IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 19, 2021, accepted December 11, 2021, date of publication December 14, 2021,
date of current version December 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3135600

Single-Trace Attack on NIST Round 3 Candidate Dilithium Using Machine Learning-Based Profiling

JAESEUNG HAN[1], TAEHO LEE[1], JIHOON KWON[2], JOOHEE LEE[2], IL-JU KIM[1], JIHOON CHO[2], DONG-GUK HAN[1], AND BO-YEON SIM[3]

[1]Department of Financial Information Security, Kookmin University, Seoul 02707, Republic of Korea
[2]Security Research Center, Samsung SDS Inc., Seoul 05510, Republic of Korea
[3]Department of Intelligent Convergence Research Laboratory, Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, Republic of Korea

Corresponding author: Bo-Yeon Sim (sboyeon37@etri.re.kr)

This work was supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government [Ministry of Science and ICT (MSIT)] (No.2021-0-00903, Development of Physical Channel Vulnerability-based Attacks and its Countermeasures for Reliable On-Device Deep Learning Accelerator Design). This work was supported as part of Military Crypto Research Center (UD210027XD) funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD).

ABSTRACT In this paper, we propose single-trace side-channel attacks against CRYSTALS-DILITHIUM. CRYSTALS-DILITHIUM is a lattice-based digital signature algorithm, one of the third round finalists of the national institute of standards and technology (NIST) standardization project. We attack the number-theoretic transform (NTT) in the signing procedure and key generation of CRYSTALS-DILITHIUM to obtain a secret key. When targeting the signing procedure, we can recover both secret key vectors s_1 and s_2. This enables forgery of signatures. However, only the secret key vector s_1 can be recovered when targeting the key generation. Thus, we additionally attack four operations, sampling, addition, rounding, and packing, to find s_2. We applied a machine learning-based profiling attack method to find the secret key vectors s_1 and s_2 with a single trace.

INDEX TERMS Digital signature, lattice-based cryptography, number-theoretic transform, side-channel attack, machine learning-based profiling attack.

I. INTRODUCTION

Thestandards for digital signature such as RSA and ECDSA form a backbone of today's internet protocols. Despite their usefulness, it is well known that they can be broken in the presence of quantum computer running Shor's algorithm [1] in polynomial time. In this regard, in 2016, the National Institute of Standards and Technology (NIST) initiated a process to select new standards for Post-Quantum Cryptography (PQC) which aims to develop public-key encryption, key exchange,[1] and digital signature schemes secure against adversaries equipped with quantum computers [2]. Since then, there has been an increased attention on PQC constructions and attacks for them. From the initial 82 submissions by November 2017, NIST narrowed the selection to 4 finalists (*resp.,* 5 alternates) for key encapsulation mechanisms (KEM) and 3 finalists (*resp.,* 3 alternates) for signatures for the third round in July 2020. Among them, the constructions based on the lattice hard problems are presumed to be one of the most promising candidates to replace the current standards for public-key cryptosystems based on the integer factorization and discrete logarithm problems; two of the three finalists for signatures, FALCON and CRYSTALS-DILITHIUM, of the NIST's third round candidates are lattice-based schemes.

The side-channel attack (SCA) [3] analyzes side-channel information while implementing the cryptographic algorithms such as timing information, power consumption, electromagnetic radiation, etc., to extract the secret information. To introduce new post-quantum standards, careful evaluations of SCA for the candidates have emerged as a very important issue; NIST explicitly mentioned that they wanted

The associate editor coordinating the review of this manuscript and approving it for publication was Tyson Brooks.

[1]Later, the public-key encryption and key exchange are merged into the key encapsulation mechanisms.

to "collect more information about the costs of implementing in a way that provides resistance to side-channel attacks" [4]. The lattice-based schemes are also at risk for various side-channel attacks such as timing attacks [5], differential power analysis [6], cache attacks [7], template attacks [8], single-trace attacks [9], and fault attacks [10].

A. RELATED WORKS

Primas *et al.* [9] presented the first single-trace attack on lattice-based encryption by using leakage from the Number-Theoretic Transform (NTT). Their attack utilized the fact that in modular operations performed during the inverse NTT, division instruction does not run in constant-time. They showed that taking only a single power trace is sufficient to recover the full secret key, and it can be applied to masked implementations. Ravi *et al.* [11] presented a side-channel assisted existential forgery attack on the CRYSTALS-DILITHIUM, which is a combination of side-channel and classical attacks. They utilized power analysis on the polynomial multiplication to retrieve partial secret key, and showed that the partial secret key could be used to forge signature. Pessl and Primas [8] presented more practical single-trace attack on the NTT. Their approach changes the attack target from decryption to encryption and improve the attack performance, and they reported attack result against constant-time implementation of CRYSTALS-KYBER [12]. However, in this case, the attacker can recover the shared symmetric key, not the secret key. Fournaris *et al.* [13] presented a correlation power attacks on the polynomial multiplication operation of CRYSTALS-DILITHIUM signature generation. Their profiling analysis of CRYSTALS-DILITHIUM was performed on ARM Cortex-M4 embedded system. Hamburg *et al.* [14] presented an improved side-channel attack that used sparse polynomials at the input of the inverse NTT operations. The attack requires only k traces on the inverse NTT of CRYSTALS-KYBER, where $k \in \{2, 3, 4\}$ is the module dimension, is also applicable to masked CRYSTALS-KYBER implementations.

Three recent works targeting the NTT and recovering sensitive input (*e.g.*, secret key or message) using a single trace have been reported [8], [9], [14]. However, there are restrictions to apply these attacks, such as collecting decryption traces using sparse chosen ciphertexts as an input or applying belief propagation techniques are needed. These tasks motivated us to investigate new profiling methods that did not require the use of chosen ciphertexts and belief propagation techniques to recover secret keys.

B. MAIN CONTRIBUTIONS

In this paper, we discover the feasibility of SCA on the lattice-based signature CRYSTALS-DILITHIUM [15] which is one of the third round finalists of the NIST's PQC standardization project. Focusing on key generation and signing procedures, we investigated the target operation of exposing secret key information.

