Chosen-Ciphertext Clustering Attack on CRYSTALS-KYBER using the Side-Channel Leakage of Barrett Reduction

Bo-Yeon Sim, Aesun Park, *Member, IEEE,* and Dong-Guk Han

***Abstract*—This study proposes a chosen-ciphertext side-channel attack against a lattice-based key encapsulation mechanism (KEM), the third-round candidate of the national institute of standards and technology (NIST) standardization project. Unlike existing attacks that target operations such as inverse NTT and message encoding/decoding, we target Barrett reduction in the decapsulation phase of CRYSTALS-KYBER to obtain a secret key. We show that a sensitive variable-dependent leakage of Barrett reduction exposes an entire secret key. The results of experiments conducted on the ARM Cortex-M4 microcontroller accomplish a success rate of 100%. We only need six chosen ciphertexts for KYBER512 and KYBER768 and eight chosen ciphertexts for KYBER1024. We also show that the m4 scheme of the pqm4 library, an implementation with the ARM Cortex-M4 specific optimization (typically in assembly), is vulnerable to the proposed attack. In this scheme, six, nine, and twelve chosen ciphertexts are required for KYBER512, KYBER768, and KYBER1024, respectively.**

***Index Terms*—Lattice-based cryptography, key decapsulation mechanism, Barrett reduction, side-channel attack, chosen-ciphertext attack.**

I. Introduction

By 2025, it is expected that there will be more than 30 billion Internet of things (IoT) connections, almost 4 IoT devices per person on average [1]. In addition, the demand for IoT security is increasing, and the global IoT security market size is expected to increase to more than 20.8 billion by 2025 [2]. Accordingly, establishing a trustworthy IoT infrastructure that ensures information protection is essential. Five areas, including cloud-based IoT security, have been reported that are particularly important for companies looking to secure their IoT devices and assets.

A key encapsulation mechanism (KEM), a public-key cryptosystem for generating a shared secret key between two parties, is needed to establish cloud-based peer-to-peer secure transactions. Diffie-Hellman (DH), Rivest-Shamir-Adleman (RSA), and elliptic curve cryptography (ECC) have been mainly used; however, they are insecure under quantum computer attacks [3]. Hence, if a large-scale quantum computation is realized, KEMs become vulnerable. Experts estimate that RSA, with a public-key size of 2000-bit, will not guarantee safety until 2030 [4]–[6].

To address this issue, the national institute of standards and technology (NIST) is working on the post-quantum cryptography (PQC) standardization project [7]. The third-round candidates (seven finalists and eight alternatives) of the NIST PQC project were notified on July 22, 2020 [8]. Accordingly, fifteen (seven, excluding alternatives) candidates were selected in the third-round of the NIST PQC project, and nine (four, excluding alternatives) of them are public-key encryption (PKE)/KEMs [8]. Lattice-based KEMs have got increasingly concerned due to their balanced performance in size and speed. Among the third-round KEM candidates, five (three, excluding alternatives) schemes are lattice-based KEMs [9]–[13]. They are classified into two types: the schemes based on the learning with error (LWE)/learning with rounding (LWR) problem [9]–[11], and the schemes based on the NTRU problem [12], [13]. CRYSTALS-KYBER, SABER, and FrodoKEM belong to the first class, whereas NTRU and NTRU Prime belong to the second.

Even if a cryptographic scheme is secure against mathematical analysis owing to the hardness of the mathematical problem, it is subject to side-channel attacks (SCAs). It was first discovered by Paul Kocher in 1996 [14], and many cryptographic schemes have been easily broken by SCAs. SCAs allow recovering secret information (*e.g.*, a cryptographic key) using physically measured side-channel information. Side-channel information includes consumed power, radiated electromagnetic wave, emitted sound, and executed time while the cryptographic device operates. Therefore, SCAs are considered major threats to the implementations of cryptographic schemes, especially for applications in embedded devices. Recently, the investigation of SCAs for PQC has attracted increasing attention in connection with the NIST PQC project. Given that most of the candidates are implemented to execute constant time, simple timing attacks that measure only execution time can be prevented. Even if the algorithms have a constant time implementation, they can be vulnerable to the other SCAs, such as power analysis and electromagnetic analysis. Not only are many researchers finding SCA vulnerabilities for PQC implements, but NIST also noted that implementations addressing SCAs are more

Manuscript received April 19, 20XX; revised August 26, 20XX.

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2021-0-00724, RISC-V based Secure CPU Architecture Design for Embedded System Malware Detection and Response)

Bo-Yeon Sim is with the Department of Intelligent Convergence Research Laboratory, Electronics and Telecommunications Research Institute, Daejeon, 34129, Republic of Korea (email: sboyeon37@etri.re.kr).

Aesun Park is with the Defense Security Support Command, Gwacheon, 13820, Republic of Korea (email: aesons@dssc.mil.kr).

Dong-Guk Han is with the Department of Information Security, Cryptology, and Mathematics, Kookmin University, Seoul, 02707, Republic of Korea (email: christa@kookmin.ac.kr).

Corresponding author: Dong-Guk Han.

meaningful than those that do not [15]. Therefore, various SCAs related to PQC are being studied to verify the side-channel resistance of PQC [16]–[38].

Most IoT devices come with limited resources, *i.e.*, power constraints, strict memory, and chip area. Currently, NIST officially requires performance evaluations of PQC's software implementations on ARM Cortex-M4 microcontrollers available in a wide range of IoT devices. Accordingly, the open-source library pqm4, the testing and benchmarking framework for PQC schemes operating on the ARM Cortex-M4 microcontroller, was initiated by the PQCRYPTO project (ICT-645622) funded by European Commission in the H2020 program [39]. The pqm4 library is specifically optimized for the ARM Cortex-M4 microcontroller. Therefore, to use IoT devices secure against SCAs must involve verifying the side-channel vulnerability against the pqm4 library.

A. Related works

Lattice-based KEMs have been studied for different types of SCAs vulnerability. Especially, several studies about side-channel assisted chosen-ciphertext attacks (CCAs), which recover the secret key, have been conducted [30]–[38]. CCAs on various operations, such as error-correcting codes, inverse NTT, message encoding/decoding, and Fujisaki-Okamoto (FO) transform, have been studied.

D'Anvers *et al.* [31] reported that the Ring-LWE scheme LAC's secret key leaked by exploiting variable runtime of error-correcting codes in decryption. They used less than 2^{16} decryption queries to recover the secret key. The following year, Ravi *et al.* [32] proposed generic side-channel-assisted CCAs on six lattice-based KEMs. They used binary information about the message through EM leakage in error-correcting procedures and FO transforms to perform key recovery. Their attacks could also be applied to implementations operate in a constant time.

More recently, Xu *et al.* [34] showed that an attacker with complete knowledge of the decrypted message for chosen ciphertexts could perform the full key recovery using small decapsulation queries for KYBER512. They targeted the inverse NTT for the clean scheme and the message encoding function for the m4 scheme. Four and eight decapsulation queries were used to recover the secret key for the clean and m4 schemes, respectively. Ravi *et al.* [35] demonstrated side-channel assisted message recovery attacks which target storage of the decrypted message in memory. In more detail, they exploited the fact that the decrypted message is stored one bit at a time. That is, it is possible to restore a message by comparing the Hamming weight of the message stored immediately before. As a result, the full message recovery of KYBER512 was possible with a single trace (actually, the success rate ramps to 98.24% with 5 averaged traces), but this method required 128k traces to profiling. Another method they proposed was to recover the message by using the targeted flip of message bits and the cyclic message rotation technique. In the presence of a side-channel Hamming weight classifier, this technique required $(w+1)$ traces to recover the full message where w is the storage width. They mentioned that implementations with shuffling and masking countermeasures could also be attacked. Unfortunately, attacking protected implementations requires a strong attack assumption that an attacker can turn off or deactivate the countermeasure. They also proposed the recovered message-based key recovery attack. Six chosen ciphertexts are needed to recover the secret key of KYBER512. However, the specification of CRYSTALS-KYBER was updated and the noise parameter of KYBER512 was increased [40]; thus, it is obvious that more chosen ciphertexts are needed than the number stated in [34], [35].

Ngo *et al.* [36] proposed the first side-channel attack on a first-order masked SABER. They used the incremental storage leakage presented in [35] and applied deep learning-based power analysis. Extracting the random mask at each execution was unnecessary because the input trace contains both where the shares $m \oplus r$ and r were computed. Thus, they could improve success probability by combining score vectors of the multiple-trace attack. The $[8,4,4]_2$ extended Hamming codes were applied to improve key-recovery attack, and sixteen chosen ciphertexts were used for LightSaber.

Although CCAs on various operations have been studied, no study has been conducted on reductions. The input value of the reduction in decryption is also affected by the secret key; thus, it can lead to attacks that use CCA to derive the secret key. Xu *et al.* [34] mentioned that operations after the inverse NTT could be vulnerable; however, they did not perform a detailed analysis. Additionally, the output of the inverse NTT can have various values; thus, there are many restrictions on finding a valid chosen ciphertext. Xu *et al.* presented that fifteen possible binary classifiers and forty possible ternary classifiers exist. The incremental storage leakage used in [35] relies only on the 1-bit value of the decoded message, requiring average preprocessing to increase the signal-to-noise ratio (SNR). Moreover, template generation is necessary for attacks. These works motivated us to investigate a new attack position that constructing chosen ciphertexts is more efficient and can maximize the side-channel leakage.

B. Main Contributions

In this study, we focus on a lattice-based KEM corresponding to the third-round candidate of the NIST PQC standardization project. Specifically, we present a comprehensive analysis and the corresponding experiment results on CRYSTALS-KYBER by focusing on Barrett reduction in the decapsulation phase, which was not considered a target operation against SCA-based chosen-ciphertext attacks. The main contributions of this study can be summarized as follows.

We introduce a chosen-ciphertext clustering attack using the side-channel leakage of Barrett reduction in the decapsulation phase. The obtained experimental results show that we can recover the full secret key using six chosen ciphertexts for KYBER512. In the ref, clean, and opt schemes, six and eight chosen ciphertexts are needed for KYBER768 and KYBER1024, respectively. In the m4 scheme, nine and twelve chosen ciphertexts are needed, respectively. Our target intermediate value can have only three values, and more than

3154 valid chosen ciphertexts exist. Moreover, the maximum difference in leakage would be noise resistant because it is proportional to 13, which is the Hamming distance between the two intermediate values. Therefore, averaging is not required to increase SNR, and template building is also unnecessary.

