Decompressing Dilithium's Public Key with Fewer Signatures Using Side Channel Analysis

Ruize Wang, Joel Gärtner, Elena Dubrova
KTH Royal Institute of Technology
Stockholm, Sweden
{ruize, jgartner, dubrova}@kth.se

***Abstract*—The CRYSTALS-Dilithium digital signature scheme, selected by NIST as a post-quantum cryptography (PQC) standard under the name ML-DSA, employs a public key compression technique intended for performance optimization. Specifically, the module learning with error instance $(\mathbf{A}, \mathbf{t})$ is compressed by omitting the low-order bits $\mathbf{t_0}$ of the vector $\mathbf{t}$. It was recently shown that knowledge of $\mathbf{t_0}$ enables more effective side-channel attacks on Dilithium implementations. Another recent work demonstrated a method for reconstructing $\mathbf{t_0}$ from multiple signatures. In this paper, we build on this method by applying profiled deep learning-assisted side-channel analysis to partially recover the least significant bit of $\mathbf{t_0}$ from power traces. As a result, the number of signatures required for the reconstruction of $\mathbf{t_0}$ can be reduced by roughly half. We demonstrate how the new $\mathbf{t_0}$ reconstruction method enhances the efficiency of recovering the secret key component $\mathbf{s}_1$, and thus facilitates digital signature forgery, on an ARM Cortex-M4 implementation of Dilithium.**

***Index Terms*—Public-key cryptography, post-quantum cryptography, Dilithium, ML-DSA, side-channel attack.**

I. Introduction

The advent of quantum computing poses significant challenges to classical public-key cryptographic schemes. In response, the National Institute of Standards and Technology (NIST) initiated the standardization of post-quantum cryptography (PQC) primitives in 2016. Following an eight-year evaluation process, the Federal Information Processing Standards (FIPS) for CRYSTALS-Kyber, CRYSTALS-Dilithium, and Sphinx+ have recently been approved under the names FIPS 203 ML-KEM, FIPS 204 ML-DSA, and FIPS 205 SLH-DSA, respectively [1].

The state-of-the-art in PQC is quickly progressing from standardization to implementation and deployment. The Internet Engineering Task Force (IETF) is currently integrating quantum-resistant cryptographic algorithms into security protocols such as X.509, Transport Layer Security (TLS), Internet Protocol Security (IPsec), and Secure Shell (SSH), which are commonly used in the Internet, corporate networks, and mobile networks. Similarly, the 3rd Generation Partnership Project (3GPP) is in the process of integrating PQC algorithms into 5G [2]. NIST expects that all high-priority systems transition to quantum-resistant cryptography by 2035. While quantum computers powerful enough to break current public-key cryptographic schemes do not yet exist, the transition to PQC is necessary due to the threat of adversaries collecting encrypted data now with the intention of decrypting it later, once the technology becomes available.

These developments highlight the need to assess the resistance of standardized PQC algorithms to various types of attacks, including physical attacks on their implementations. Each identified implementation weakness presents an opportunity for developers to improve future versions, ultimately resulting in more secure cryptographic systems.

The ARM Cortex-M4 is an important platform for evaluations, as demonstrated by NIST's emphasis on this architecture during the performance assessment of PQC standardization process candidates [3]. The ARM Cortex-M4 is widely used in internet-of-things devices such as smart locks, smart appliances, wearables, vending machines, in-car entertainment systems, etc. These devices are particularly susceptible to side-channel and fault attacks due to their physical accessibility, with attackers potentially being the devices' users themselves.

This paper focuses on the evaluation of Dilithium's implementations on ARM Cortex-M4. Dilithium digital signature scheme is strongly existentially unforgeable under chosen message attack (EUF-CMA-secure) in the classical and quantum random oracle models [4]. In theory, this ensures that adversaries cannot forge a signature for a new message or alter an existing signature for a message they have already seen. However, previous work has demonstrated that Dilithium's theoretical EUF-CMA-security can be bypassed via side-channel attacks [5]–[11] or fault attacks [12], [13].

Our Contributions: We present a side-channel attack on Dilithium that targets the recovery of the low-order bits $\mathbf{t_0}$ of the vector $\mathbf{t}$. None of the previous attacks on Dilithium specifically target $\mathbf{t}_0$. This is likely because $\mathbf{t}_0$ is not considered a secret contributing to the security of Dilithium. The compression of $\mathbf{t}$ is applied to optimize performance, and cryptanalytic methods, such as [14], can reconstruct $\mathbf{t_0}$ from multiple signatures.