We show that, from a single power trace taken from the implementation of the NTT in signing procedure, one can recover the full secret key of CRYSTALS-DILITHIUM with a 100% success rate, regardless of optimization level. Unlike [8], [9], [14] presented that the sensitive NTT inputs can be extracted by profiling attack and belief propagation technique, we only applied profiling attack to recover the inputs of the NTT. The attack presented in this paper can be applied not only to CRYSTALS-DILITHIUM but also to algorithms that perform the NTT operations of the same implementation.

Applying masking [16], [17] to the signing procedure as a countermeasure against differential power analysis eliminates the advantage of our attack. Therefore, we also investigated the operation of exposing the secret key in the key generation procedure. This paper aims to find secret key vectors s_1 and s_2 of CRYSTALS-DILITHIUM. In the key generation procedure, s_1 is used as the input value of the NTT operation, but s_2 is not. Therefore, when targeting the key generation, further strategies to attack sampling, addition, rounding and packing are required to find the full secret key: We achieved 100% (*resp.* at most 98%) success rate to find s_1 (*resp.* s_2), where $s = (s_1, s_2)$ is the secret key of CRYSTALS-DILITHIUM in its simplified form.

C. ORGANIZATION

The rest of the paper is organized as follows. In Section II, we present notation and review the algorithms of CRYSTALS-DILITHIUM. In Section III, we propose a single-trace attack on CRYSTALS-DILITHIUM. The experimental results of our attacks can be found in Section IV. In Section V, we summarize the conclusion.

II. PRELIMINARIES

In this section, we define the notation used throughout this paper and review the algorithms in CRYSTALS-DILITHIUM.

A. NOTATION

For a positive integer q, we use $\mathbb{Z} \cap (-q/2, q/2]$ as a representative of $\mathbb{Z}_q$. We denote the polynomial rings $R = \mathbb{Z}[X]/(X^N + 1)$ and $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ where $N = 256$ and q is $8380417 = 2^{23} - 2^{13} + 1$. The polynomials in R and R_q are identified with the coefficient vectors in $\mathbb{Z}^N$ and $\mathbb{Z}_q^N$, respectively. We denote secret key vectors of CRYSTALS-DILITHIUM as s_1 and s_2; the bound of the ℓ_∞ norm of s_1 and s_2 of is $\eta \in \{2, 4\}$, i.e., the secret coefficient $s_{i,j,k} \in [-\eta, \eta]$, where $1 \le i \le 2, 0 \le j < L$, and $0 \le k < N$. B_τ is the set of elements of R with τ coefficients of -1 or 1 and the rest of 0.

B. CRYSTALS-DILITHIUM

In this subsection, we briefly review CRYSTALS-DILITHIUM [15] that is one of the most promising finalists of NIST's PQC candidates for standardization. CRYSTALS-DILITHIUM is a lattice-based signature scheme of which security is based on the hardness assumption of

Module Short Integer Solution (MSIS) and Module Learning with Errors (MLWE) in the (quantum) random oracle model. The high level design method is based on the "Fiat-Shamir with Aborts" approach [18], [19]. It contains three tuple of algorithms; Key Generation, Signing, and Verification, the former two of which are presented in Algorithm 1 and 2, respectively. CRYSTALS-DILITHIUM has three parameter sets according to NIST security level 2, 3, and 5. For all parameter sets, they use the same base ring R_q, but use the different sets of (k, ℓ) which determines the number of rows and columns of the public matrix A so that it directly affects public key and signature sizes. (k, ℓ) also represents a pair of the dimension of the module lattices (increasing (k, ℓ) by 1 increases security by $\approx$ 30 bits) and the number of MLWE samples.

NTT and Inverse NTT. In CRYSTALS-DILITHIUM, we use the NTT domain representations of the polynomials in R_q to accelerate polynomial multiplications. For an 8-bit number k, $brv(k)$ denotes the bitreversal of k. The NTT representation of $a \in R_q$ is $\hat{a} = (a(r_0), a(-r_0), \cdots, a(r_{127}), a(-r_{127})) \in \mathbb{Z}_q^{256}$, where $r_i = r^{brv(128+i)} \pmod q$ and $r = 1753$ which is the 512-th root of unity in modulo q. We denote by NTT and NTT^{-1} the NTT operation $a \in R_q \mapsto \hat{a} \in \mathbb{Z}_q^{256}$ and its inverse, respectively. More details can be found in Section III.

Subroutines. CRYSTALS-DILITHIUM utilizes several hash functions as follows. H denotes a cryptographic hash function. ExpandA maps a bitstring $\rho \in \{0,1\}^{256}$ to a matrix $A \in R_q^{k\times\ell}$ in the NTT domain representation. CRH : $\{0,1\}^* \to \{0,1\}^{384}$ is a collision resistant hash function. SampleInBall inputs a random bit string in $\{0,1\}^{256}$ and hashes it onto B_τ.

Also, CRYSTALS-DILITHIUM uses several simple algorithms with an aim to recover higher order bits of $r + z$, given a random $r \in \mathbb{Z}_q$ and small $z \in \mathbb{Z}_q$, without storing z. $\mathsf{Power2Round}_q(r, d)$ outputs a pair of $r_0 = r \pmod{2^d}$ and $r_1 = (r - r_0)/2^d$. For $r \in \mathbb{Z}_q$ and α which divides $q - 1$, letting $r_0 = r \pmod\alpha$ and $r_1 = (r - r_0)/\alpha$, $\mathsf{HighBits}_q(r, \alpha)$ and $\mathsf{LowBits}_q(r, \alpha)$ outputs r_1 and r_0, respectively. $\mathsf{MakeHint}_q(z, r, \alpha)$ outputs 1 if $\mathsf{HighBits}_q(r, \alpha) \neq \mathsf{HighBits}_q(r + z, \alpha)$, and 0 otherwise.