C. Organization

The rest of the paper is organized as follows. In Section II, we briefly explain the specification of CRYSTALS-KYBER. We explain the proposed chosen-ciphertext clustering attack methodology in Section III, and we show experimental results in Section IV. In Section V, we recommend countermeasures. Finally, we summarize the conclusions in Section VI.

II. Preliminaries

A. Notation

- Let n and q be positive integers.
- Let $\mathcal{R}$ be a base ring defined as $\mathbb{Z}[x]/\langle x^n+1\rangle$. $\mathcal{R}$ can be represented as

$$\left\{\sum_{i=0}^{n-1} a_i x^i : a_i \in \mathbb{Z},\ 0 \le i \le n-1\right\}.$$

- Let $\mathcal{R}_q := \mathcal{R}/q\mathcal{R}$. The quotient ring $\mathcal{R}_q$ can be represented as

$$\left\{\sum_{i=0}^{n-1} a_i x^i : a_i \in \mathbb{Z}_q,\ 0 \le i \le n-1\right\}.$$

- Bold lower-case letters $\mathbf{s}$ represent column vectors with coefficients s_i in $\mathcal{R}_q$, $0 \le i \le k-1$, *i.e.*, $\mathbf{s} \in \mathcal{R}_q^k$
- $\mathbf{s}^\intercal$ is transpose of a vector $\mathbf{s}$.

B. CRYSTALS-KYBER

CRYSTALS-KYBER [40] is a lattice-based KEM using a PKE scheme similar to the LPR encryption scheme suggested by Lyubashevsky, Peikert, and Regev [41]. It is based on a polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n+1\rangle$ of the dimension $n = 256$ and modulus $q = 3329$. The parameters k, p, and t are different according to the security level. Three parameter sets, namely, KYBER512, KYBER768, and KYBER1024, aim to support NIST security levels 1, 3, and 5, respectively.

For NIST security level 1, the first component of a ciphertext is of rank 2 over $\mathcal{R}_q$, *i.e.*, $k = 2$ in Algorithm 1. For NIST security levels 3 and 5, $k = 3$ and $k = 4$, respectively. The secret key is sampled from centered binomial distribution B_{η_1}. The parameter η_1 is 3, 2, and 2, according to the supported security level. Here, for NIST security levels 1 and 3, p and t for Compress and Decompress are set to be 2^{10} and 2^3, respectively. They are set to be 2^{11} and 2^5, respectively, for NIST security level 5. The bit length ℓ of message μ and shared key K is 256.

Hash1 and Hash2 are SHA3-256 and SHA3-512, respectively. KDF is implemented using SHAKE-256. Compress$_{q,\log p}(x)$ and Compress$_{q,\log t}(x)$ take an element $x \in \mathbb{Z}_q$ and output $\log p$- and $\log t$-bit integers, respectively. Decompress$_{q,\log p}(x)$ and Decompress$_{q,\log t}(x)$ take $\log p$- and $\log t$-bit integers, respectively, and output $y \in \mathbb{Z}_q$.

Algorithm 1 Message Decapsulation of CRYSTALS-KYBER (refer to [40])

```
Require: Ciphertext $c = (c_1 \| c_2) \in \mathcal{R}_p^k \times \mathcal{R}_t$
Require: Secret key $sk \in \mathcal{R}_q^k$
Require: Public key $pk = (a \in \mathcal{R}_q^{k\times k}, b \in \mathcal{R}_q^k)$
Require: Random value $z \in \{0,1\}^\ell$
Ensure:  Shared key $K \in \{0,1\}^\ell$
 1: /*Decryption*/
 2: $\mathbf{s} = sk$
 3: $\mathbf{u} = \mathsf{Decompress}_{q,\log p}(c_1)$
 4: $v = \mathsf{Decompress}_{q,\log t}(c_2)$
 5: $\mu' = \mathsf{decode}(v - \mathbf{s}^\intercal\mathbf{u} \bmod q)$
 6: /*=== FO transform ===*/
 7: $(\bar{K}', seed') = \mathsf{Hash2}(\mu' \| \mathsf{Hash1}(pk))$
 8: /*Encryption*/
 9: Sampling $r', e_1' \in \mathcal{R}_q^{k\times 1}$, and $e_2' \in \mathcal{R}_q$ using $seed'$
10: $c_1' = \mathsf{Compress}_{q,\log p}(ar' + e_1' \bmod q)$
11: $c_2' = \mathsf{Compress}_{q,\log t}(b^\intercal r' + e_2' + \mathsf{encode}(\mu') \bmod q)$
12: $c' = (c_1' \| c_2')$
13: /*Shared key derivation*/
14: if $c = c'$ then
15:     $K = \mathsf{KDF}(\bar{K}' \| \mathsf{Hash1}(c))$
16: else
17:     $K = \mathsf{KDF}(z \| \mathsf{Hash1}(c))$
18: end if
19: Return $K$
```

encode is message encoding that converts ℓ-bit message to a polynomial. decode is message decoding that is the inverse of encode. Algorithm 1 illustrates message decapsulation of CRYSTALS-KYBER. To construct the IND-CCA2-secure KEM, a slightly tweaked FO transform is applied on a CPA-secure PKE.

III. Proposed Chosen-Ciphertext Clustering Attack on CRYSTALS-KYBER

In this section, we propose a chosen-ciphertext clustering attack on CRYSTALS-KYBER using a sensitive variable-dependent leakage of Barrett reduction.

A. Sensitive Variable-dependent Leakage of Barrett Reduction

We target step 5 of Algorithm 1. We focus on the $v - \mathbf{s}^\intercal\mathbf{u} \bmod q$ operation, which calculates the input of decode.

Listing 1, Listing 2, and Listing 3 illustrated decryption, reduction, and Barrett reduction in CRYSTALS-KYBER, respectively. In Listing 1, skpv, bp, and v are $\mathbf{s}$, $\mathbf{u}$, and v described in Algorithm 1, respectively. At steps 12-14 of Listing 1, $\mathbf{s}^\intercal\mathbf{u}$ is calculated in the NTT domain, and Montgomery

```c
1  // Decryption function of the CPA-secure
2  void indcpa_dec(uint8_t m[KYBER_INDCPA_MSGBYTES],
3                  const uint8_t c[KYBER_INDCPA_BYTES],
4                  const uint8_t sk[
       KYBER_INDCPA_SECRETKEYBYTES])
5  {
6    polyvec bp, skpv;
7    poly v, mp;
8
9    unpack_ciphertext(&bp, &v, c);
10   unpack_sk(&skpv, sk);
11
12   polyvec_ntt(&bp);
13   polyvec_pointwise_acc_montgomery(&mp, &skpv, &bp);
14   poly_invntt_tomont(&mp);
15
16   poly_sub(&mp, &v, &mp);
17   poly_reduce(&mp);
18
19   poly_tomsg(m, &mp);
20 }
```

Listing 1. Decryption of CRYSTALS-KYBER (in C code)

```c
1  // Applies Barrett reduction
2  // to all coefficients of a polynomial
3  void poly_reduce(poly *r)
4  {
5    unsigned int i;
6    for(i=0;i<KYBER_N;i++)
7      r->coeffs[i] = barrett_reduce(r->coeffs[i]);
8  }
```

Listing 2. Reduction of CRYSTALS-KYBER (in C code)

```c
1  // given a 16-bit integer a, computes 16-bit integer
2  // congruent to a mod q in {0,...,q}
3  int16_t barrett_reduce(int16_t a)
4  {
5    int16_t t;
6    const int16_t v = ((1U << 26) + KYBER_Q/2)/KYBER_Q;
7
8    t  = (int32_t)v*a >> 26;
9    t *= KYBER_Q;
10   return a - t;
11 }
```

Listing 3. Barrett reduction of CRYSTALS-KYBER (in C code)

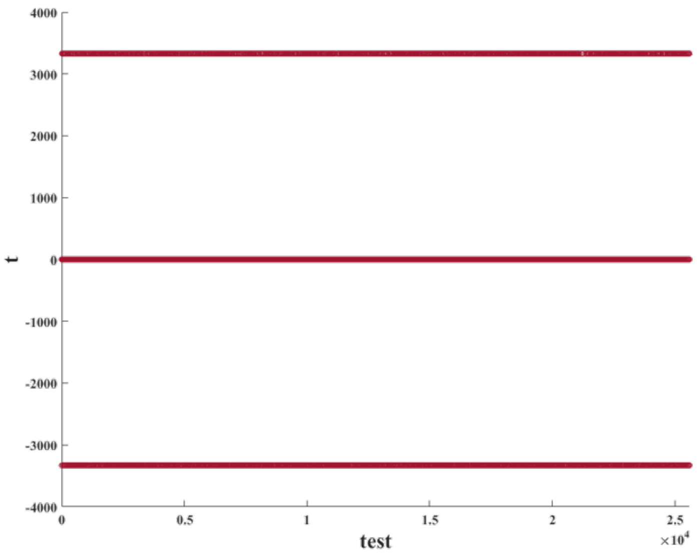


Fig. 1. t value

reduction is applied to the output. Hence, for the output polynomial mp of poly_invntt_tomont(), all coefficients mp_i satisfy

$$-3328 \leq mp_i \leq 3328.$$

For a polynomial v, all coefficients v_i satisfy

$$0 \leq v_i \leq 3328.$$

Accordingly, for the output polynomial mp of poly_sub() at step 16 of Listing 1, all coefficients mp_i satisfy

$$-3328 \leq mp_i \leq 6656.$$

Here, mp is $v - \mathbf{s}^\intercal \mathbf{u}$, and it is the input of poly_reduce().

As shown in steps 6-7 of Listing 2, Barrett reduction applies to all coefficients of the input polynomial mp. The intermediate value t at steps 9-10 of Listing 3 is described as follows.

$$\mathsf{t} = \begin{cases} 3329 & , if \quad 3329 \leq mp_i \leq 6656; \\ 0 & , if \quad 0 \leq mp_i < 3329; \\ -3329 & , if \quad -3328 \leq mp_i < 0. \end{cases}$$

We simulated CRYSTALS-KYBER KEM 100 times to verify the intermediate value t. The simulated results are shown in Figure 1. Given the dimension $n = 256$, the number of tests is 25,600. The intermediate value t is determined by one of three values depending on the coefficient of $v - \mathbf{s}^\intercal \mathbf{u}$, as shown in Figure 1. Given that $\mathbf{s}$ is a secret key, *i.e.*, sensitive variable, the intermediate value t can leak sensitive variable-dependent information.

