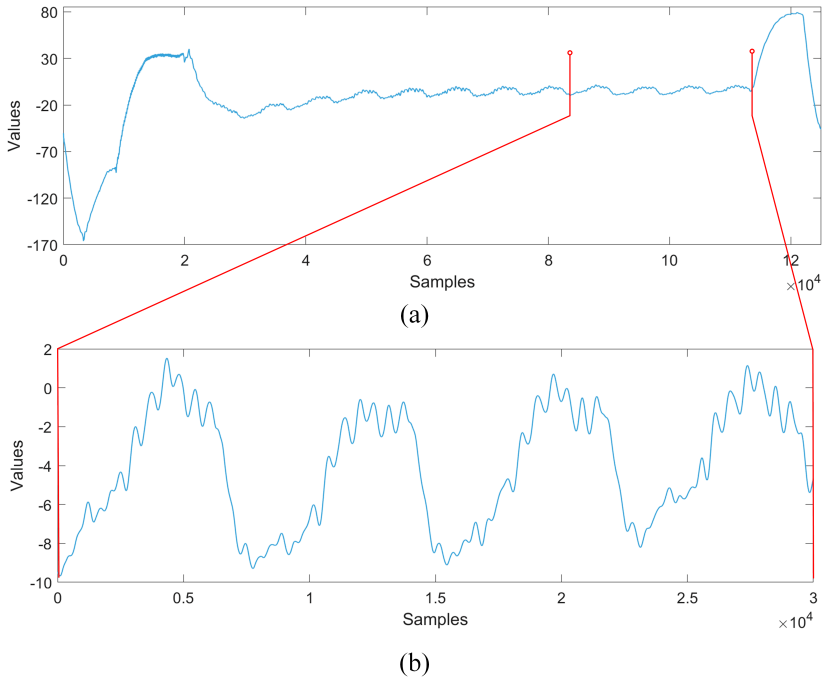


Fig. 8. Original power trace of Kyber768. (a) The whole hash operation $\mathcal{G}$ with twelve for loops in Kyber.KEM.Decaps() (i.e., line 6 in Algorithm 2); (b) The last four rounds of $\mathcal{G}$.

Traces Acquire. We set the STM32F429 microcontroller as a server and our laptop as a client. Every time we selected a random message $\mathbf{m}$ and encapsulated it with the public key into ciphertext $\mathbf{c}$ on the client, then we sent $\mathbf{c}$ to the server through a socket.

During the decapsulation of the profiling stage, we captured power traces and saved O or X (i.e., $\mathbf{m}' = 0$ or $\mathbf{m}' = 1$) as labels. For each type of template, we collected 9,000 traces, each with a length of 30,000. Then we combined the

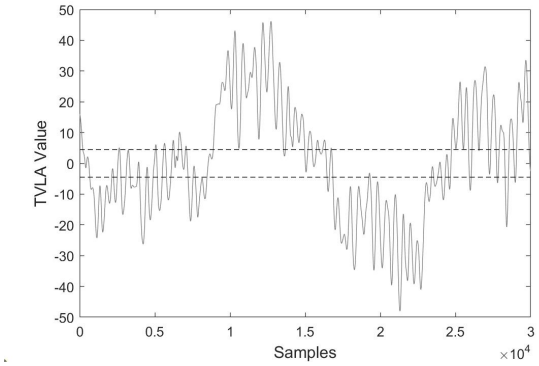


Fig. 9. TVLA results for the last four rounds between O and X.

traces in order of the five chosen ciphertexts in Table 2. The templates we get are as shown in Fig. 10, and we only selected two localized positions for zoomed-in display (five values of $\mathbf{sk}[i]$ are represented by five lines with different colors respectively):

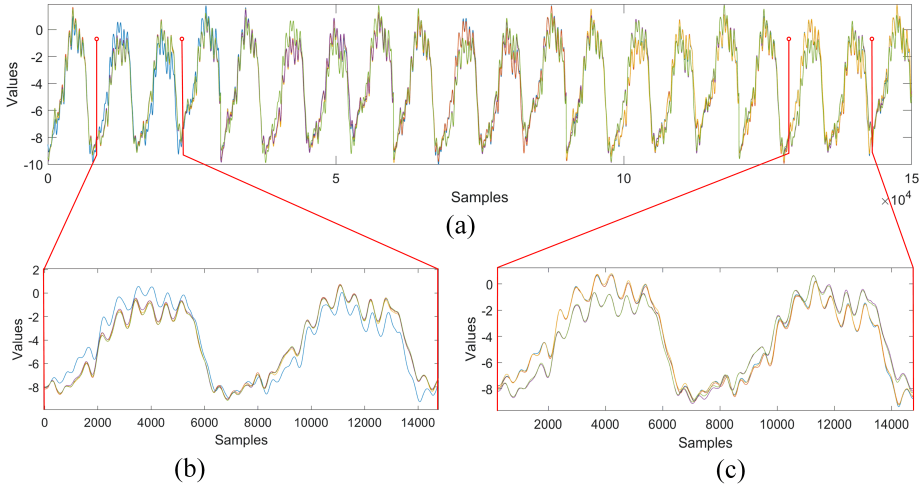


Fig. 10. Constructed template of $\mathbf{sk}[i]$. (a) Complete template for $\mathbf{sk}[i]$ after trace pre-process; (b) and (c) The expansion of an interval somewhere in the template of $\mathbf{sk}[i]$.