Algorithm 1 Key Generation of **CRYSTALS-DILITHIUM** (Refer to [15])

Ensure: Public key $pk = (\rho, t_1)$, and secret key $sk = (\rho, K, tr, s_1, s_2, t_2)$

$\zeta \leftarrow \{0,1\}^{256}$
$(\rho, \varsigma, K) \in \{0,1\}^{256\times3} := H(\zeta)$
$(s_1, s_2) \in S_\eta^\ell \times S_\eta^k := H(\varsigma)$
/*$\mathbf{A}$ is stored in the NTT representation as $\hat{A}$*/
$A \in R_q^{k\times\ell} := \mathbf{ExpandA}(\rho)$
/*Compute As_1 as $\mathbf{NTT}^{-1}(\hat{A} \cdot \mathbf{NTT}(s_1))$*/
$t := As_1 + s_2$
$(t_1, t_0) := \mathbf{Power2Round}_q(t, d)$
$tr \in \{0,1\}^{384} := \mathbf{CRH}(\rho \parallel t_1)$
Return $pk = (\rho, t_1)$, $sk = (\rho, K, tr, s_1, s_2, t_0)$

Algorithm 2 Signing Procedure of **CRYSTALS-DILITHIUM** (Refer to [15])

Require: Secret key sk and message M
Ensure: Signature $\sigma = (z, h, \tilde{c})$

/*$\mathbf{A}$ is stored in the NTT representation as $\hat{A}$*/
$A \in R_q^{k\times\ell} := \mathbf{ExpandA}(\rho)$
$\mu \in \{0,1\}^{384} := \mathbf{CRH}(tr \parallel M)$
$\kappa := 0, (z, h) := \perp$
/*$\rho \leftarrow \{0,1\}^{384}$ for randomized signing*/
$\rho' \in \{0,1\}^{384} := \mathbf{CRH}(K \parallel \mu)$
$\hat{s}_1 := \mathbf{NTT}(\mathbf{s}_1), \hat{s}_2 := \mathbf{NTT}(\mathbf{s}_2), \hat{t}_0 := \mathbf{NTT}(\mathbf{t}_0)$
while $(z, h) = \perp$ **do**
  $y \in \tilde{S}_{\gamma_1}^\ell$
  $w := Ay$
  $w_1 := \mathbf{HighBits}_q(W, 2\gamma_2)$
  $\tilde{c} \in \{0,1\}^{256} := H(\mu \parallel w_1)$
  /*Store c in the NTT representation as $\hat{c} = \mathbf{NTT}(c)$*/
  $c \in B_\tau := \mathbf{SampleInBall}(\tilde{c})$
  /*Compute cs_1 as $\mathbf{NTT}^{-1}(\hat{c} \cdot \hat{s}_1)$*/
  $z := y + cs_1$
  /*Compute cs_2 as $\mathbf{NTT}^{-1}(\hat{c} \cdot \hat{s}_2)$*/
  $r_0 := \mathbf{LowBits}_q(w - cs_2, 2\gamma_2)$
  if $\|z\|_\infty \geq \gamma_1 - \beta$ or $\|r_0\|_\infty \geq \gamma_2 - \beta$ **then**
    $(z, h) := \perp$
  else
    /*Compute ct_0 as $\mathbf{NTT}^{-1}(\hat{c} \cdot \hat{t}_0)$*/
    $h := \mathbf{MakeHint}_q(-ct_0, w - cs_2 + ct_0, 2\gamma_2)$
    if $\|ct_0\|_\infty \geq \gamma_2$ or (the # of 1's in h) $> w$ **then**
      $(z, h) := \perp$
    end if
    $\kappa := \kappa + \ell$
  end if
end while
Return $\sigma = (z, h, \tilde{c})$

III. PROPOSED SINGLE-TRACE ATTACK ON DILITHIUM

This section proposes a machine learning-based profiling attack on CRYSTALS-DILITHIUM using a sensitive variable-dependent leakage. According to [11], recovering only s_1 allows an attacker to produce a sufficiently forged signature with the attack method for the NTT presented in this paper because it can make the forged signature value that passes the signature verification at a high probability. However, by recovering the whole secret key s_1, s_2, and t_0, we can forge a signature for any chosen messages with a 100% success rate; thus, we attempt to recover not only s_1 but also s_2.

A. LEAKAGE OF SIGNING PROCEDURE

Here, we target step 7 of Algorithm 2 computing the NTT representations of the secret key vectors s_1 and s_2.

We target step 6 of Algorithm 1 computing the NTT representation of the secret key vector s_1. In more detail, we focus on step 12 of Listing 1.

```
// zetas is a precomputed table
void ntt(int32_t a[N])
{
  unsigned int len, start, j, k;
  int32_t zeta, t;

  k = 0;
  for(len = 128; len > 0; len >>= 1) {
    for(start = 0; start < N; start = j + len) {
      zeta = zetas[++k];
      for(j = start; j < start + len; ++j) {
        t = mont_reduce((int64_t)zeta * a[j + len]);
        a[j + len] = a[j] - t;
        a[j] = a[j] + t;
      }
    }
  }
}
```

Listing 1. NTT of CRYSTALS-DILITHIUM (in C code).

In Listing 1, the initial value of *len* is 128, divided by 2 in each loop. Thus, we can divide the NTT operation into eight stages. At m-th stage, the value of *len* is 2^{8-m}, and each stage is consisted of 2^{m-1} substages for $1 \leq m \leq 8$. The input value a of Listing 1 is the polynomial $s_{1,i}$ of the secret key vector s_1, thus, at the first stage, $a[j]$ is the secret coefficient $s_{1,i,j}$, where $0 \leq i < L$ and $0 \leq j < N$.

We redefine the notation of the variables of steps 11 to 15 of Listing 1. In other words, we denote the intermediate values $a[j]$, t and zeta of each first substage of m-th stage as $a^m[j]$, $t^m[j]$, and $zeta^m$, respectively, where $0 \leq j < len$. Accordingly, when the value of *start* is zero, the intermediate value t at step 12 of Listing 1 is described as follow.

$$t^m[j] = \mathsf{mont\_reduce}(zeta^m \times a^m[j + len]), \quad 0 \leq j < len.$$

Similarly, when the value of *start* is zero, the intermediate value at steps 13-14 are described as follow.

$$a^{m+1}[j + len] = a^m[j] - t^m[j];$$
$$a^{m+1}[j] = a^m[j] + t^m[j], \quad 0 \leq j < len.$$