B. Constructing Chosen Ciphertexts

We construct chosen ciphertexts to magnify the difference in the sensitive variable-dependent leakage of t depending on the coefficient value of $\mathbf{s}$. We establish criteria for constructing chosen ciphertexts as follow.

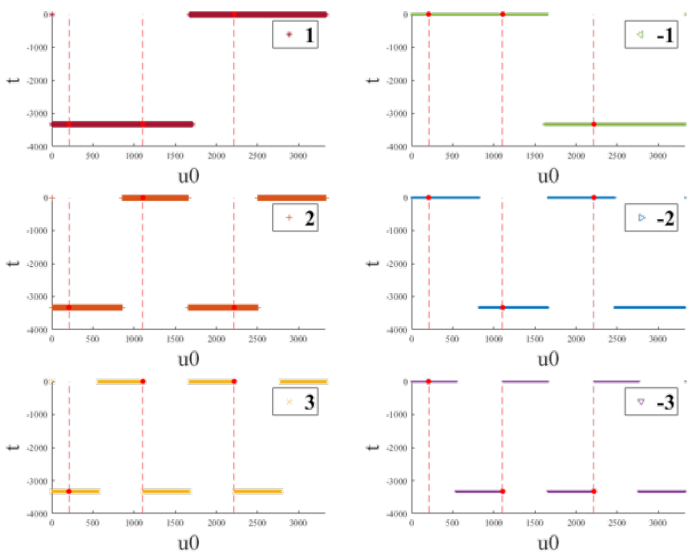


Fig. 2. t value according to u_0 value

1) Because the Hamming weight difference between 0 and -3329 is 13, which is the largest, we configure ciphertexts so that t is 0 or -3329.
2) t is configured so that only one coefficient value of the secret key is affected.

Let $\mathbf{s} = (s_0, \cdots, s_{k-1}) \in \mathcal{R}_q^k, \mathbf{u} = (u_0, \cdots, u_{k-1}) \in \mathcal{R}_q^k$. Here, s_i and u_i are polynomials in the ring $\mathcal{R}_q$. We denote $s_{i,j}$ and $u_{i,j}$ as the j-th coefficient of polynomial s_i and u_i, respectively. To make the intermediate value t affected by only one coefficient value of s_0, we set all coefficients of $\mathbf{u}$, except $u_{0,0}$, to zero. Thus, u_0 is a constant, and u_i for $1 \le i \le k-1$ is zero. We also set v as zero to remove its effects (We can set the values of all coefficients v_i to the same value. In this case, the value of the chosen-ciphertext is slightly changed.). Accordingly, all coefficients of the input polynomial mp of poly_reduce are determined as $mp_j = -s_{0,j}u_{0,0}$ for $0 \le j \le n-1$.

For KYBER512, $\mathbf{s} = (s_0, s_1)$ and $\mathbf{u} = (u_0, u_1)$. Thus, we set $(u_0, u_1) = (x, 0)$ and $v = 0$, where $x \in \mathbb{Z}_q$. To cluster t values according to the sensitive variable $s_{0,j} \in \{-3, -2, -1, 0, 1, 2, 3\}$, we calculate t values according to all u_0 values, as shown in Figure 2. When $s_{0,j} = 0$ for $0 \le j \le n-1$, the intermediate value t is always zero.

We choose three u_0 values, as shown in Table I, and make sequences based on the value of t. The chosen ciphertexts selected in this study are examples and can be selected variously based on Table I. In this setting, there are 3154 valid chosen ciphertexts we can use (There are 970 values except duplicate values due to Compress.). Set to 0 when t is zero; otherwise, 1 to create sequences. As shown in Table I, the sequence according to the sensitive variable $s_{0,j}$ is different. Therefore, if the sequence is obtained using the side-channel leakage, then the sensitive variable $s_{0,j}$ is discovered. Accordingly, we can recover s_0, half of the secret key, by performing coefficient-wise analysis. Similarly, we can recover s_1 by using chosen ciphertexts $\mathbf{u} = (0, 208), \mathbf{u} = (0, 1109)$, and $\mathbf{u} = (0, 2217)$ (v is always zero). As a result, we can acquire the secret key $\mathbf{s}$ using six chosen ciphertexts.

For KYBER768 and KYBER1024, $s_{i,j} \in \{-2, -1, 0, 1, 2\}$ because the parameter η_1 is 2 at both levels. Therefore, similar chosen ciphertexts can be used as before. Since $k = 3$ and $k = 4$ for each level, 3×3 = 9 and 4×3 = 12 chosen ciphertexts are needed, respectively. However, if we additionally use the leakage that occurs at steps 8 and 10 of Listing 3, we can reduce the number of chosen ciphertexts. If $s_{i,j} = 0$, then the input coefficient of Barrett reduction is always zero; otherwise, it is nonzero. Thus, a leakage difference depending on the operand value at steps 8 and 10 of Listing 3 can be used to distinguish zero from the others. Accordingly, we can distinguish $s_{0,j}$ values using $\mathbf{u} = (208, 0, 0)$ and $\mathbf{u} = (1109, 0, 0)$ for KYBER768. As a result, it only needs $3 \times 2 = 6$ and $4 \times 2 = 8$ chosen ciphertexts for each level.

C. Attack Methodology

We target reference codes submitted to the NIST website by developers. All reference codes were implemented based on the C language; thus, we applied the Hamming weight power consumption model, commonly supposed in software implementations. Based on the previous analysis results, we can figure out the power consumption properties of 9-10 steps of Listing 3 as follows.

Property 1. *The power consumed in a software implementation is proportional to the Hamming weight of an intermediate value. Therefore, when the intermediate value* t *is* `0x0000`*, consuming power in proportion to* 0 *is occurred. Whereas, when the* t *value is equal to* `-3329 = 0xf2ff`*, consuming power in proportion to* 13 *is occurred. Here,* 13 *is the Hamming weight of the* t *value when* t *is a 16-bit integer.*

Algorithm 2 shows an attack algorithm based on the leakage that occurs at steps 9-10 of Listing 3. A significant difference in the performance of analysis exists, depending on the position of the attack. Therefore, specific points of interest (PoIs) must be found. Based on profiling, we can select the PoIs where significant variances are observed depending on secret coefficient values when using specifically chosen ciphertexts. We can identify the PoIs by calculating the sum of squared pairwise t-differences (SOST) [42] of the traces and then identifying the location of the information-leaking point. The SOST of two groups, $\mathbb{G}_1$ and $\mathbb{G}_2$, is calculated as follows.

$$SOST = \sum_{i,j=1}^{g} \left(\frac{E(\mathbb{G}_i) - E(\mathbb{G}_j)}{\sqrt{\frac{\sigma(\mathbb{G}_i)^2}{\#\mathbb{G}_i} + \frac{\sigma(\mathbb{G}_j)^2}{\#\mathbb{G}_j}}} \right)^2 \quad \text{for } i \ge j,$$

$E(\cdot)$, $\sigma(\cdot)$, #, and g denote the mean, standard deviation, number of elements, and number of groups, respectively. Here, g is 2.

For each $s_{i,j}$, we take the points where the t value is computed, stored, and loaded. We take these points $p_{c,j}$, which consume power proportional to the Hamming weight of the t value, as the PoIs and sort them into two groups using a clustering algorithm. Here, we can apply various clustering

TABLE I
THE INTERMEDIATE VALUE t

$s_{0,j}$		-3	-2	-1	0	1	2	3
$\mathbf{u} = (208, 0)$	$v = 0$	0	0	0	0	-3329	-3329	-3329
$\mathbf{u} = (1109, 0)$	$v = 0$	-3329	-3329	0	0	-3329	0	0
$\mathbf{u} = (2217, 0)$	$v = 0$	-3329	0	-3329	0	0	-3329	0
Sequence		011	010	001	000	110	101	100

algorithms, such as k-means, fuzzy k-means, and expectation-maximization (EM) [43]–[46].

By using one of these clustering algorithms, $p_{c,j}$ can be sorted into two groups: $\mathbb{G}_1$ and $\mathbb{G}_2$. Here, $\mathbb{G}_1$ and $\mathbb{G}_2$ represented each clustered group. Because power consumption depends on the Hamming weight of intermediate values, the mean values of $\mathbb{G}_1$ and $\mathbb{G}_2$ are different. Therefore, supposing that the larger the hamming weight, the less power consumed, we can identify the corresponding t value for each group according to the mean value of the two groups. This supposition depends on the structure of the measuring equipment; in this study, the supposition is established according to the structure of the ChipWhisperer-Lite main board used to obtain the power consumption of the target board [27].

Thus, for instance, when $E(\mathbb{G}_1)$ is larger than $E(\mathbb{G}_2)$, the value of t belonging to $\mathbb{G}_1$ has a value of 0 and that belonging to $\mathbb{G}_2$ has a value of -3329. $E(\mathbb{G}_1)$ and $E(\mathbb{G}_2)$ are the mean values of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. In Algorithm 2, $ss_{c,j}$ is the value for creating sequences. Therefore, it is set to 0 when t is zero; otherwise, it is set to 1. After repeating as many as the number of chosen ciphertexts, we can acquire a sequence $(ss_{0,j} \cdots ss_{cc-1,j})$ of each coefficient $s_{i,j}$. Hence, s_i, the part of the secret key, can be found. As a result, by repeating as many as rank, we can acquire the secret key $\mathbf{s}$.

Remark. The pqm4 library includes four schemes, namely ref, clean, opt, and m4 [39]. The schemes ref, clean, and opt are implemented in plain C; Listing 1, Listing 2, and Listing 3 are all identical in ref, clean, and opt. An implementation optimized for Cortex-M4 is the m4 scheme; it is typically implemented in assembly language as described in Appendix A.

IV. Experiment Results

In this section, we present experimental results that the secret key $\mathbf{s}$ could be recovered using six chosen ciphertexts for KYBER512. Side-channel vulnerability depends on how algorithms are implemented. Therefore, we utilized reference codes submitted to the NIST website by developers. All reference codes were implemented based on the C language; thus, we used the Hamming weight power consumption model, commonly supposed in software implementations. The experiments were conducted by focusing on ARM Cortex-M4 at NIST's request. We used gcc-arm-none-eabi compiler and options `-O3` and `-Os`, which optimize speed (High) and size, respectively.