Although the knowledge of $\mathbf{t_0}$ does not enhance traditional cryptanalysis, it facilitates more effective side-channel attacks on Dilithium implementations, as demonstrated in [11]. To minimize the number of signatures required for the reconstruction of $\mathbf{t_0}$ by a cryptanalytic method, attackers may attempt to recover portions of $\mathbf{t}_0$ through side-channel analysis. For example, in the case of Dilithium-2, reconstructing the entire $\mathbf{t}_0$ using a method outlined in [14] requires two

million signatures. However, if the least significant bit (LSB) of $\mathbf{t_0}$ is known, the number of signatures required for the reconstruction of $\mathbf{t}_0$ can be reduced by approximately half. This motivated us to explore the feasibility of recovering the LSB of $\mathbf{t_0}$ through side-channel analysis.

We demonstrate that the LSB of $\mathbf{t_0}$ can be partially recovered from a small number of power traces using side-channel information leaked during the secret key unpacking in the signing algorithm. The attack employs a profiled deep learning-assisted power analysis. It utilizes two novel ideas:

1) *Detecting LSB errors:* During the recovery of the LSB of $\mathbf{t_0}$ by neural network ensembles trained at the profiling stage, we detect potential errors in the predictions using the following strategy. Instead of relying on the majority voting aggregation method commonly employed in ensemble learning, we apply the *unanimous voting with error detection*. In this method, the ensemble makes a decision on the LSB value only if all the neural networks in the ensemble agree. Otherwise, the LSB remains undecided, indicating a possible error.
2) *Recovering $\mathbf{s}_1$ with partial knowledge of $\mathbf{t_0}$*: We extend the atatck method of Wang et al. [11] to recover the vector $\mathbf{s}_1$ (a component of the secret key) from linear equations based on $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ in which some coefficients of the vector $\mathbf{t_0}$ are unknown. This modification is necessary to account for the presence of undecided LSB values.

Paper Organization: In Section II, we describe how $\mathbf{t_0}$ can be reconstructed from multiple signatures and why knowing $\mathbf{t}_0$ aids in extracting $\mathbf{s}_1$ via side-channel analysis. Section III presents the $\mathbf{t_0}$ LSB recovery method. Results of the attacks are summarized in Section IV. Section V concludes the paper.

II. Role of $\mathbf{t_0}$ in Secret Key Recovery

A. Reconstructing $\mathbf{t_0}$ from multiple signatures

The method of Oliveira et al. [14] recovers $\mathbf{t}_0$ by using the fact that, for each signature, the hint vector $\mathbf{h}$ leaks some information about $\mathbf{t}_0$. Specifically, each coefficient of $\mathbf{h}$ provides an inequality that bounds the range of possible values for the coefficients of $\mathbf{t}_0$. By accumulating sufficiently many such inequalities, the entire vector $\mathbf{t}_0$ can be recovered. Both, hint coefficients equal to zero and one, contribute to this process.

In [14], it shows that approximately two million signatures are sufficient to recover $\mathbf{t}_0$ when both types of inequalities, from zero and one hints, are considered. This is because the inequalities reduce the range of possible values for each coefficient of $\mathbf{t}_0$ to within 0.5. Since the coefficients are integers, this results in a unique solution.

The recovery of $\mathbf{t}_0$ is an iterative, multi-round process, where each round $i \in \{1, 2, \ldots, 14\}$ progressively narrows down the range C_i of possible values for the coefficients of $\mathbf{t}_0$. When C_i shrinks to 1, each coefficient of $\mathbf{t}_0$ becomes uniquely defined within the range of 1, i.e. up to its LSB. Consequently, if the LSBs of the coefficients of $\mathbf{t}_0$ are already known, the algorithm of [14] can terminate as soon as $C_i = 1$. From [14], achieving $C_i = 1$ requires approximately one million signatures. Thus, knowing the LSBs reduces the total number of signatures required to recover $\mathbf{t}_0$ by approximately half.

Note that the method described in [14] only considers the recovery of the first element of the $\mathbf{t}_0$ vector. To recover the other elements of $\mathbf{t}_0$, the algorithm has to be run again. However, the same signatures can be reused, so recovering the entire $\mathbf{t}_0$ is possible with the same number of signatures as required for recovering the first element of $\mathbf{t}_0$.

B. How knowledge of $\mathbf{t}_0$ helps recover the secret $\mathbf{s}_1$

The experiments of [11] show that, for Dilithium-2, the average probability to recover a single coefficient of the vector $\mathbf{s}_1$ during unpacking is 0.964. Thus, the estimated probability to recover the full vector $\mathbf{s}_1$ is $0.964^{1024} \approx 0$.

The solution proposed in [11] is to first predict values of all 2048 coefficients of the secret vector $\mathbf{s}_1$ and the error vector $\mathbf{s}_2$ using neural networks, then to sort the predictions in descending order according to the maximum probability of their score vectors, and finally to accept as correct the top half of the predictions in the sorted list. The remaining half of the coefficients are recovered by solving a system of linear equations based on $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. If $\mathbf{t}_0$ is known, 1024 coefficients recovered by side-channel analysis are sufficient for these linear equations to have a unique solution.