At the first substage of the first stage, the value of $zeta^1$ is fixed as 25847, and the value of $a^1[j + len]$ is the secret coefficient $s_{1,i,j+len}$, where $0 \leq i < L$ and $0 \leq j < len$. Thus, the intermediate value $t^1[j]$ satisfies

$$t^1[j] = \begin{cases} \texttt{0xFFDA47FA}, & if\ s_{1,i,j+len} = \texttt{0xFFFFFFFC}; \\ \texttt{0x0023A5FC}, & if\ s_{1,i,j+len} = \texttt{0xFFFFFFFD}; \\ \texttt{0xFFED23FD}, & if\ s_{1,i,j+len} = \texttt{0xFFFFFFFE}; \\ \texttt{0x003681FF}, & if\ s_{1,i,j+len} = \texttt{0xFFFFFFFF}; \\ \texttt{0x00000000}, & if\ s_{1,i,j+len} = \texttt{0x00000000}; \\ \texttt{0xFFC97E01}, & if\ s_{1,i,j+len} = \texttt{0x00000001}; \\ \texttt{0x0012DC03}, & if\ s_{1,i,j+len} = \texttt{0x00000002}; \\ \texttt{0xFFDC5A04}, & if\ s_{1,i,j+len} = \texttt{0x00000003}; \\ \texttt{0x0025B806}, & if\ s_{1,i,j+len} = \texttt{0x00000004}; \end{cases}$$

and can leak sensitive variable-dependent information.

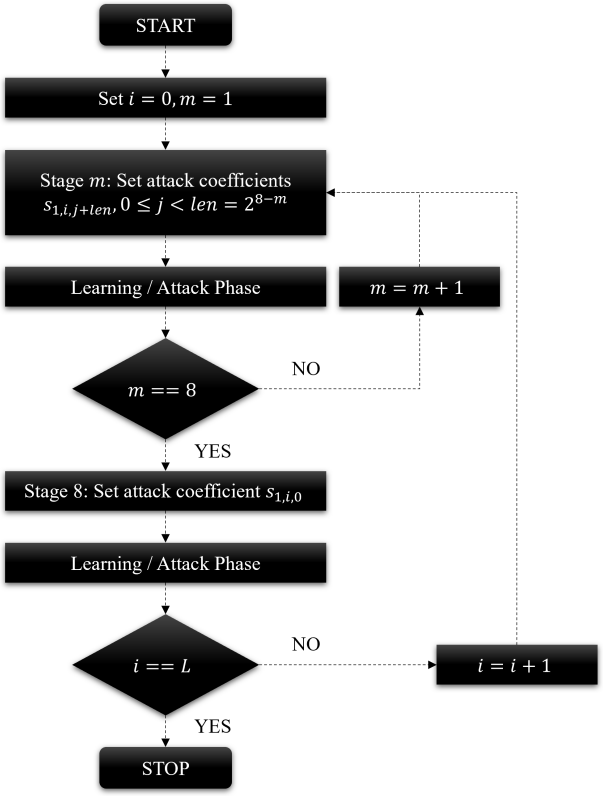


FIGURE 1. Attack flowchart using the leakage of the NTT operation.

Since we target a reference code implemented based on the C language, *i.e.*, software implementation, we suppose that the power consumed proportional to the Hamming weight of an intermediate value. Thus, the distributions of power consumption when $s_{1,i,j} = -2$ and $s_{1,i,j} = -1$ are very similar and difficult to distinguish because the difference of Hamming weight is only 1. However, here is a significant difference in the Hamming weight value of $t^1[j]$ when $s_{1,i,j+len} = -2$ and $s_{1,i,j+len} = -1$.

Accordingly, when $t^1[j]$ is calculated, stored, and loaded, there is a significant difference in power consumption according to the $s_{1,i,j+len}$ value. Therefore, it is more efficient to classify $s_{1,i,j+len}$, used as the input of mont_reduce, than to classify $s_{1,i,j}$, not used as the input of mont_reduce. When we apply ml-based profiling attacks, input and output side-channel information are combined to learn, so in this case, we recommend targeting the value used as the input of mont_reduce. Hence, we can recover $s_{1,i,128}, s_{1,i,129}, \cdots, s_{1,i,255}$, the half of the secret key polynomial $s_{1,i}$, by targeting when $t^1[j]$ is calculated, $0 \leq j < len = 128$.

Similarly, we can also recover $s_{1,i,j+len}$ by targeting $t^m[j]$ in stage from 2 to 8, $0 \leq j < len = 2^{8-m}$, as shown in Figure 1. That is, $s_{1,i,j}$, $0 \leq i < L$, $0 < j < N$ can be recovered by repeatedly attacking step 12 of each first substage before 8-*th* stage. Here, the value of the coefficient restored in the previous step is fixed, and the attack is performed by newly

learning information on the $s_{1,i,j+len}$ value to be attacked. For example, after fixing $s_{1,i,128}, s_{1,i,129}, \cdots, s_{1,i,255}$, the newly collected second stage traces are learned and used to find $s_{1,i,64}, s_{1,i,65}, \cdots, s_{1,i,127}$.

In case of secret coefficient $s_{1,i,0}$, it is not used as the input of Montgomery reduction. However, $s_{1,i,0}$ affects the results of all the coefficients $a^{m+1}[j]$, where $m = 8$. Therefore, we can restore the secret coefficient $s_{1,i,0}$ by summing the information leaking from $a^{m+1}[j]$.

Hence, the secret key vectors s_1 and s_2 are extracted by attacking the NTT operation. However, when applying masking [16], [17] to the signing procedure as a countermeasure, the benefit of our attack is lost, as all input coefficients become uniformly distributed. This motivates us to target the key generation procedure to recover secret key vectors. We described the case where η is 4. However, the proposed method can also be applied if η is 2 in the same manner.

B. LEAKAGE OF KEY GENERATION

Because Algorithm 1 generates secret keys, it is obvious that secret key information is exposed. In this paper, we present operations suitable for profiling attacks through experiments in several different operations. The secret key vector s_1 can be recovered by attacking the NTT operation, as depicted in Section III-A. The secret key vector s_2 can be recovered by attacking sampling, addition, rounding, and packing operations. The rest secret vector t_0 can be calculated using recovered s_1 and s_2.

The secret key vector s_1 can be recovered by attacking Listing 1, thus, this section focus on recovering the secret key vector s_2. In Algorithm 1, there are lots of points expose the secret key vector s_2, such as sampling at step 3, addition at step 7, rounding at step 8, and packing at step 10.