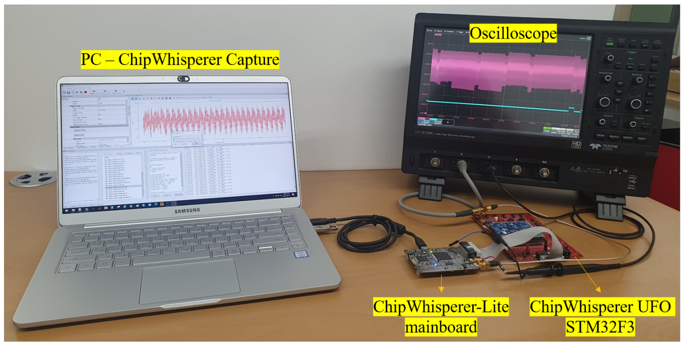

PC – ChipWhisperer Capture
Oscilloscope
ChipWhisperer-Lite mainboard
ChipWhisperer UFO STM32F3

Fig. 3. Experiment environment

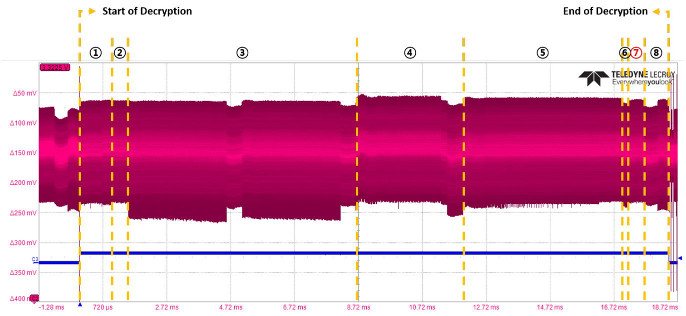


Fig. 4. Power consumption trace of Listing 1 (Optimization Level 3)

We acquired power consumption traces for the different secret key $\mathbf{s}$ when Listing 1 was running on the ChipWhisperer UFO STM32F3 target board [47]. The ChipWhisperer-Lite mainboard and the ChipWhisperer-Pro Kit can only collect up to 24,573 and 98,119 samples, respectively; therefore, we used a Teledyne Lecroy HDO6104A oscilloscope when acquiring whole traces of the decryption function, as shown in Figure 3 and Figure 4. Power consumption traces of Listing 1 were measured at a sampling rate of 2.5 GS/s. Each part of Figure 4 is as follows.

① unpack_ciphertext
② unpack_sk
③ polyvec_ntt
④ polyvec_pointwise_acc_montgomery
⑤ poly_invntt_tomont
⑥ poly_sub
⑦ poly_reduce
⑧ poly_tomsg

⑦ is our target function Listing 2, and we used traces of

Algorithm 2 Chosen-Ciphertext Clustering Attack on Barrett Reduction in CRYSTALS-KYBER

```
Require: Trace sets $T = (T_0, \cdots, T_{k-1})$
Require: Secret sequence $Seq = (seq_0, \cdots, seq_{2\eta_1})$
Require: Secret coefficient value $cv = (cv_0, \cdots, cv_{2\eta_1})$
Ensure:  Secret key $\mathbf{s} = (s_0, \cdots, s_{k-1})$
 1: /*as many as rank*/
 2: for $i = 0$ up to $k-1$ do
 3:   /*as many as the number of chosen ciphertexts*/
 4:   for $c = 0$ up to $cc-1$ do
 5:     /*as many as the size of the dimension*/
 6:     for $j = 0$ up to $n-1$ do
 7:       /*position identified in profiling phase*/
 8:       Select the PoIs $p_{c,j}$ associated with $s_{i,j}$
 9:     end for
10:     Classify $p_{c,j}$ into two groups, $\mathbb{G}_1$ and $\mathbb{G}_2$, using a clustering algorithm
11:     Compute the mean values $E(\mathbb{G}_1)$ and $E(\mathbb{G}_2)$, respectively, of $\mathbb{G}_1$ and $\mathbb{G}_2$
12:     for $j = 0$ up to $n-1$ do
13:       /*assume that $E(\mathbb{G}_1) > E(\mathbb{G}_2)$*/
14:       if $p_{c,j} \in \mathbb{G}_1$ then
15:         /*$ss_{c,j} = 0$ when it follows the Property 1*/
16:         $ss_{c,j} \leftarrow 0$
17:       else
18:         /*$ss_{c,j} = 1$ when it follows the Property 1*/
19:         $ss_{c,j} \leftarrow 1$
20:       end if
21:     end for
22:   end for
23:   for $j = 0$ up to $n-1$ do
24:     $e = 0$
25:     while $(ss_{0,j} \cdots ss_{cc-1,j}) \neq seq_e$ do
26:       $e++$
27:     end while
28:     $s_{i,j} = cv_e$
29:   end for
30: end for
31: Return $(s_0, \cdots, s_{k-1})$
```

this position as the input traces of Algorithm 2. A low-noise amplifier is equipped on the ChipWhisperer-Lite mainboard [48]; thus, input power consumption traces of Algorithm 2 were measured by the ChipWhisperer-Lite mainboard at a sampling rate of 29.54 MS/s.

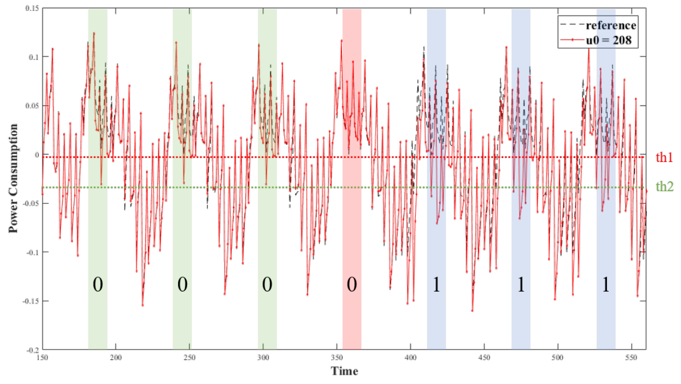


Fig. 5. Power consumption trace when we set ciphertext as $\mathbf{u} = (208, 0)$ and $v = 0$ (Optimization Level 3)

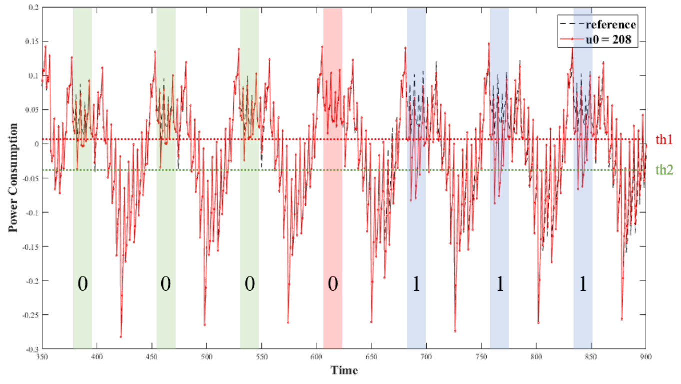


Fig. 6. Power consumption trace when we set ciphertext as $\mathbf{u} = (208, 0)$ and $v = 0$ (Optimization Level s)

Figure 5 and Figure 6 show that the parts of power consumption traces when $s_{0,j}$ is sequentially set -3 to -2, -1, 0, 1, 2, and 3. Given that the chosen-ciphertext consists of $\mathbf{u} = (208, 0)$ and $v = 0$, $\mathfrak{t}$ values are 0, 0, 0, 0, -3329, -3329, and -3329 for each coefficient, as described in Table I. If we set to 0 when $\mathfrak{t}$ is zero and otherwise set 1, we can acquire a sequence (0, 0, 0, 0, 1, 1, 1).

We drew lines *th1*, *th2* in Figure 5 and Figure 6, and we marked them as 0 if the value of the y-axis in the highlighted area is bigger than *th2*; otherwise, we marked them as 1. Sequences denoted in Figure 5 and Figure 6 are the same as the sequence (0, 0, 0, 0, 1, 1, 1) obtained in accordance with the $\mathfrak{t}$ value. Therefore, we can see that the information of the $\mathfrak{t}$ value is leaking, and the differences in power consumption are big enough to be exploited.

To identify the PoIs, we computed the SOST values of measured power consumption traces, as shown in Figure 7 and Figure 10. Figure 8 (b) and Figure 11 (b) show the distributions at a 195 and 387 points, respectively. The differences between $E(\mathbb{G}_1)$ and $E(\mathbb{G}_2)$ are large enough to be visually distinct, and no error rate is observed.

Figure 9 and Figure 12 show that power consumption traces measured using three chosen ciphertexts. The magnification

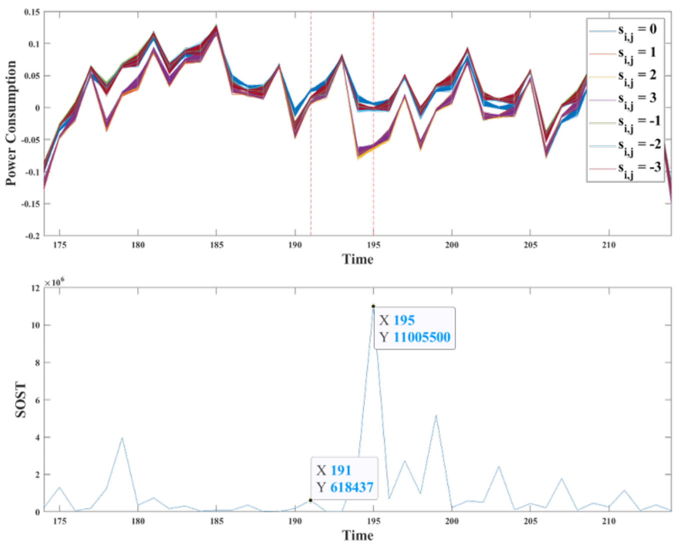


Fig. 7. The SOST values when we set ciphertext as $\mathbf{u} = (208, 0)$ and $v = 0$ (Optimization Level 3)

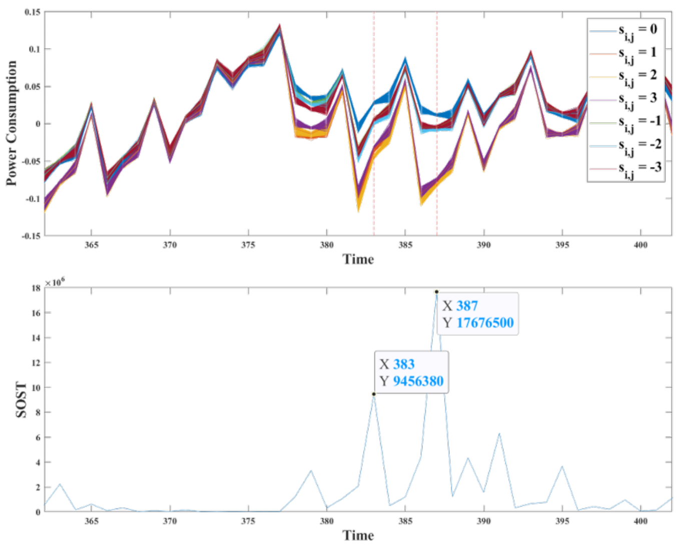


Fig. 10. The SOST values when we set ciphertext as $\mathbf{u} = (208, 0)$ and $v = 0$ (Optimization Level s)