In the attack stage, we only need to poll these five chosen ciphertexts in order and collect the same power traces as in the profiling stage for the same pre-processing.

4.3 Model Training

By adjusting the CNN network architecture and hyperparameters, we obtained the CNN model that performs best on our dataset. This model is inherited from [19]. The architecture of which is shown in Table 3. It has seven convolutional layers and four fully-connected layers. In the Function row, conv1d(F) denotes the operation at each layer and F is the filter size. The stride of the filter is two and the padding of it is one. After each convolutional layer, batch normalization and SeLU activation are used, and finally, a 2×2 size average pooling layer is connected to reduce the dimensionality. The convolutional layers are followed by four fully-connected layers in our network architecture. The first fully-connected layer consists of 1000 neurons. Then followed by two fully-connected layers with 200 neurons each. The final layer has 5 neurons and utilizes `softmax` activation for classification.

Table 3. NN architecture

	Input	Output	Function	Normalization	Activation	Pooling
$Conv1$	150000×1	4	conv1d(3)	Yes	`SELU`	Avg(2)
$Conv2$	75000×4	4	conv1d(3)	Yes	`SELU`	Avg(2)
$Conv3$	37500×4	4	conv1d(3)	Yes	`SELU`	Avg(2)
$Conv4$	18750×4	8	conv1d(3)	Yes	`SELU`	Avg(2)
$Conv5$	9375×8	8	conv1d(3)	Yes	`SELU`	Avg(2)
$Conv6$	4687×8	8	conv1d(3)	Yes	`SELU`	Avg(2)
$Conv7$	2343×8	8	conv1d(3)	Yes	`SELU`	Avg(2)
$Flatten$	1171×8	9368	flatten	-	-	-
$FC1$	9368	1000	dense	-	`SELU`	-
$FC2$	1000	200	dense	-	`SELU`	-
$FC3$	200	200	dense	-	`SELU`	-
$FC4$	200	5	dense	-	`Sigmoid`	-

In the following experiments, we employed CUDA 11.6, cuDNN 8.3.0, and Pytorch-gpu 1.13.1 on NVIDIA GeForce GTX 3050 to carry out the NN training. The Adam optimizer is utilized with a learning rate of 0.00005, the batch size was 128, and the number of epochs was 50. We used the cross-entropy loss function during training and validated it after each epoch.

4.4 Experimental results and comparison

The loss values of this model trained on our dataset for 50 epochs and the accuracy of the validation set are shown in Fig. 11. After 50 epochs of training, the model's loss stabilizes around 0.9 and the accuracy of the validation set improves to 88%.

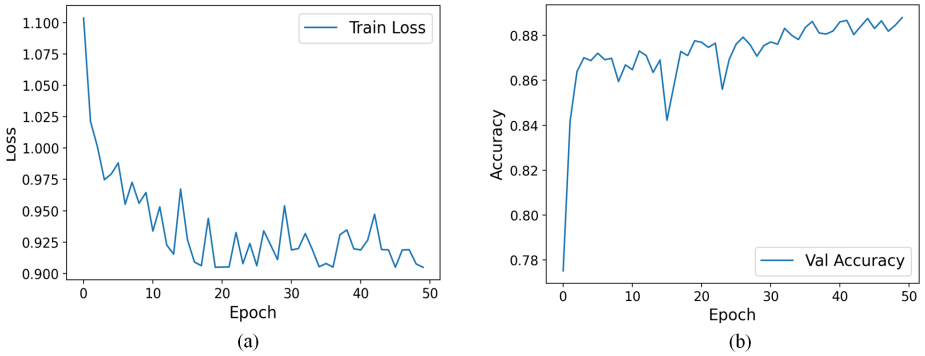


Fig. 11. Train loss (a) and validation accuracy (b) of our approach..

As shown in Fig. 12, our approach significantly improves the success probability of recovering secret key values. Compared to Ravi's method [15], the average attack success rate for a secret key value increases by 53.69%. It also outperforms distinguishing message $\mathbf{m}'$ using neural networks [19] by 27.45%. Our approach can also tolerate invalid attacks due to errors in *single-feature recovery*.