Step 3 of Algorithm 1 is performed with Listing 2, and the output a is the secret key vector s_1 or s_2. Step 7 of Algorithm 1 is performed with Listing 3, thus, the input polynomials a and b are As_1 and s_2, respectively. Step 8 of Algorithm 1 is performed with Listing 4, and the input polynomial a is $As_1 + s_2$. At step 10 of Algorithm 1, packing of s_1 and s_2 is performed with Listing 5, thus, the input polynomial a is s_1 or s_2.

Accordingly, side-channel information of the secret key vector s_2 is directly leaked in Listing 2, Listing 3, and Listing 5. In addition, Listing 4 exposes side-channel information related to s_2. Therefore, the secret key vector s_2 can be found by attacking sampling, addition, rounding, and packing operations in the key generation procedure. Here, A is a public value, and we suppose that s_1 is a value obtained by attacking the NTT operation.

IV. EXPERIMENTAL RESULTS

This section presents experimental results that the secret key vector s_1 is recovered by attacking step 12 of Listing 1. Additionally, experimental results that the secret key vector s_2 is recovered by using the leakage of sampling, addition, rounding, and packing are presented. Since the results of side-channel analysis depend on how to target algorithms are implemented, we use reference codes submitted by developers. 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.

For the ease of the experiment, only the target computational traces were collected to perform the experiments. We measured 60,000 power consumption traces for the different secret key vectors s_1 and s_2 when Listing 1 was operating on the ChipWhisperer UFO STM32F3 target board equipped with an ARM Cortex-M4 [20], and the sampling rate was 29.54 MS/s. In this experiment, 10-fold cross-validation [21] was performed; 45,000 were used for training, 5,000 were

```
static unsigned int rej_eta(int32_t *a,
                            unsigned int len,
                            const uint8_t *buf,
                            unsigned int buflen)
{
  unsigned int ctr, pos;
  uint32_t t0, t1;
  DBENCH_START();

  ctr = pos = 0;
  while(ctr < len && pos < buflen) {
    t0 = buf[pos] & 0x0F;
    t1 = buf[pos++] >> 4;

#if ETA == 2
    if(t0 < 15) {
      t0 = t0 - (205*t0 >> 10)*5;
      a[ctr++] = 2 - t0;
    }
    if(t1 < 15 && ctr < len) {
      t1 = t1 - (205*t1 >> 10)*5;
      a[ctr++] = 2 - t1;
    }
#elif ETA == 4
    if(t0 < 9)
      a[ctr++] = 4 - t0;
    if(t1 < 9 && ctr < len)
      a[ctr++] = 4 - t1;
#endif
  }

  DBENCH_STOP(*tsample);
  return ctr;
}
```

Listing 2. Sample uniformly random coefficients (in C code).

```
void poly_add(poly *c,const poly *a, const poly *b)
{
  unsigned int i;
  DBENCH_START();

  for(i = 0; i < N; ++i)
    c->coeffs[i] = a->coeffs[i] + b->coeffs[i];

  DBENCH_STOP(*tadd);
}
```

Listing 3. Add polynomials of CRYSTALS-DILITHIUM (in C code).

```
int32_t p2r(int32_t *a0, int32_t a)
{
  int32_t a1;

  a1 = (a + (1 << (D-1)) - 1) >> D;
  *a0 = a - (a1 << D);
  return a1;
}

void poly_power2round(poly *a1,poly *a0, const poly *a)
{
  unsigned int i;
  DBENCH_START();

  for(i = 0; i < N; ++i)
    a1->coeffs[i]=p2r(&a0->coeffs[i], a->coeffs[i]);

  DBENCH_STOP(*tround);
}
```

Listing 4. **Rounding of** CRYSTALS-DILITHIUM **(in C code).**

```
void polyeta_pack(uint8_t *r, const poly *a)
{
  unsigned int i;
  uint8_t t[8];
  DBENCH_START();

#if ETA == 2
  for(i = 0; i < N/8; ++i) {
    t[0] = ETA - a->coeffs[8*i+0];
    t[1] = ETA - a->coeffs[8*i+1];
    t[2] = ETA - a->coeffs[8*i+2];
    t[3] = ETA - a->coeffs[8*i+3];
    t[4] = ETA - a->coeffs[8*i+4];
    t[5] = ETA - a->coeffs[8*i+5];
    t[6] = ETA - a->coeffs[8*i+6];
    t[7] = ETA - a->coeffs[8*i+7];

    r[3*i+0] = (t[0]>>0)|(t[1]<<3)|(t[2]<<6);
    r[3*i+1] = (t[2]>>2)|(t[3]<<1)|(t[4]<<4)|(t[5]<<7);
    r[3*i+2] = (t[5]>>1)|(t[6]<<2)|(t[7]<<5);
  }
#elif ETA == 4
  for(i = 0; i < N/2; ++i) {
    t[0] = ETA - a->coeffs[2*i+0];
    t[1] = ETA - a->coeffs[2*i+1];
    r[i] = t[0] | (t[1] << 4);
  }
#endif

  DBENCH_STOP(*tpack);
}
```

Listing 5. **Bit-pack polynomial (in C code).**

used for validation. The remaining 10,000 traces were used for the attack, *i.e.*, 10,000 single-trace attacks were performed. The success rate represents the probability of being correctly classified when a single-trace attack was performed on 10,000 traces. Since this is independent of each coefficient, it can be seen as the success rate of finding the entire key. A network structure for machine learning-based profiling was constructed as depicted in Table 1. It is not an optimized structure, and finding an optimal structure would be an interesting topic for further works. Neural network models

TABLE 1. **Network structure for ML-based PA.**

Layer	node (in, out)	kernel initializer
InputLayer	(x, x)	-
Batch Normalization	(x, x)	-
Dense	$(x, 32)$	he_uniform
Batch Normalization	$(32, 32)$	-
ReLU	$(32, 32)$	-
Dense	$(32, y)$	he_uniform
Softmax	(y, y)	-

* Input Normalization: all values are within the range of -1 and 1
* Loss function: categorical_crossentropy
* Optimizer: Nadam (lr=0.0001, epsilon=1e-08)
* Label encoding: one-hot encoding
* Batch size and epochs: 32 and maximum 500, respectively

were implemented using Python, using the Keras library with TensorFlow as backend.