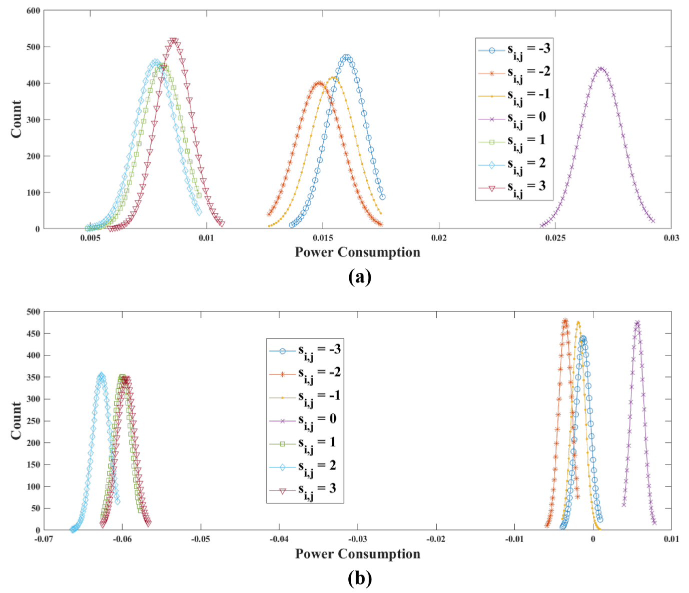


Fig. 8. Distributions of the PoIs when we set ciphertext as $\mathbf{u} = (208, 0)$ and $v = 0$; (a) 191 point and (b) 195 point (Optimization Level 3)

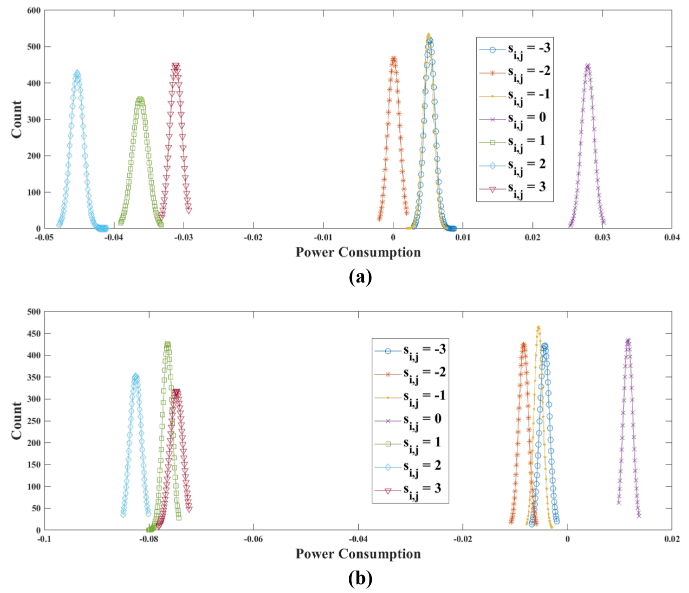


Fig. 11. Distributions of the PoIs when we set ciphertext as $\mathbf{u} = (208, 0)$ and $v = 0$; (a) 383 point and (b) 387 point (Optimization Level s)

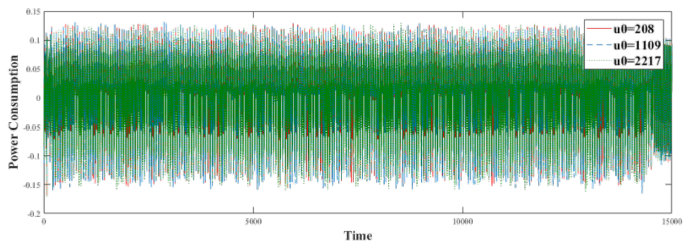


Fig. 9. Measurement traces of Listing 3 using three chosen ciphertexts (Optimization Level 3)

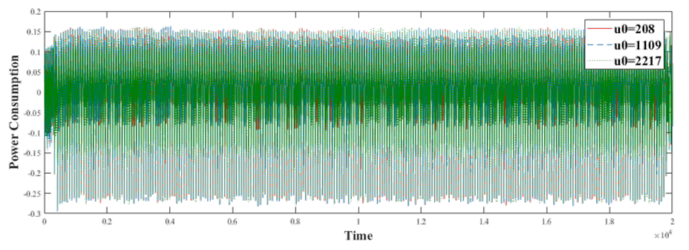


Fig. 12. Measurement traces of Listing 3 using three chosen ciphertexts (Optimization Level s)

of the positions for each coefficient is shown in Figure 13 and Figure 14. We marked sequences according to the value of the y-axis; thus, they are the same as the sequence in Table I. We split power consumption traces in Figure 9 and

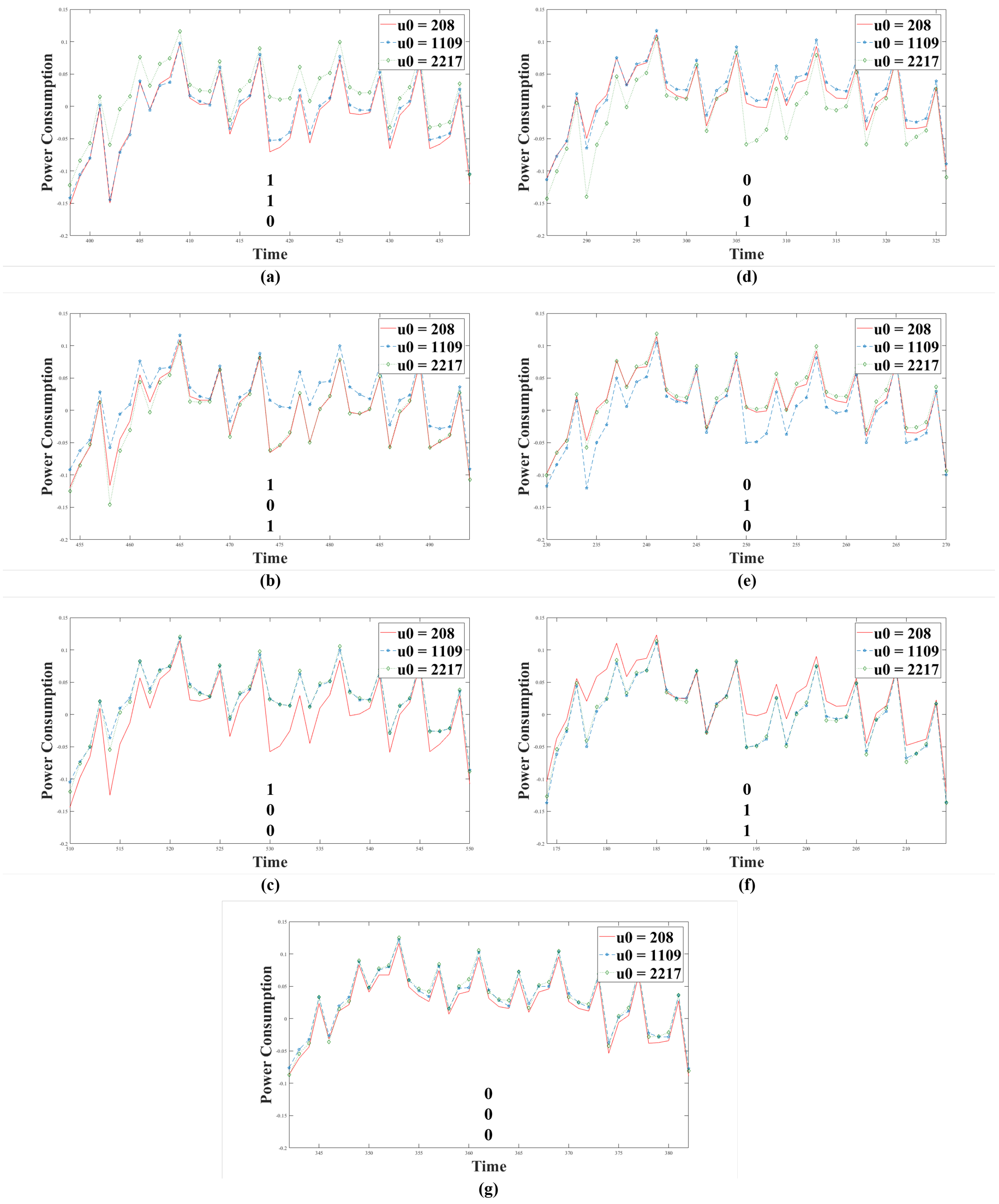
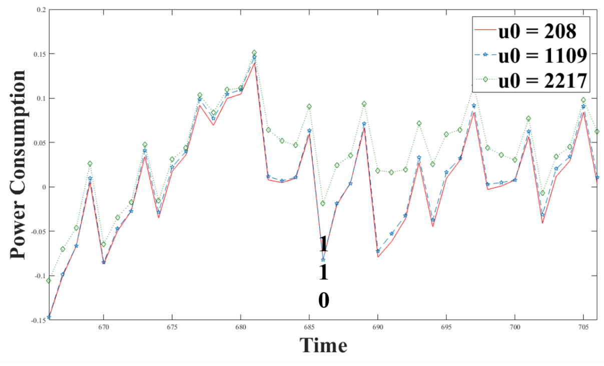
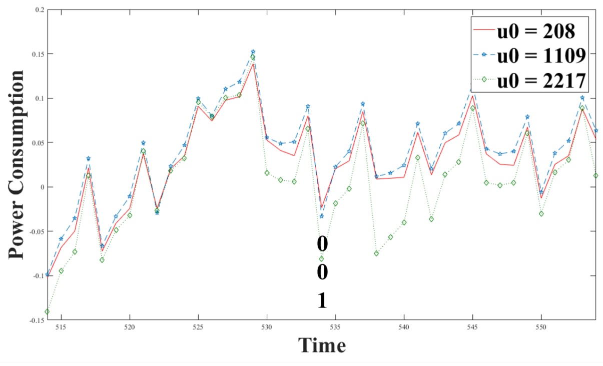


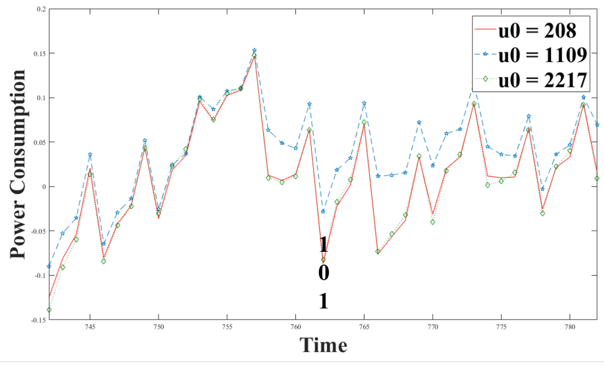
Fig. 13. Measurement traces using three chosen ciphertexts when (a) $s_{0,j} = 1$, (b) $s_{0,j} = 2$, (c) $s_{0,j} = 3$, (d) $s_{0,j} = -1$, (e) $s_{0,j} = -2$, (f) $s_{0,j} = -3$, and (g) $s_{0,j} = 0$ (Optimization Level 3).