In this paper, we extend this method to handle cases where some of the coefficients of the vector $\mathbf{t}_0$ are unknown. This restricts us to only using the equations of $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ where the coefficients of $\mathbf{t}_0$ are known. The same total number of coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$ must still be recovered by side-channel analysis to ensure that the linear system of equations has a unique solution. However, the coefficients of $\mathbf{s}_2$ that correspond to unknown parts of $\mathbf{t}_0$ are no longer of interest, and therefore a larger fraction of the interesting unknown coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$ must be recovered by side-channel analysis. As the system of equations no longer depends on all coefficients of $\mathbf{s}_2$, the entire $\mathbf{s}_2$ vector cannot be reconstructed in this way. However, this is not an issue as, for forging signatures, it is sufficient to fully reconstruct the $\mathbf{s}_1$ vector. The details of the extended method are described in Section IV-C.

III. LSB Recovery by Side-Channel Analysis

This section describes the deep learning-assisted power analysis method which we use for recovering the LSB of $\mathbf{t_0}$.

A. Attack point

The presented side-channel attack targets the secret key unpacking function executed during the first step of the signing algorithm. Fig. 1 shows the C code of the procedure `polyt0_unpack()` which unpacks $\mathbf{t_0}$ in the Dilithium implementation of [15]. The function `polyt0_unpack()` is invoked to unpack 1024 coefficients of $\mathbf{t_0}$ which are 13-bit signed integers.

```
void polyt0_unpack(poly *r, uint8_t *a)
/* a is the input byte array of t_0 in the secret key*/
/* r is the corresponding output array of polynomial coefficients of t_0 */
unsigned int i;
1: for (i = 0; i < N/8; ++i)  do /* N = 256, D = 13 in Dilithium-2 */
2:    r->coeffs[8*i+0] = a[13*i+0];
3:    r->coeffs[8*i+0] |= (uint32_t)a[13*i+1] << 8;
4:    r->coeffs[8*i+0] &= 0x1FFF;

5:    r->coeffs[8*i+1] = a[13*i+1] >> 5;
6:    r->coeffs[8*i+1] |= (uint32_t)a[13*i+2] << 3;
7:    r->coeffs[8*i+1] |= (uint32_t)a[13*i+3] << 11;
8:    r->coeffs[8*i+1] &= 0x1FFF;

9:    r->coeffs[8*i+2] = a[13*i+3] >> 2;
10:   r->coeffs[8*i+2] |= (uint32_t)a[13*i+4] << 6;
11:   r->coeffs[8*i+2] &= 0x1FFF;

12:   r->coeffs[8*i+3] = a[13*i+4] >> 7;
13:   r->coeffs[8*i+3] |= (uint32_t)a[13*i+5] << 1;
14:   r->coeffs[8*i+3] |= (uint32_t)a[13*i+6] << 9;
15:   r->coeffs[8*i+3] &= 0x1FFF;

16:   r->coeffs[8*i+4] = a[13*i+6] >> 4;
17:   r->coeffs[8*i+4] |= (uint32_t)a[13*i+7] << 4;
18:   r->coeffs[8*i+4] |= (uint32_t)a[13*i+8] << 12;
19:   r->coeffs[8*i+4] &= 0x1FFF;

20:   r->coeffs[8*i+5] = a[13*i+8] >> 1;
21:   r->coeffs[8*i+5] |= (uint32_t)a[13*i+9] << 7;
22:   r->coeffs[8*i+5] &= 0x1FFF;

23:   r->coeffs[8*i+6] = a[13*i+9] >> 6;
24:   r->coeffs[8*i+6] |= (uint32_t)a[13*i+10] << 2;
25:   r->coeffs[8*i+6] |= (uint32_t)a[13*i+11] << 10;
26:   r->coeffs[8*i+6] &= 0x1FFF;

27:   r->coeffs[8*i+7] = a[13*i+11] >> 3;
28:   r->coeffs[8*i+7] |= (uint32_t)a[13*i+12] << 5;
29:   r->coeffs[8*i+7] &= 0x1FFF;

30:   r->coeffs[8*i+0] = (1 << (D-1)) - r->coeffs[8*i+0];
31:   r->coeffs[8*i+1] = (1 << (D-1)) - r->coeffs[8*i+1];
32:   r->coeffs[8*i+2] = (1 << (D-1)) - r->coeffs[8*i+2];
33:   r->coeffs[8*i+3] = (1 << (D-1)) - r->coeffs[8*i+3];
34:   r->coeffs[8*i+4] = (1 << (D-1)) - r->coeffs[8*i+4];
35:   r->coeffs[8*i+5] = (1 << (D-1)) - r->coeffs[8*i+5];
36:   r->coeffs[8*i+6] = (1 << (D-1)) - r->coeffs[8*i+6];
37:   r->coeffs[8*i+7] = (1 << (D-1)) - r->coeffs[8*i+7];
38: end for
```

Fig. 1. The C code of the `polyt0_unpack()` procedure which unpacks $\mathbf{t_0}$ in the Dilithium implementation of [15].