Here, x represents the number of points in each trace, and y is the number of classification labels. The label moved the value between -4 and 4 by 4 to make it between 0 and 8. The coefficient $s_{1,i,j}$ a 32 bits value and can have only one of the $2\eta+1$ values $(-\eta, \cdots, -1, 0, 1, \cdots,$ and $\eta)$. Therefore, if the last 8 bits can be distinguished, the remaining 24 bits are automatically determined. The maximum of y is nine because the maximum value of $2\eta+1$ is nine. Since we applied 10-fold cross-validation, Table 2 shows the average values of success rates for ten profiled models in Table 1.

A. ATTACK ON NUMBER-THEORETIC TRANSFORM

In this subsection, we propose the attack against the NTT. Figure 2 and Figure 3 show the results of learning a model that finds $s_{1,i,128}$ and $s_{1,i,0}$, respectively, using the power consumption traces of the first stage, $0 \le i < L$, $0 \le j < N$. As shown in Figure 2, the validation accuracy converges to 1 very quickly because points, when the difference in side-channel leakage of the $t^1[0]$ values depending on the value of $s_{1,i,128}$ is significant, are also utilized for learning.

On the other hand, the validation accuracy of Figure 3 does not converge to 1 even though 500 epochs, and it increased very slowly. The single-trace attack success rates finding $s_{1,i,128}$ and $s_{1,i,0}$ for 10 profiled models in Table 1 are 100% and 90.27%, respectively. At optimization levels s, 100% and 92.91%, respectively. Thus, we can recover the secret coefficient $s_{1,i,j}$, $0 \le i < L$, $0 < j < N$ with a 100% success rate using a single trace regardless of an optimization level by targeting $s_{1,i,j+len}$, used as the input of **mont_reduce**, when calculating $t^m[j]$, $1 \le m \le 7$ and $0 \le j < len = 2^{8-m}$.

As shown in Figure 4, there are lots of points in power consumption traces at stage 8 are affected with $s_{1,i,0}$. Thus, $s_{1,i,0}$ can be recovered by summing the information leaking at stage 8. That is, the distributed information is automatically combined during the learning phase. Figure 5 shows the

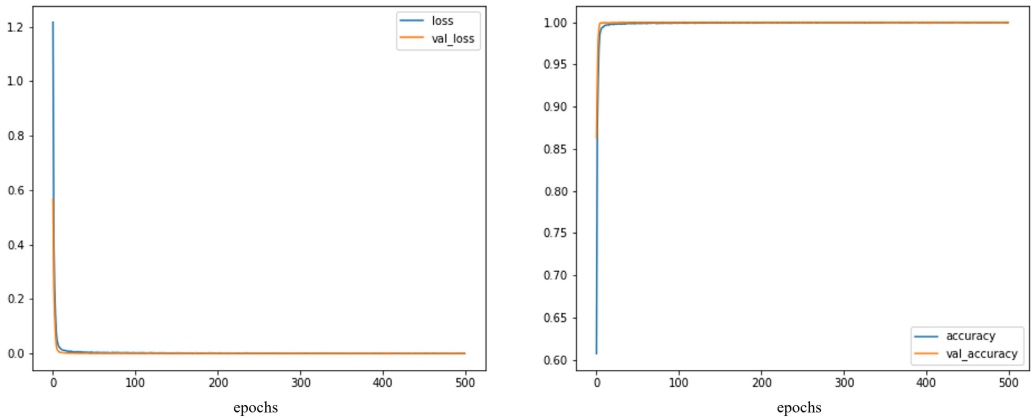


FIGURE 2. Attack on stage 1 of the NTT operation targeting $s_{1,i,128}$ (optimization level 3).

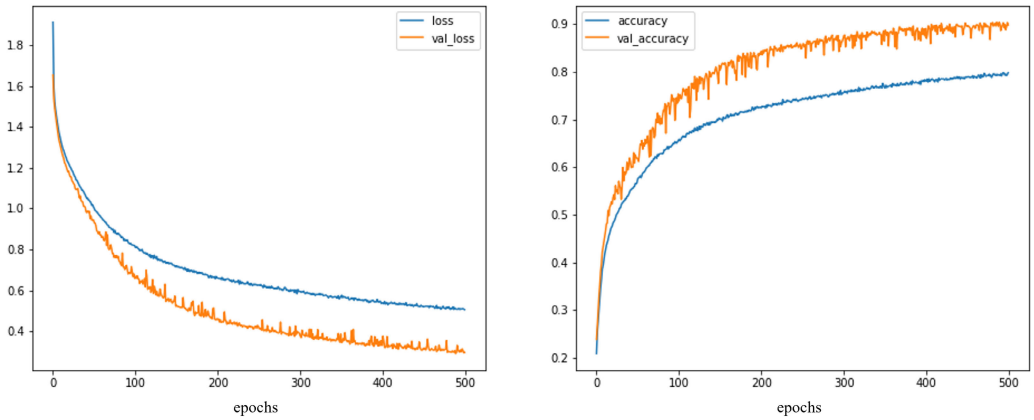


FIGURE 3. Attack on stage 1 of the NTT operation targeting $s_{1,i,0}$ (optimization level 3).

TABLE 2. The success rates of the single-trace attacks on the NTT operation.

Optimization Level	Stage m: $s_{1,i,j},\ 1 \leq m < 8,\ 0 \leq i < L,\ 0 < j < N$	Stage 8: $s_{1,i,0},\ 0 \leq i < L$
`-O3`	100%	100%
`-Os`	100%	100%

TABLE 3. The success rates of the single-trace attacks on sampling, addition, rounding, and packing.

Optimization Level	Sampling	Addition	Rounding	Packing
`-O3`	94.03	98.38	89.49	76.04
`-Os`	95.38	98.47	91.21	75.67

result of learning a model that finds $s_{1,i,0}$, using the power consumption traces of stage 8, and $s_{1,i,0}$ is recovered with a 100% success rate using a single trace regardless of an optimization level.

As a result, when we target the key generation, the secret key vector s_1 can be extracted with a 100% success rate using a single trace; when we target the signing procedure, the secret key vector s_1 and s_2 can be extracted.