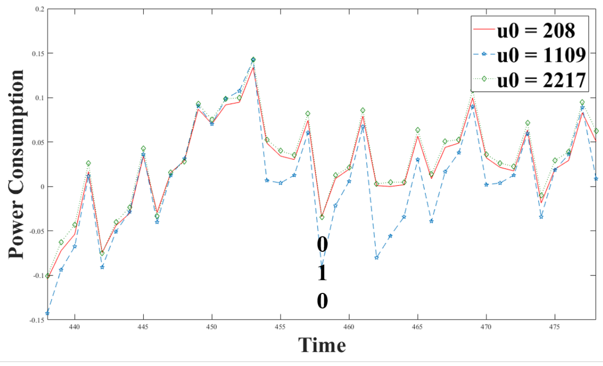
(a)



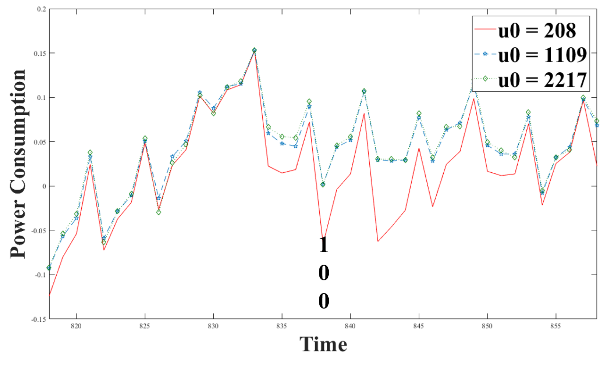
(d)



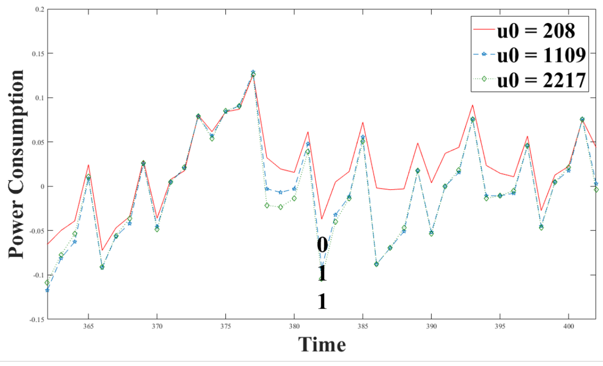
(b)



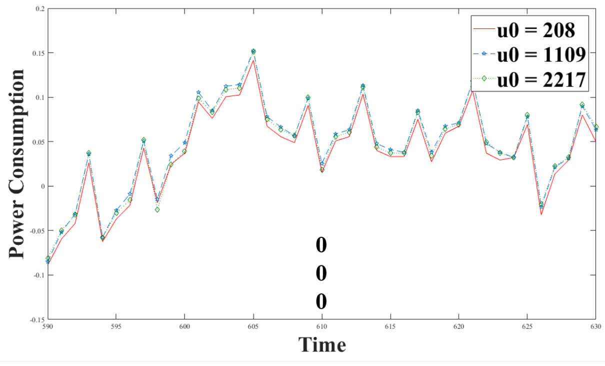
(e)



(c)



(f)



(g)

Fig. 14. Measurement traces using three chosen ciphertexts when (a) $s_{0,j} = 1$, (b) $s_{0,j} = 2$, (c) $s_{0,j} = 3$, (d) $s_{0,j} = -1$, (e) $s_{0,j} = -2$, (f) $s_{0,j} = -3$, and (g) $s_{0,j} = 0$ (Optimization Level s)

Figure 12 into sub-traces for each coefficient and applied r max normalization. As a result, the secret key can be extra with a 100% success rate using Algorithm 2 based on the algorithm.

As shown in Figure 5 and Figure 6, whether t = 0 or not be distinguished by identifying whether power consump trace is higher than *th1* or not. Moreover, Figure 8 Figure 11 show that clustering into three groups is possi thus, distinguishing whether t = 0 or not is also possible. reduces the number of chosen ciphertexts from three to to recover s_i of KYBER768 and KYBER1024. Accordin the number of chosen ciphertexts required to recover KYBER768 and KYBER1024 are six and eight, respectiv We split three power consumption traces into sub-traces each coefficient and applied min-max normalization. We then slightly modified steps 10-21 of Algorithm 2 to cluster into three groups. As a result, the secret key can be extracted with a 100% success rate using the EM algorithm.

Experimental results on the m4 scheme. We also show that the m4 implementation with Cortex-M4 specific optimizations (typically in assembly) is vulnerable to the proposed attack. Since Barrett reduction is implemented in assembly language as shown in Listing 7 and Listing 8, we only report the experiment results for compiler option `-O3`.

In contrast to the ref scheme, the m4 scheme performs Barrett reduction on two coefficients simultaneously. The intermediate value tmp and tmp2 in Listing 8 are for two coefficients $s_{0,j}$ and $s_{0,j+1}$, respectively. Figure 15 shows power consumption traces when Listing 8 is in operation. Power consumption is affected by a sequence of t values for two coefficients. For example, if $s_{0,j} = -1$ and $s_{0,j+1} = 3$ when $\mathbf{u} = (208, 0)$ and $v = 0$, then a sequence of t values is 01. Accordingly, it can be classified into four groups according to the power consumption pattern of four clock cycles in which steps 7-10 of Listing 8 are performed. In particular, Figure 16 (b) shows that clustering into four groups is possible with no error rate. In Figure 16 (a), distributions of 01 and 10 are overlapped; thus, they would be classified into same groups. Accordingly, if we use the point 31, clustering into three groups is possible.

Figure 17 and Figure 18 show that power consumption traces measured using three chosen ciphertexts. We marked sequences according to the patterns of the four clock cycles in which steps 7-10 of Listing 8. When rearranged into a sequence by each coefficient, they are the same as the sequence in Table I. In the m4 scheme, distinguishing when $s_{0,j} = 0$ or $s_{0,j+1} = 0$ from other cases is difficult because two coefficients are computed simultaneously. Therefore, for KYBER768 and KYBER1024, $3 \times 3 = 9$ and $4 \times 3 = 12$ chosen ciphertexts are needed, respectively. We split three power consumption traces into sub-traces for two coefficients and applied z-score normalization. As a result, the secret key can be extracted with with a 100% success rate using the EM algorithm.

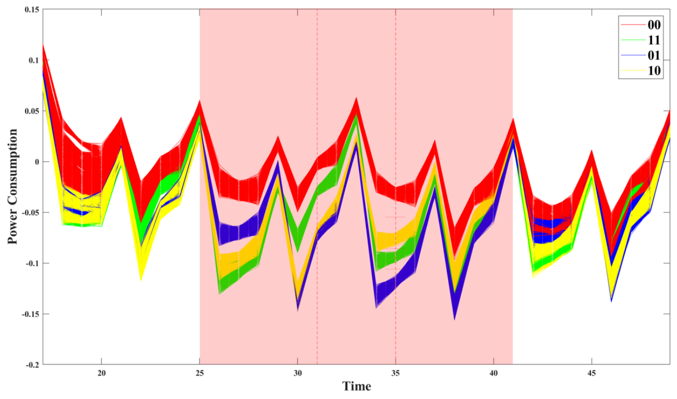


Fig. 15. Power consumption traces when we set ciphertext as $\mathbf{u} = (208, 0)$ and $v = 0$ (m4 scheme, Optimization Level 3)

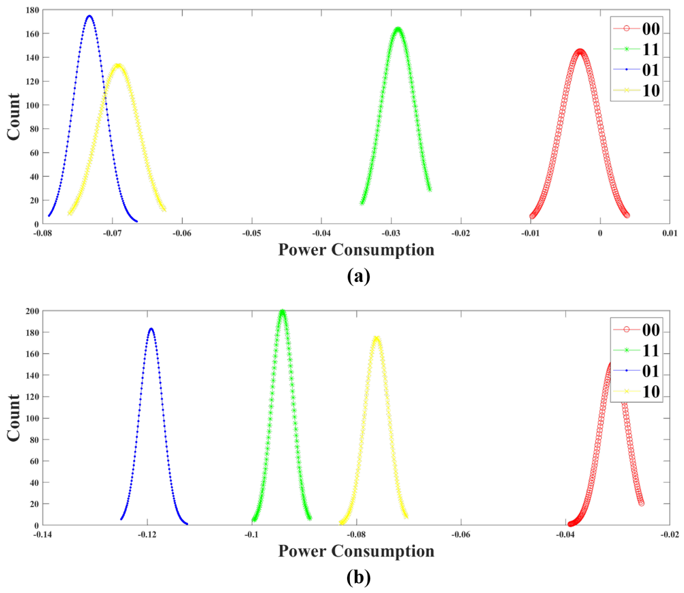


Fig. 16. Distributions of the PoIs when we set ciphertext as $\mathbf{u} = (208, 0)$ and $v = 0$; (a) 31 point and (b) 35 point (m4 scheme, Optimization Level 3)