Fig. 2(a) shows a complete power trace recorded during the execution of the Dilithium-2 signing algorithm in the implementation [15]. We capture traces using the same equipment as in [11]. The trace covers the entire execution of `polyt0_unpack()` procedure. One can clearly see four similarly-looking blocks, each corresponding to the unpacking of 256 polynomial coefficients of the vector $\mathbf{t_0}$. Within each block, there are 32 identical segments, each representing 13 bytes used to pack eight coefficients of $\mathbf{t_0}$ (see line 2-37 in Fig. 1). Fig. 2(b) provides a closer view of a 550-point segment representing the unpacking of eight consecutive coefficients of $\mathbf{t_0}$. Such segments are provided as input to the neural networks during training and inference.

B. Leakage analysis

We conduct leakage analysis by applying Welch's t-test [16] on a set of power traces, $\mathbf{T}$, captured from a profiling device executing the signing algorithm with a known secret key. The

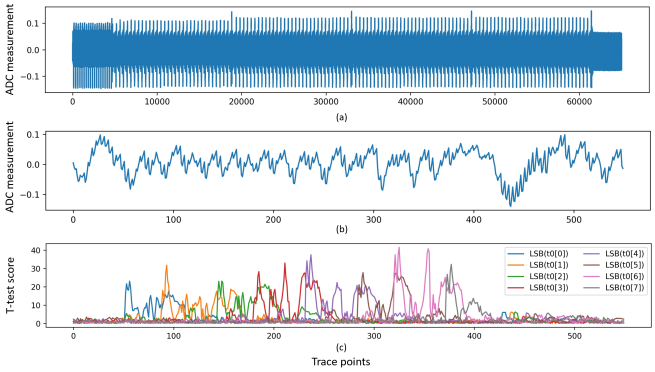

Fig. 2. (a) A power trace of the `polyt0_unpack()` procedure in Dilithium-2 (averaged over 30 samples); (b) 550-point segment representing the unpacking of eight consecutive $\mathbf{t_0}$ coefficients; (c) T-test results for LSBs of the coefficients in (b) for $10,000$ traces.

Welch's t-test determines if there is a noticeable difference in the means of two datasets, $\mathbf{T_0}$ and $\mathbf{T_1}$, by computing:

$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{\sigma_0^2}{|\mathbf{T_0}|} + \frac{\sigma_1^2}{|\mathbf{T_1}|}}},$$

where μ_i and σ_i are the mean and the standard deviation of the dataset $\mathbf{T_i}$, for $i \in \{0, 1\}$.

Let $\mathbf{t}_0[i]$ denote the ith coefficient of $\mathbf{t}_0$, for $i \in \{0, 1, \ldots, 1023\}$. To analyze if the LSBs of $\mathbf{t}_0[i]$ potentially leaks side-channel information, we group the traces of $\mathbf{T}$ according to the value of the LSB:

$$\begin{aligned} \mathbf{T}_0 &= \{T \in \mathbf{T} \mid LSB(\mathbf{t}_0[i]) = 0\}, \\ \mathbf{T}_1 &= \{T \in \mathbf{T} \mid LSB(\mathbf{t}_0[i]) = 1\}. \end{aligned} \tag{1}$$

Fig. 2(c) shows the t-test results for the first eight coefficients of $\mathbf{t_0}$. We can see that all eight coefficients have two main peaks. The first one corresponds to converting elements of the byte array to 13-bit unsigned integers (see line 2-29 in Fig. 1). The second peak corresponds to converting the resulting unsigned integers to 13-bit signed integers (see line 30-37 in Fig. 1). The maximum t-test score for each coefficient is listed in the first row of Table I.

We can also see from Fig. 2(c) that each coefficient of $\mathbf{t}_0$ within a group of eight consecutive coefficients exhibits distinct leakage patterns. This is due to differences in the unpacking of individual coefficients by the `polyt0_unpack()` procedure (see the pseudocode in Fig. 1). For this reason we train a different neural network for each of the eight coefficients.

C. Noise reduction

It is well-known that, in order to distinguish the variations due to the data-dependent part of cryptographic operations from noise, the signal-to-noise ratio (SNR) should be sufficiently high [17]. It is also known that the SNR of an averaged signal increases proportionally to the square root of the number of samples which are averaged [18].