B. ATTACK ON SAMPLING, ADDITION, ROUNDING AND PACKING

Because there is no NTT operation for s_2 in the key generation, attacks on sampling, addition, rounding and packing are needed to find s_2. The success rates of the single-trace attacks on each operation is shown in Table 3. Attacking addition operation shows the best results with a success rate over 98% and attacking packing operation shows the worst. Similar to

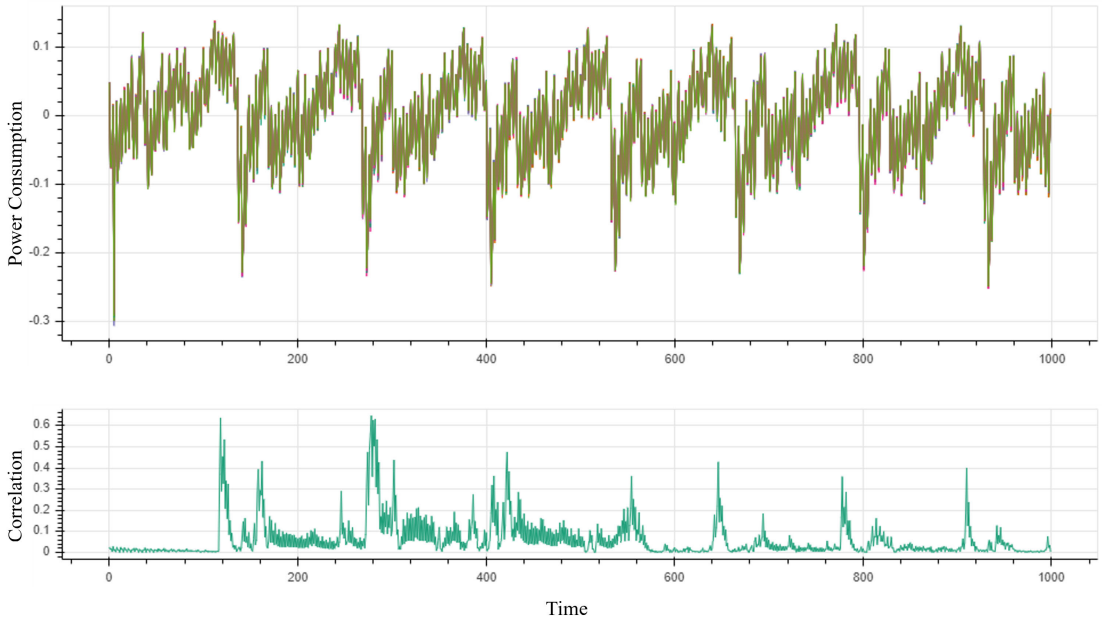


FIGURE 4. Correlation coefficients with $s_{1,i,0}$ at stage 8 (optimization level 3).

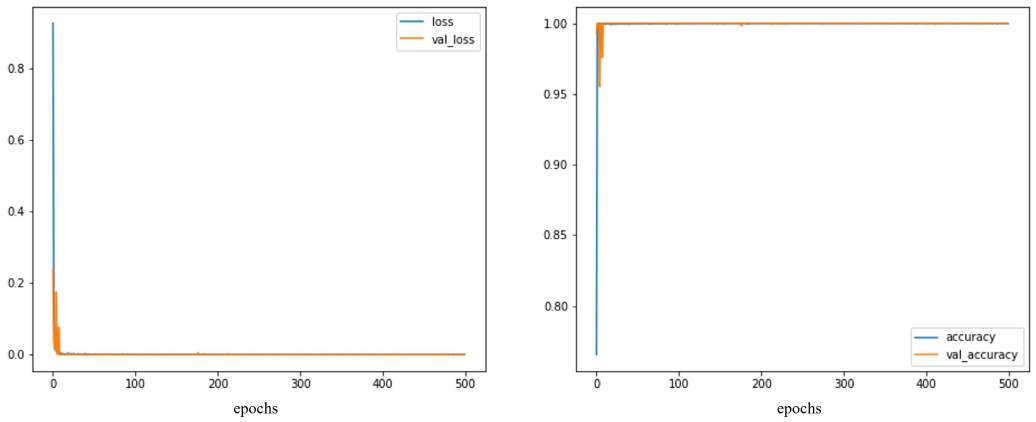


FIGURE 5. Attack on stage 8 of the NTT operation targeting $s_{1,i,0}$ (optimization level 3).

attack on stage 1 of the NTT operation targeting $s_{1,i,0}$, it is not easy to distinguish secret coefficients having the Hamming weight difference 1. Since s_2 has a success rate of $< 100\%$, we consider setting a threshold that is not misclassified and conducting an exhausting search on coefficient values that do not exceed the threshold. This is not covered in this paper and is left for further works.

V. CONCLUSION

This paper proposed single-trace attacks on the NTT operation and showed that the secret key vector of CRYSTALS-DILITHIUM recovered with a 100% success rate. Accordingly, the secret key vectors s_1 and s_2 can be found by attacking the NTT operation in the signing procedure. When using masking as a countermeasure of the signing procedure, we targeted the key generation. The secret key vector s_1 would be discovered by targeting the NTT operation with a 100% success rate. To find the rest secret key vector s_2, attacking sampling, addition, rounding, and packing operations are needed. Because the key generation is carried out once in the initial setup phase, only a single trace can be used during the attack. Even though single-trace attacks could not guarantee 100% success rates when targeting sampling, addition, rounding, and packing, we achieved at most 98% success rate.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, Nov. 1994, pp. 124–134.

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

[3] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1996, pp. 104–113.

[4] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone. (2020). *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*. [Online]. Available: https://csrc.nist.gov/publications/detail/nistir/8309/final

[5] 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 *Proc. 40th Annu. Int. Cryptol. Conf.* (Lecture Notes in Computer Science), vol. 12171, D. Micciancio and T. Ristenpart, Eds. Santa Barbara, CA, USA: Springer, 2020, pp. 359–386.

[6] W.-L. Huang, J.-P. Chen, and B.-Y. Yang, "Power analysis on NTRU prime," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, pp. 123–151, Nov. 2019.

[7] A. Facon, S. Guilley, M. Lec'Hvien, A. Schaub, and Y. Souissi, "Detecting cache-timing vulnerabilities in post-quantum cryptography algorithms," in *Proc. IEEE 3rd Int. Verification Secur. Workshop (IVSW)*, Jul. 2018, pp. 7–12.