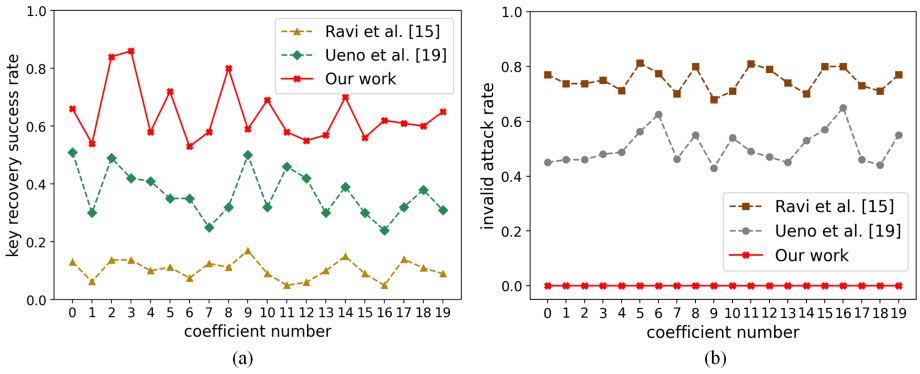


Fig. 12. Compare three methods of key recovery success rate (a) and invalid attack rate (b).

5 Conclusion

Our Multi-feature-based SCA is a novel attack technique that extracts secret key templates from multivariate features and employs the optimal CNN architecture. All attacks presented in this paper are performed directly on the target device. Our experimental results demonstrate that CNN can significantly improve profiling efficiency as an effective approach. Notably, our approach only uses the traces collected in a single experiment when recovering the secret key. Based on the results, voting across multiple experiments can achieve 100% attack success rate. Our work reiterates the need for effective countermeasures against side-channel attacks in cryptographic implementations.

Acknowledgements. This work is supported by National Key R&D Program of China (No. 2022YFB3103800), and the National Natural Science Foundation of China under Grant 62272457. We thank the anonymous reviewers for their valuable comments and suggestions.

References

1. Announcing the commercial national security algorithm suite 2.0. Tech. rep. (2022)
2. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., et al.: Status report on the third round of the nist post-quantum cryptography standardization process. US Department of Commerce, NIST (2022)
3. D'Anvers, J.P., Tiepelt, M., Vercauteren, F., Verbauwhede, I.: Timing attacks on error correcting codes in post-quantum schemes. In: Proceedings of ACM Workshop on Theory of Implementation Security Workshop. pp. 2–9 (2019)
4. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of cryptology **26**, 80–101 (2013)
5. Gidney, C., Ekerå, M.: How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. Quantum **5**, 433 (2021)
6. Guo, Q., Johansson, T., Nilsson, A.: A key-recovery timing attack on post-quantum primitives using the fujisaki-okamoto transformation and its application on frodokem. In: Annual International Cryptology Conference. pp. 359–386. Springer (2020)
7. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: Pqm4: Post-quantum crypto library for the arm cortex-m4 (2019)
8. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 148–179 (2019)
9. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Advances in CryptologyCRYPTO96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16. pp. 104–113. Springer (1996)
10. Koeune, F., Standaert, F.X.: A tutorial on physical security and side-channel attacks. International School on Foundations of Security Analysis and Design pp. 78–108 (2004)

11. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings 6. pp. 3–26. Springer (2016)
12. Moody, D.: Post-quantum cryptography standardization: Announcement and outline of nists call for submissions. In: International Conference on Post-Quantum Cryptography-PQCrypto (2016)
13. Qin, Y., Cheng, C., Zhang, X., Pan, Y., Hu, L., Ding, J.: A systematic approach and analysis of key mismatch attacks on lattice-based nist candidate kems. In: Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV 27. pp. 92–121. Springer (2021)
14. Ravi, P., Roy, S.S.: Side-channel analysis of lattice-based pqc candidates. In: Round 3 Seminars, NIST Post Quantum Cryptography (2021)
15. Ravi, P., Roy, S.S., Chattopadhyay, A., Bhasin, S.: Generic side-channel attacks on cca-secure lattice-based pke and kems. IACR transactions on cryptographic hardware and embedded systems pp. 307–335 (2020)
16. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J., Seiler, G., Stehlé, D.: Crystals-kyber: Algorithm specifications and supporting documentation (version 3.0). NIST Post-Quantum Cryptography–Round **3**
17. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)
18. Sim, B.Y., Kwon, J., Lee, J., Kim, I.J., Lee, T.H., Han, J., Yoon, H., Cho, J., Han, D.G.: Single-trace attacks on message encoding in lattice-based kems. IEEE Access **8**, 183175–183191 (2020)
19. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/em analysis on post-quantum kems. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 296–322 (2022)
20. Wang, R., Ngo, K., Dubrova, E.: A message recovery attack on lwe/lwr-based pke/kems using amplitude-modulated em emanations. In: International Conference on Information Security and Cryptology. pp. 450–471. Springer (2022)
21. Xu, Z., Pemberton, O., Roy, S.S., Oswald, D., Yao, W., Zheng, Z.: Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. IEEE Transactions on Computers **71**(9), 2163–2176 (2021)