TABLE I
MEAN T-TEST SCORES FOR LSBS($\mathbf{t}_0[i]$) GROUPED BY (i MOD 8), FOR $i \in 0, 1, \ldots, 1023$, FOR $10,000$ TRACES AVERAGED OVER M SAMPLES.

M	i mod 8								Avg.
	0	1	2	3	4	5	6	7	
1	23.1	31.8	23.1	33.0	37.5	27.8	41.6	32.3	31.3
10	33.1	45.7	32.5	60.4	39.0	34.5	55.4	34.7	42.0
20	33.6	45.2	37.1	65.6	40.3	35.9	59.0	35.8	44.1
30	34.3	45.3	40.1	67.9	41.8	36.6	59.7	35.9	45.2

To estimate how many samples are required to be averaged to increase the SNR, we capture four sets of $10,000$ traces, each computed as an average of $M = 1$, 10, 20, and 30 samples, respectively, for a fixed secret key selected at random. Table I shows the t-test results for the LSBs of the coefficients $t_0[i]$ grouped according to the index (i mod 8), for all $i \in \{0, 1, \ldots, 1023\}$.

As expected, the t-test score increases as M grows. However, the rate of increase slows significantly once M reaches 30. Since capturing $10,000$ traces averaged over 40 samples takes more than a week, we chose to use $M = 30$ for our experiments.

D. Profiling details

For profiling, we capture $4,000$ power traces from each of the five profiling devices, with every trace averaged over 30 samples, during the execution of the signing algorithm for known secret keys and messages selected at random. The 550-point segments of type shown in Fig. 2(b), representing the unpacking of eight consecutive coefficients of $\mathbf{t}_0$, are then extracted and combined. This results in a training set, $\mathbf{T}_{tr}$, of size $20,000 \times 32 \times 4 = 2,560,000$ traces.

We use a multilayer perceptron (MLP) architecture with three layers containing of 512, 256, and 256 neurons. Using $\mathbf{T}_{tr}$, we train eight types of neuron networks, $\mathcal{N}_j$, to predict the LSBs of $\mathbf{t}_0[i]$, for all $i \in \{0, 1, \ldots, 1023\}$ and $j = (i \bmod 8)$. For each $j \in \{0, 1, \ldots, 7\}$, 21 models are trained on randomly selected subsets containing 70% of the traces in $\mathbf{T}_{tr}$. These models are then combined into an ensemble, leveraging ensemble learning which has proven effective in side-channel analysis [19]. In total, $21 \times 8 = 168$ neural networks are trained.

IV. EXPERIMENTAL RESULTS

In this section, we evaluate the feasibility of recovering the LSB of $\mathbf{t_0}$ for 10 different secret keys selected at random. Power traces for all side-channel attacks are collected from a device different from those used for profiling. As in neural network training, each trace is averaged over 30 samples.

A. LSB recovery attack using majority voting

First, we analyze how many traces are required to achieve a high accuracy in recovering LSBs. For each secret key, we capture $1,000$ power traces from the device under attack executing the signing algorithm on $1,000$ messages selected at random. The 550-point segments representing the unpacking of eight consecutive coefficients of $\mathbf{t}_0$ are then passed to the ensemble of models $\mathcal{N}_{i \bmod 8}$ to get predictions on the LSBs of $\mathbf{t}_0[i]$, for all $i \in \{0, 1, \ldots, 1023\}$. In this attack, the predictions of the individual models within each ensemble are aggregated using majority voting.

TABLE II
THE NUMBER OF INCORRECTLY RECOVERED LSBS OF $\mathbf{t}_0[i]$, FOR $i \in \{0, 1, \ldots, 1023\}$, IN THE ATTACK USING N TRACES.

Secret key	# Test traces										
	1	2	4	8	16	32	64	128	256	512	1000
1	70	72	66	68	65	63	64	67	68	67	66
2	72	69	63	73	70	72	74	75	71	72	71
3	58	51	45	43	40	42	44	42	45	49	50
4	66	64	68	60	60	58	56	60	58	59	61
5	56	50	49	51	51	51	51	53	53	51	49
6	61	64	61	63	62	61	65	63	63	65	65
7	65	65	62	59	60	57	55	56	58	56	57
8	70	65	67	68	66	65	62	63	62	62	61
9	59	47	52	44	42	42	40	41	40	43	42
10	69	70	66	63	61	63	62	64	64	62	60
Avg.	64.6	61.7	59.9	59.2	57.7	57.4	57.3	58.4	58.2	58.6	58.2

Table II shows the number of incorrectly recovered LSBs in the attacks using N traces. In an N-trace attack, we first combine the predictions of individual models by computing the cumulative probability of the score vectors (representing the probabilities of LSB values) for each trace, and then aggregate the resulting model predictions within the ensemble by majority voting. Recall that we are attempting to recover values that remain constant regardless of the message being signed. It is known that the success rate of profiled side-channel attacks targeting constant values can be improved by utilizing multiple traces [20]. However, as the number of traces increases, the improvement may diminish and eventually flatten due to correlated errors.