[8] P. Pessl and R. Primas, "More practical single-trace attacks on the number theoretic transform," in *Proc. Int. Conf. Cryptol. Inf. Secur. Latin Amer.* Cham, Switzerland: Springer, 2019, pp. 130–149.

[9] R. Primas, P. Pessl, and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.* Cham, Switzerland: Springer, 2017, pp. 513–533.

[10] P. Pessl and L. Prokop, "Fault attacks on CCA-secure lattice KEMs," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, pp. 37–60, Feb. 2021.

[11] P. Ravi, M. P. Jhanwar, J. Howe, A. Chattopadhyay, and S. Bhasin, "Side-channel assisted existential forgery attack on dilithium–A NIST PQC candidate," IACR Cryptol. ePrint Arch., Tech. Rep. 821, 2018.

[12] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS–kyber: A CCA-secure module-lattice-based KEM," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Apr. 2018, pp. 353–367.

[13] A. P. Fournaris, C. Dimopoulos, and O. G. Koufopavlou, "Profiling dilithium digital signature traces for correlation differential side channel attacks," in *Proc. SAMOS* (Lecture Notes in Computer Science), vol. 12471, A. Orailoglu, M. Jung, and M. Reichenbach, Eds. Samos, Greece: Springer, 2020, pp. 281–294.

[14] M. Hamburg, J. Hermelink, R. Primas, S. Samardjiska, T. Schamberger, S. Streit, E. Strieder, and C. Van Vredendaal, "Chosen ciphertext K-trace attacks on masked CCA2 secure kyber," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, no. 4, pp. 88–113, Aug. 2021.

[15] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehle. (2020). *CRYSTALS-Dilithium : Algorithm Specifications and Supporting Documentation*. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions

[16] O. Reparaz, S. S. Roy, F. Vercauteren, and I. Verbauwhede, "A masked ring-LWE implementation," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2015, pp. 683–702.

[17] O. Reparaz, S. Roy, R. de Clercq, F. Vercauteren, and I. Verbauwhede, "Masking ring-LWE," *J. Cryptograph. Eng.*, vol. 6, no. 2, pp. 139–153, 2016.

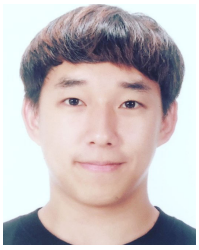
[18] V. Lyubashevsky, "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures," in *Advances in Cryptology–(ASIACRYPT)*, M. Matsui, Ed. Berlin, Germany: Springer, 2009, pp. 598–616.

[19] S. Bai and S. D. Galbraith, "An improved compression technique for signatures based on learning with errors," in *Topics in Cryptology–(CT-RSA)*, J. Benaloh, Ed. Cham, Switzerland: Springer, 2014, pp. 28–47.

[20] NT *ChipWhisperer UFO*. [Online]. Available: https://wiki.newae.com/CW308T-STM32F

[21] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics), vol. 4. New York, NY, USA: Springer, 2006.

JAESEUNG HAN received the B.S. degree in information security, cryptology, and mathematics from Kookmin University, Seoul, Republic of Korea, in 2020, where he is currently pursuing the master's degree in financial information security. His research interests include side-channel attacks, symmetric key cryptography, and lattice-based cryptography.



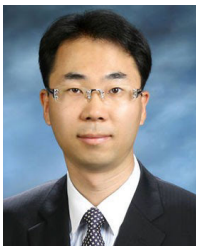
TAEHO LEE received the B.S. degree in information security, cryptology, and mathematics from Kookmin University, Seoul, Republic of Korea, in 2020, where he is currently pursuing the master's degree in financial information security. His specialty lies in the area of information security, and his research interests include the side-channel-analysis of symmetric key and post quantum cryptosystems, and SIM card and IC card.

JIHOON KWON received the B.S. degree in mathematics and the Ph.D. degree in information security from Korea University, Seoul, South Korea, in 2010 and 2018, respectively. He is currently a Senior Engineer with the Security Algorithm Team, Samsung SDS, Seoul. His research interests include cryptography and information security, and their efficient implementation.

JOOHEE LEE received the B.S. degree in mathematical education from Korea University, Seoul, South Korea, in 2013, and the Ph.D. degree from the Department of Mathematical Science, Seoul National University (SNU), South Korea, in 2019. She is currently a Senior Engineer with the Security Algorithm Team, Samsung SDS. Her current research interests include lattice cryptography, cryptographic protocols, and information security.

IL-JU KIM received the B.S. degree in mathematics from Kookmin University, Seoul, Republic of Korea, in 2019, where he is currently pursuing the master's degree in financial information security. His specialty lies in the area of information security. His research interests include the side-channel-analysis, lattice-based cryptography, code-based cryptography, and financial IC card.

JIHOON CHO received the M.Math. degree in cryptography from the University of Waterloo, and the Ph.D. degree in information security from the Royal Holloway, University of London. He is currently the Director of security research at Samsung SDS, South Korea. Previously, he worked as a Security Architect for mobile devices at LG Electronics, South Korea.



DONG-GUK HAN received the B.S. and M.S. degrees in mathematics from Korea University, Seoul, Republic of Korea, in 1999 and 2002, respectively, and the Ph.D. degree in engineering in information security from Korea University, in 2005. He was a Postdoctoral Researcher at Future University Hakodate, Hokkaido, Japan. After finishing his doctoral course, he was then an Exchange Student with the Department of Computer Science and Communication Engineering, Kyushu University, Japan, from April 2004 to March 2005. From 2006 to 2009, he was a Senior Researcher with the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea. He is currently working as a Professor with the Department of Information Security, Cryptology, and Mathematics, Kookmin University, Seoul. He is a member of KIISC, IEEK, and IACR.

BO-YEON SIM received the Ph.D. degree in information security from Kookmin University, Seoul, Republic of Korea, in 2020. She worked as a Research Professor with Kookmin University, in 2020. She is currently working as a Researcher with the Electronics and Telecommunications Research Institute (ETRI). Her research interests include side-channel attacks, cryptography, reverse engineering, and implementation of information protection technology for embedded systems.