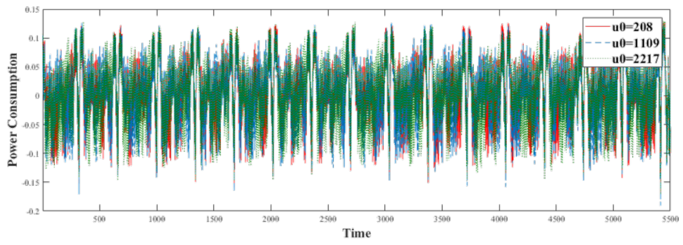
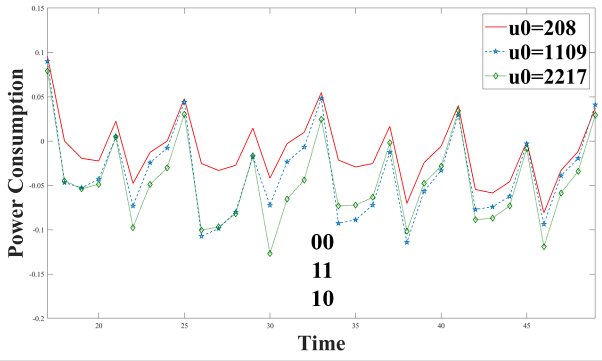
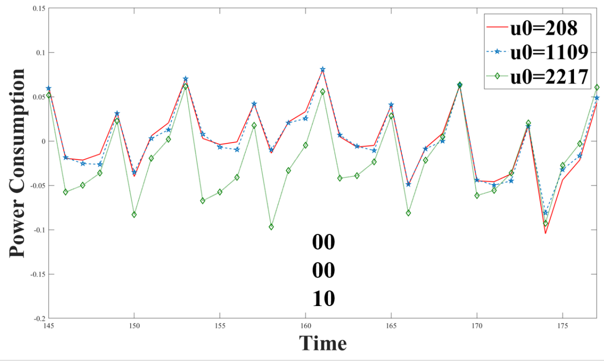


Fig. 17. Measurement traces of Listing 7 using three chosen ciphertexts (m4 scheme, Optimization Level 3)

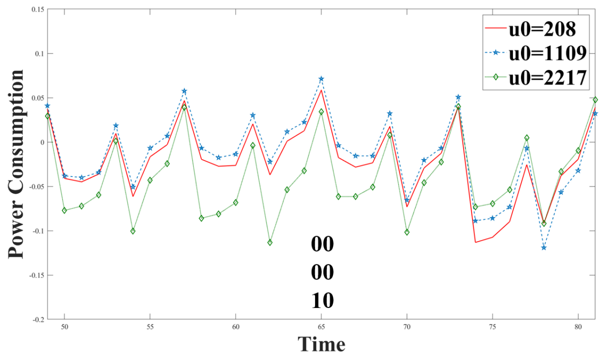
Remark. Because [34], [35] did not experiment on the updated specification, accurate comparisons are not possible. However, since the noise parameter was increased [40], it is obvious that more chosen ciphertexts are needed than the number stated in [34], [35]. Accordingly, our proposed method much more efficient for the m4 scheme.



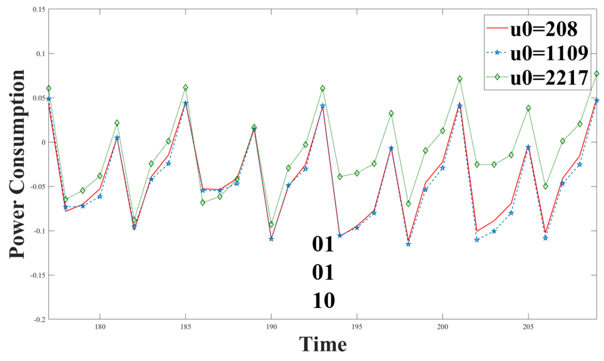
(a)



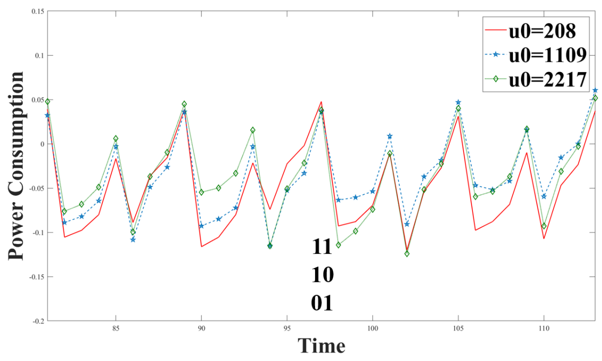
(e)



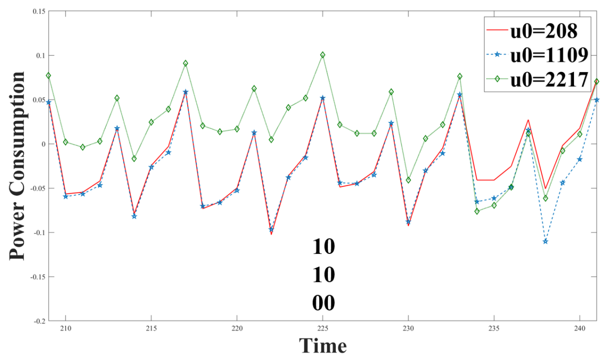
(b)



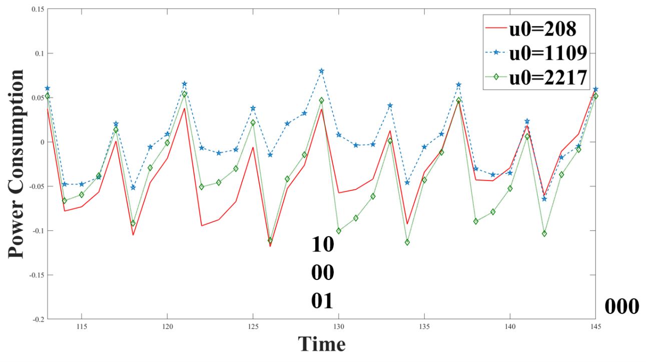
(f)



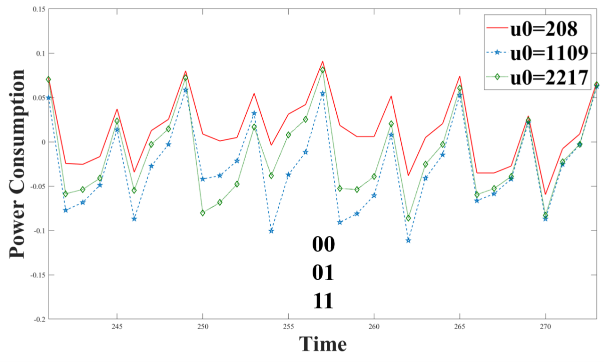
(c)



(g)



(d)



(h)

Fig. 18. Measurement traces using three chosen ciphertexts when (a) $s_{0,j} = -3$, $s_{0,j+1} = -2$, (b) $s_{0,j} = -1$, $s_{0,j+1} = 0$, (c) $s_{0,j} = 1$, $s_{0,j+1} = 2$, (d) $s_{0,j} = 3$, $s_{0,j+1} = -1$, (e) $s_{0,j} = -1$, $s_{0,j+1} = 0$, (f) $s_{0,j} = -1$, $s_{0,j+1} = 1$, (g) $s_{0,j} = 1$, $s_{0,j+1} = 0$, and (h) $s_{0,j} = -1$, $s_{0,j+1} = -3$ (m4 scheme, Optimization Level 3)

V. Countermeasures

Since the proposed attack constructs an intermediate value t, which is affected by only one coefficient of s_i, and exploits it, masking [22], [49] can be secure against the proposed attack. However, substantial time and memory resources are needed because masking would not be appropriate for use in resource-constrained IoT devices due to its high-performance overhead. Thus, using shuffling and hardware noise-addition to increase attack complexity might be a good idea. Similar to [29], shuffling can be applied by generating a shuffling index array, as shown in Listing 4. To attack the shuffling, $k{\times}3{\times}256$ chosen ciphertexts are required because the target coefficient v_j must be changed. That is, it is possible to recover one coefficient at a time. Thus, if the key reuse period is properly adjusted, it can be fully responded to. Using another reduction method, such as Montgomery reduction, can also be a countermeasure.

```c
// Applies Barrett reduction
// to all coefficients of a polynomial
void poly_reduce(poly *r, int *shuffled_index)
{
  unsigned int i;
  for(i=0;i<KYBER_N;i++)
    r->coeffs[shuffled_index[i]]
     = barrett_reduce(r->coeffs[shuffled_index[i]]);
}
```

Listing 4. Reduction of CRYSTALS-KYBER (in C code)

VI. Conclusion

In this study, we proposed a chosen-ciphertext clustering attack on CRYSTALS-KYBER using sensitive variable-dependent leakage of Barrett reduction. We took advantage of the fact that the intermediate value of an operation is determined to be the value of one of the three values, and the difference in the Hamming weight of the intermediate value is larger than 4. To magnify the difference in the sensitive variable-dependent leakage, we used chosen ciphertexts. As a result, we could acquire the full secret key using only six chosen ciphertexts for KYBER512. Depending on an implementation scheme, recovering the secret key of KYBER768 requires six or nine chosen ciphertexts. For KYBER1024, eight or twelve chosen ciphertexts are required depending on an implementation scheme.

Vulnerability occurred due to implementation methods that prevent timing leakage. Barrett reduction, used in CRYSTALS-KYBER, is secure against timing attack; however, it does not guarantee security against power analysis. Especially, the method applied to secure implementation against timing attacks led to a greater amount of side-channel leakage. Therefore, research should be conducted on how to avoid such leakage.

References

[1] K. L. Lueth, "State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time," https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/, 2020.

[2] M. Rykov, "5 IoT Security best practices to consider after the Covid-19 lockdown," https://iot-analytics.com/5-iot-security-best-practices-after-the-covid-19-lockdown/, 2020.

[3] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *35th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1994, pp. 124–134.

[4] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, "Report on Post-Quantum Cryptography," https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf, 2016.

[5] M. Mariantoni, "Building a superconducting quantum computer," 2014. [Online]. Available: https://www.youtube.com/watch?v=wWHAs--HA1c

[6] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?" *IEEE Secur. Priv.*, vol. 16, no. 5, pp. 38–41, 2018.

[7] NIST, "Post-Quantum Cryptography, Workshops and Timeline, NIST Computer Security Resource Center," https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline, 2017.

[8] ——, "Post-Quantum Cryptography, Round 3 Submissions, NIST Computer Security Resource Center," https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement, 2020.

[9] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS - kyber: A cca-secure module-lattice-based KEM," in *IEEE European Symposium on Security and Privacy*. IEEE, 2018, pp. 353–367.

[10] J. D'Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren, "Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure KEM," in *International Conference on Cryptology in Africa*. Springer, 2018, pp. 282–305.

[11] J. W. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! practical, quantum-secure key exchange from LWE," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1006–1018.