From Table II, we can see that using 32 traces for an attack seems to be an optimal trade-off between the number of traces and the success rate. For 64 traces, the average number of errors is only 0.1 lower. Increasing the number of traces further does not lead to better results.

Table III provides additional details of the attacks using 32 traces. We can see that, for the coefficients $\mathbf{t}_0[i]$ with indices (i mod 8) = 3, the LSBs are correctly predicted for all but one secret key. This is because, during unpacking of $\mathbf{t}_0$, only these LSBs are individually extracted from a byte of the input array `a` and assigned to the corresponding coefficient in the output array `r` (line 12 in Fig. 1). For other indices, the LSB is processed along with adjacent bits. For example, for (i mod 8) = 6, the last two bits of a byte from `a` are processed together (line 23 in Fig. 1). The coefficients with indices (i mod 8) = 0 are one of the most difficult case to recover, likely because the value of `a` is directly assigned to `r` without any shift to extract bits (line 2 in Fig. 1).

TABLE III
THE RESULTS OF LSB RECOVERY ATTACK USING 32 TRACES AND MAJORITY VOTING.

Secret key	Total # wrong LSBs	# Wrong LSBs in coefficients $\mathbf{t}_0[i \bmod 8]$							
		0	1	2	3	4	5	6	7
1	63	23	4	9	0	5	4	0	18
2	72	20	7	6	0	2	4	1	32
3	42	9	1	4	0	2	2	2	22
4	58	19	4	8	1	1	0	5	20
5	51	16	4	4	0	3	4	4	16
6	61	17	6	4	0	3	6	1	24
7	57	13	7	7	0	4	5	1	20
8	65	17	5	6	0	3	5	3	26
9	42	11	3	9	0	2	1	1	15
10	63	21	5	9	0	2	7	2	17
Avg.	57.4	16.6	4.6	6.6	0.1	2.7	3.8	2.0	21.0

TABLE IV
THE RESULTS OF LSB RECOVERY ATTACK USING 32 TRACES AND UNANIMOUS VOTING.

Secret key	Total # wrong LSBs	# Wrong LSBs in coefficients $\mathbf{t}_0[i \bmod 8]$							
		0	1	2	3	4	5	6	7
1	25	18	2	3	0	0	1	0	1
2	17	10	0	2	0	0	1	0	4
3	7	1	0	2	0	1	0	0	3
4	20	8	2	4	0	1	0	0	5
5	8	6	0	1	0	0	0	1	0
6	14	7	1	1	0	1	1	0	3
7	13	7	3	0	0	0	1	0	2
8	20	8	1	3	0	0	1	1	6
9	8	4	0	1	0	0	0	0	3
10	21	14	3	2	0	0	1	0	1
Avg.	15.3	8.3	1.2	1.9	0	0.3	0.6	0.2	2.8

B. LSB recovery attack using unanimous voting

We modify the strategy for combining the models' predictions within the ensemble to incorporate error detection. We replace majority voting by *unanimous voting* which takes the decision on the LSB value only if the predictions of all models agree. If there is any disagreement among the models, the sign "?" indicating a possible error in model's predictions is given as an outcome:

$$f(x_1, \ldots, x_n) = \begin{cases} 0, & \text{if } x_i = 0, \forall i \in \{1, \ldots, n\}, \\ 1, & \text{if } x_i = 1, \forall i \in \{1, \ldots, n\}, \\ ?, & \text{otherwise,} \end{cases}$$

where $x_i \in \{0, 1\}$ is the prediction of the ith model and n is the number of models in ensemble, for $n \geq 2$.

Table IV lists the results of LSB recovery attacks using the new aggregation method for $n = 21$. In addition, Table V shows the distribution of detected errors.

From these tables, we can see that, on average, unanimous voting reduces the number of incorrect LSBs from 57.4 to 15.3 and results in 126.3 detected errors. Notably, the error distribution is not uniform. We leverage this to extend the $\mathbf{s}_1$ recovery method of [11] as described next.

TABLE V
THE DISTRIBUTION OF DETECTED ERRORS IN LSB RECOVERY ATTACK USING 32 TRACES AND UNANIMOUS VOTING.