[12] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *International Algorithmic Number Theory Symposium*. Springer, 1998, pp. 267–288.

[13] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. van Vredendaal, "NTRU prime: Reducing attack surface at low cost," in *International Conference on Selected Areas in Cryptography*. Springer, 2017, pp. 235–260.

[14] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Annual International Cryptology Conference*. Springer, 1996, pp. 104–113.

[15] NIST, "Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process," https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf, 2016.

[16] J. H. Silverman and W. Whyte, "Timing attacks on ntruencrypt via variation in the number of hash calls," in *The Cryptographers' Track at the RSA Conference*. Springer, 2007, pp. 208–224.

[17] A. Park and D. Han, "Chosen ciphertext simple power analysis on software 8-bit implementation of ring-lwe encryption," in *2016 IEEE Asian Hardware-Oriented Security and Trust, AsianHOST 2016, Yilan, Taiwan, December 19-20, 2016*. IEEE Computer Society, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/AsianHOST.2016.7835555

[18] A. Atici, L. Batina, B. Gierlichs, and I. Verbauwhede, "Power analysis on ntru implementations for rfids: First results," *RFIDSec 2008*, pp. 128–139, 2008.

[19] M. Lee, J. E. Song, D. Choi, and D. Han, "Countermeasures against power analysis attacks for the NTRU public key cryptosystem," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 93-A, no. 1, pp. 153–163, 2010.

[20] A. Aysu, Y. Tobah, M. Tiwari, A. Gerstlauer, and M. Orshansky, "Horizontal side-channel vulnerabilities of post-quantum key exchange protocols," in *IEEE International Symposium on Hardware Oriented Security and Trust*. IEEE Computer Society, 2018, pp. 81–88.

[21] J. W. Bos, S. Friedberger, M. Martinoli, E. Oswald, and M. Stam, "Assessing the feasibility of single trace power analysis of frodo," in *International Conference on Selected Areas in Cryptography*. Springer, 2018, pp. 216–234.

[22] T. Oder, T. Schneider, T. Pöppelmann, and T. Güneysu, "Practical cca2-secure and masked ring-lwe implementation," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2018, no. 1, pp. 142–174, 2018.

[23] O. Reparaz, R. de Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede, "Additively homomorphic ring-lwe masking," in *International Conference on Post-Quantum Cryptography*. Springer, 2016, pp. 233–244.

[24] O. Reparaz, S. S. Roy, R. de Clercq, F. Vercauteren, and I. Verbauwhede, "Masking ring-lwe," *J. Cryptographic Engineering*, vol. 6, no. 2, pp. 139–153, 2016.

[25] R. Primas, P. Pessl, and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 513–533.

[26] P. Pessl and R. Primas, "More practical single-trace attacks on the number theoretic transform," in *International Conference on Cryptology and Information Security in Latin America*. Springer, 2019, pp. 130–149.

[27] W. Huang, J. Chen, and B. Yang, "Power analysis on NTRU prime," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2020, no. 1, pp. 123–151, 2020.

[28] D. Amiet, A. Curiger, L. Leuenberger, and P. Zbinden, "Defeating newhope with a single trace," in *International Conference on Post-Quantum Cryptography*. Springer, 2020, pp. 189–205.

[29] B. Sim, J. Kwon, J. Lee, I. Kim, T. Lee, J. Han, H. J. Yoon, J. Cho, and D. Han, "Single-trace attacks on message encoding in lattice-based kems," *IEEE Access*, vol. 8, pp. 183 175–183 191, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3029521

[30] A. Bauer, H. Gilbert, G. Renault, and M. Rossi, "Assessment of the key-reuse resilience of newhope," in *The Cryptographers' Track at the RSA Conference*. Springer, 2019, pp. 272–292.

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

[32] P. Ravi, S. S. Roy, A. Chattopadhyay, and S. Bhasin, "Generic side-channel attacks on cca-secure lattice-based PKE and kems," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2020, no. 3, pp. 307–335, 2020. [Online]. Available: https://doi.org/10.13154/tches.v2020.i3.307-335

[33] P. Ravi, S. Bhasin, S. S. Roy, and A. Chattopadhyay, "Drop by drop you break the rock - exploiting generic vulnerabilities in lattice-based pke/kems using em-based physical attacks," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 549, 2020. [Online]. Available: https://eprint.iacr.org/2020/549

[34] Z. Xu, O. Pemberton, S. S. Roy, and D. Oswald, "Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 912, 2020. [Online]. Available: https://eprint.iacr.org/2020/912

[35] P. Ravi, S. Bhasin, S. S. Roy, and A. Chattopadhyay, "On exploiting message leakage in (few) NIST PQC candidates for practical message recovery and key recovery attacks," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1559, 2020. [Online]. Available: https://eprint.iacr.org/2020/1559

[36] K. Ngo, E. Dubrova, Q. Guo, and T. Johansson, "A side-channel attack on a masked IND-CCA secure saber KEM," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 79, 2021. [Online]. Available: https://eprint.iacr.org/2021/079

[37] Q. Guo, T. Johansson, and A. Nilsson, "A key-recovery timing attack on post-quantum primitives using the fujisaki-okamoto transformation and its application on frodokem," in *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, D. Micciancio and T. Ristenpart, Eds., vol. 12171. Springer, 2020, pp. 359–386. [Online]. Available: https://doi.org/10.1007/978-3-030-56880-1_13

[38] S. Bhasin, J. D'Anvers, D. Heinz, T. Pöppelmann, and M. V. Beirendonck, "Attacking and defending masked polynomial comparison for lattice-based cryptography," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 104, 2021. [Online]. Available: https://eprint.iacr.org/2021/104

[39] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and benchmarking NIST PQC on ARM cortex-m4," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 844, 2019. [Online]. Available: https://eprint.iacr.org/2019/844

[40] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS - KYBER: Algorithm specifications and supporting documentation (version 3.01)," https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf, 2021.

[41] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 1–23.

[42] B. Gierlichs, K. Lemke-Rust, and C. Paar, "Templates vs. stochastic methods," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2006, pp. 15–29.

[43] Y. Anzai, *Pattern Recognition & Machine Learning*. Elsevier, 1992. [Online]. Available: https://doi.org/10.1016/c2009-0-22409-3

[44] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996, pp. 226–231.

[45] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 32–40, 1975.

[46] L. Rokach and O. Maimon, "Clustering methods," in *The Data Mining and Knowledge Discovery Handbook*. Springer, 2005, pp. 321–352.

[47] N. T. Inc, "ChipWhisperer UFO," https://wiki.newae.com/CW308T-STM32F.

[48] ——, "ChipWhisperer-Lite," https://wiki.newae.com/CW1173_ChipWhisperer-Lite.

[49] J. W. Bos, M. Gourjon, J. Renes, T. Schneider, and C. van Vredendaal, "Masking kyber: First- and higher-order implementations," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 483, 2021. [Online]. Available: https://eprint.iacr.org/2021/483

Appendix A
pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4 [41]

Listing 5, Listing 6, Listing 7, and Listing 8 are codes of m4 schemes. Barrett reduction is implemented in assembly language and performs on two coefficients simultaneously. This is due to Cortex-M4 implements the ARMv7E-M architecture, offers single instruction multiple data (SIMD) instructions.

```
// Decryption function of the CPA-secure
void __attribute__ ((noinline)) indcpa_dec
(unsigned char *m,
 const unsigned char *c,
 const unsigned char *sk)
{
  poly mp, bp;
  poly *v = &bp;

  poly_unpackdecompress(&mp, c, 0);
  poly_ntt(&mp);
  poly_frombytes_mul(&mp, sk);
  for(int i = 1; i < KYBER_K; i++) {
    poly_unpackdecompress(&bp, c, i);
    poly_ntt(&bp);
    poly_frombytes_mul(&bp, sk + i*KYBER_POLYBYTES);
    poly_add(&mp, &mp, &bp);
  }

  poly_invntt(&mp);
  poly_decompress(v, c+KYBER_POLYVECCOMPRESSEDBYTES);
  poly_sub(&mp, v, &mp);
  poly_reduce(&mp);

  poly_tomsg(m, &mp);
}
```

Listing 5. Decryption of CRYSTALS-KYBER (m4 scheme)

```
// Applies Barrett reduction
// to all coefficients of a polynomial
void poly_reduce(poly *r)
{
  asm_barrett_reduce(r->coeffs);
}
```

Listing 6. Reduction of CRYSTALS-KYBER (m4 scheme)

```
// reduce.S
.syntax unified
.cpu cortex-m4
.thumb

.global asm_barrett_reduce
.type asm_barrett_reduce,%function
.align 2
asm_barrett_reduce:
  push    {r4-r11, r14}

  poly         .req r0
  poly0        .req r1
  poly1        .req r2
  poly2        .req r3
  poly3        .req r4
  poly4        .req r5
  poly5        .req r6
  poly6        .req r7
  poly7        .req r8
  loop         .req r9
  barrettconst .req r10
  q            .req r11
  tmp          .req r12
  tmp2         .req r14

  movw barrettconst, #20159
  movw q, #3329

  movw loop, #16
  1:
    ldm poly, {poly0-poly7}

    doublebarrett poly0, tmp, tmp2, q, barrettconst
    doublebarrett poly1, tmp, tmp2, q, barrettconst
    doublebarrett poly2, tmp, tmp2, q, barrettconst
    doublebarrett poly3, tmp, tmp2, q, barrettconst
    doublebarrett poly4, tmp, tmp2, q, barrettconst
    doublebarrett poly5, tmp, tmp2, q, barrettconst
    doublebarrett poly6, tmp, tmp2, q, barrettconst
    doublebarrett poly7, tmp, tmp2, q, barrettconst

    stm poly!, {poly0-poly7}

    subs.w loop, #1
  bne.w 1b

  pop     {r4-r11, pc}
```

Listing 7. Barrett reduction of CRYSTALS-KYBER (m4 scheme)

```
// given a 16-bit integer a, computes 16-bit integer
// congruent to a mod q in {0,...,q}
// macros.i
.macro doublebarrett a, tmp, tmp2, q, barrettconst
  smulbb \tmp, \a, \barrettconst
  smultb \tmp2, \a, \barrettconst
  asr \tmp, \tmp, #26
  asr \tmp2, \tmp2, #26
  smulbb \tmp, \tmp, \q
  smulbb \tmp2, \tmp2, \q
  pkhbt \tmp, \tmp, \tmp2, lsl#16
  usub16 \a, \a, \tmp
.endm
```

Listing 8. Barrett reduction of CRYSTALS-KYBER (m4 scheme)