Secret key	# Detected errors	# Detected errors in coefficients $\mathbf{t}_0[i \bmod 8]$							
		0	1	2	3	4	5	6	7
1	139	14	14	16	0	14	14	10	57
2	134	19	17	13	1	8	9	8	59
3	107	20	8	9	0	4	10	4	52
4	131	17	11	13	1	7	18	13	51
5	110	14	6	8	1	13	10	11	47
6	125	18	12	13	0	9	8	12	53
7	133	15	12	16	2	11	13	1	63
8	125	17	13	12	0	11	11	7	54
9	126	15	15	19	0	8	11	6	52
10	133	15	11	20	0	11	15	5	56
Avg.	126.3	16.4	11.9	13.9	0.5	9.6	11.9	7.7	54.4
Occurrence freq.		0.1	0.1	0.1	0	0.1	0.1	0.1	0.4

C. Secret $\mathbf{s}_1$ recovery attack

The side-channel attack method presented in [11] requires the knowledge of the full $\mathbf{t}_0$ vector to recover the $\mathbf{s}_1$ vector. However, as Tables IV and V show, it is not possible to extract the LSBs of all 1024 coefficients of $\mathbf{t}_0$ by side-channel analysis with high probability.[1]. Only for $(i \bmod 8) = 3$, all LSBs of $\mathbf{t}_0[i]$ are recovered without errors. The next best case is $(i \bmod 8) = 6$, for which 80% of LSBs are recovered correctly. Furthermore, there are many undecided LSBs.

To account for LSB values that cannot be recovered with high probability, we extend the method of [11] as follows:

1) For all $i \in \{0, 1, \ldots, 1023\}$ such that $(i \bmod 8) = 3$ or 6:
   - Recover the LSBs of $\mathbf{t}_0[i]$ using the side-channel analysis method described in Section IV-B.
   - Reconstruct the full coefficient $\mathbf{t}_0[i]$ from multiple signatures using the cryptanalytic method of [14].
2) For all coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$, compute score vectors representing the probabilities of coefficient values $\{-2, -1, 0, 1, 2\}$ using the side-channel analysis method of [11].
3) Sort the predicted coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$ in the descending order based on the maximum probability of their score vectors. Let L be the resulted sorted list.
4) For all $i \in \{0, 1, \ldots, 1023\}$ such that either the LSBs of $\mathbf{t}_0[i]$ is marked as "?", or $(i \bmod 8) \neq 3$ or 6:
   - Remove the coefficient $\mathbf{s}_2[i]$ from L.
   - Remove the ith equation from the system of linear equations $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$.
5) Substitute 1024 top elements of L in the reduced system of linear equations and solve the equations to recover $\mathbf{s}_1$.

The side-channel attack described in [11] can recover a half of 2048 coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$ from $1,000$ traces with a probability close to one. Next, we assess the probability of

[1]We can extract all LSBs if the same device is used for both profiling and attack, however, this is not a realistic scenario.

TABLE VI
EMPIRICAL PROBABILITY TO RECOVER 80.5% OF $\mathbf{s}_1$ AND $\mathbf{s}_2$ COEFFICIENTS FROM 960 TRACES.

Test	1	2	3	4	5	6	7	8	9	10	Avg.
Sucess rate	0.86	0.87	0.88	0.87	0.89	0.87	0.87	0.89	0.88	0.87	0.875

recovering a larger fraction of the coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$ by side-channel analysis.

In the extended $\mathbf{s}_1$ recovery method, to get a unique solution, we need to recover 80.5% of the coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$ by side-channel analysis since the reduced system of equations consists of 248 equations with 1,272 unknowns on average (in step 4, the total number of removed equations is equal to 6×128 plus the number of detected errors in columns of Table V with indices 3 and 6).

The results of 10 tests summarized in Table VI show that 80.5% of the coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$ can be recovered with an average success rate of 0.875 from 960 traces. We carried out these tests using the traces and neural networks from the experiments of [11]. Undecided LSB positions are sampled based on the distribution shown in the last row of Table V. Clearly, in a real attack, the same set of traces can be reused to recover both $\mathbf{t}_0$ and $\mathbf{s}_1$. Since we use 32 traces averaged 30 times in the $\mathbf{t}_0$ LSB recovery attack in Section IV-B, so the total number of traces used in both attacks is the same, $32 \times 30 = 960$.

Since the average probability of recovering all LSBs of $\mathbf{t}_0[i]$ for $(i \bmod 8) = 3$ and 6 is 0.8, and the average probability of recovering 80.5% of the coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$ is 0.875, the overall average probability of reconstructing the full $\mathbf{s}_1$ from 960 traces is $0.875 \times 0.8 = 0.7$.

V. CONCLUSION

The main contribution of this paper is new deep learning-assisted power analysis method for the partial reconstruction of $\mathbf{t_0}$. This method enables the recovery of the $\mathbf{s}_1$ component of Dilithium's secret key with a 0.7 success probability using fewer than $1,000$ power traces and approximately one million signatures. This is a significant improvement over the prior side-channel attack [11] relying on the $\mathbf{t}_0$ reconstruction method which requires two million signatures [14]. The time for capturing $1,000$ power traces is at least $1,000$ times shorter than the time for generating one million signatures, thereby proportionally reducing access time to the device under attack.

VI. ACKNOWLEDGMENTS

This work was supported in part by the Swedish Civil Contingencies Agency (Grant No. 2020-11632) and the KTH Center for Cyber Defense and Information Security (CDIS). The authors are also grateful to Sönke Jendral for his comments and suggestions which helped improve the paper.

REFERENCES

[1] NIST, "Announcing approval of three federal information processing standards (FIPS) for post-quantum cryptography," NIST Federal Register Notice 2024-17956, 13 August 2024. [Online]. Available: https://csrc.nist.gov/publications/fips

[2] J. P. Mattsson, E. Thormarker, and B. Smeets, "Migration to quantum-resistant algorithms in mobile networks." [Online]. Available: https://www.ericsson.com/en/blog/2023/2/quantum-resistant-algorithms-mobile-networks

[3] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "PQM4: Testing and benchmarking NIST PQC on ARM Cortex-M4," 2019.

[4] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium algorithm specifications and supporting documentation," 2021, https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.

[5] P. Ravi, M. P. Jhanwar, J. Howe, A. Chattopadhyay, and S. Bhasin, "Side-channel assisted existential forgery attack on Dilithium - a NIST PQC candidate," Cryptology ePrint Archive, Paper 2018/821, 2018, https://eprint.iacr.org/2018/821.

[6] J. Han, T. Lee, J. Kwon, J. Lee, I.-J. Kim, J. Cho, D.-G. Han, and B.-Y. Sim, "Single-trace attack on NIST round 3 candidate Dilithium using machine learning-based profiling," *IEEE Access*, vol. 9, pp. 166 283–166 292, 2021.

[7] Z. Chen, E. Karabulut, A. Aysu, Y. Ma, and J. Jing, "An efficient non-profiled side-channel attack on the CRYSTALS-Dilithium post-quantum signature," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, 2021, pp. 583–590.

[8] V. Q. Ulitzsch, S. Marzougui, M. Tibouchi, and J.-P. Seifert, "Profiling side-channel attacks on Dilithium: A small bit-fiddling leak breaks it all," in *International Conference on Selected Areas in Cryptography*. Springer, 2022, pp. 3–32.

[9] A. Berzati, A. C. Viera, M. Chartouny, S. Madec, D. Vergnaud, and D. Vigilant, "Exploiting intermediate value leakage in Dilithium: a template-based approach," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 4, pp. 188–210, 2023.

[10] O. Bronchain, M. Azouaoui, M. ElGhamrawy, J. Renes, and T. Schneider, "Exploiting small-norm polynomial multiplication with physical attacks: Application to CRYSTALS-Dilithium," *Cryptology ePrint Archive*, 2023.

[11] R. Wang, K. Ngo, J. Gärtner, and E. Dubrova, "Unpacking needs protection - A single-trace secret key recovery attack on Dilithium," *IACR Communications in Cryptology*, vol. 1, no. 3, 2024.

[12] L. G. Bruinderink and P. Pessl, "Differential fault attacks on deterministic lattice signatures," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2018, no. 3, pp. 21–43, 2018.

[13] S. Jendral, J. P. Mattsson, and E. Dubrova, "A single-trace fault injection attack on hedged module lattice digital signature algorithm (ML-DSA)," in *2024 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*, 2024, pp. 34–43.

[14] P. A. Oliveira, A. C. Viera, B. Cogliati, and L. Goubin, "Uncompressing Dilithium's public key," *Cryptology ePrint Archive*, 2024, https://eprint.iacr.org/2024/1373.

[15] A. Abdulrahman, V. Hwang, M. J. Kannwischer, and A. Sprenkels, "Faster Kyber and Dilithium on the Cortex-M4," in *International Conference on Applied Cryptography and Network Security*. Springer, 2022, pp. 853–871.

[16] B. L. Welch, "The generalization of 'Student's' problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.

[17] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[18] B. P. Lathi, *Signal Processing and Linear Systems*. Berkeley-Cambridge Press, 1998.

[19] H. Wang and E. Dubrova, "Tandem deep learning side-channel attack against FPGA implementation of AES," in *2020 IEEE International Symposium on Smart Electronic Systems (iSES)*, 2020, pp. 147–150.

[20] R. Wang, K. Ngo, and E. Dubrova, "Side-channel analysis of Saber KEM using amplitude-modulated EM emanations," in *2022 25th Euromicro Conference on Digital System Design*. IEEE, 2022, pp. 488–495.