A Novel Power Analysis Attack against CRYSTALS-Dilithium Implementation

Yong Liu*, Yuejun Liu*, Yongbin Zhou*†‡, Yiwen Gao*, Zehua Qiao†‡, and Huaxin Wang*
*School of Cyber Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
†Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
‡School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{fengzhongren, liuyuejun, zhouyongbin, gaoyiwen, wanghuaxin}@njust.edu.cn
qiaozehua@iie.ac.cn

***Abstract*—Post-Quantum Cryptography (PQC) was proposed due to the potential threats quantum computer attacks against conventional public key cryptosystems, and four PQC algorithms besides CRYSTALS-Dilithium (Dilithium for short) have so far been selected for NIST standardization. However, the selected algorithms are still vulnerable to side-channel attacks in practice, and their physical security need to be further evaluated. This study introduces two efficient power analysis attacks, the optimized fast two-stage approach and the single-bit approach, aimed at reducing the key guess space in NTT polynomial multiplication on an STM32F405 device (ARM Cortex-M4 core). Our findings reveal that the optimized approach outperforms the conservative approach and the fast two-stage approach proposed in ICCD 2021 by factors of 519 and 88, respectively. Similarly, the single-bit approach demonstrates speedups of 365 and 62 times compared to these two approaches, respectively.**

***Index Terms*—Side-Channel Attack, CRYSTALS-Dilithium, Post-Quantum Cryptography, Montgomery Reduction, Number Theoretic Transform**

I. Introduction

With Shor's algorithms [1], quantum computers are able to solve the Integer Factorization Problem (IFP) and Discrete Logarithm Problem (DLP) in polynomial time, making the conventional public key cryptosystems like RSA, DSA not secure any more. For this, Post-Quantum Cryptogrpahy (PQC) was proposed in the past decades, and National Institute of Standards and Technology (NIST) has been calling for PQC algorithms since 2016. Until now, four PQC algorithms, i.e. CRYSTALS-Kyber, CRYSTALS-Dilithium (Dilithium for short), Falcon and SPHINCS+ have been selected for NIST standardization. In this work, we focus on Dilithium, which is the primary signature standard.

The PQC algorithms are generally considered resistant to quantum computer attacks, but they remain vulnerable to side-channel attacks when deployed on computing devices [2]. Migliore *et al.* conducted a side-channel evaluation of Dilithium on an ARM Cortex-M3 device using Welch's t-test, revealing observable leakage [3]. Similarly, Fournaris *et al.* investigated Dilithium's leakage on an ARM Cortex-M4 device, identifying potential attack locations [4]. While both studies indicated the feasibility of side-channel attacks, neither conductud practical attacks. For practical attacks, there are profiled attacks [5] and non-profiled attacks. Berzati *et al.* proposed a new profiled side-channel attack on the Dilithium signature scheme [6], demonstrating an efficient key-recovery attack exploiting leakage in the vector $\mathbf{w_0}$ through experiments on ARM Cortex-M4. Although this leakage is not directly related to the private key, it enables partial key recovery using mathematical methods. However, profiled attacks, despite their effectiveness, have limitations such as the significant time required to build a template and the need for an attacker to control a device very similar to the target [7], [8], [9], [10].

In contrast, non-profiled attacks such as differential power analysis (DPA) [11], and correlation power analysis (CPA) [12] do not need to consider these factors. Steffen *et al.* proposed an FPGA-based attack targeting NTT polynomial multiplication [13]. Due to the parallel operation of many modules on FPGA and relatively high noise levels, a large number of power traces are needed to recover the private key, resulting in high attack costs. Chen *et al.* recovered the private key using both a conservative approach and a fast two-stage approach in their practical attacks on ARM Cortex-M4 [14]. Initially, they successfully recovered the private key using the conservative approach, which involved reducing the guess space based on the features of the private key in the NTT domain. Although this approach can recover up to 99.99% of the private key coefficients, the wide range of these coefficients in the NTT domain makes the attack costly. Subsequently, they introduced the fast two-stage approach to expedite the attack. While this approach does reduce the attack cost, recovering the entire private key is still time-consuming due to the large number of coefficients involved. Therefore, a more efficient approach is necessary.

In this study, we have enhanced the fast two-stage approach originally proposed by Chen *et al.* [14], resulting in our optimized version running 88 times faster. Additionally, we introduce a single-bit approach tailored for polynomial multiplication in the NTT domain. This approach significantly

This work is supported in part by National Key R&D Program of China (No.2022YFB3103800), Defense Industrial Technology Development Program (No.JCKY2021606B013), National Natural Science Foundation of China (No.U1936209, No.62202231 and No.62202230), China Postdoctoral Science Foundation (No.2021M701726 and No.2023M741709), Jiangsu Funding Program for Excellent Postdoctoral Talent (No.2023ZB031), Yunnan Province Science and Technology Major Project (NO.202302AD080002) and CCF-Tencent Rhino-Bird Open Research Fund (No.CCF-Tencent RAGR20230114).

reduces the guess space for each private key coefficient from 2^{27} to 703, greatly improving the efficiency of the attack. Moreover, we have successfully resolved the challenge of key recovery in the single-bit approach when the least significant bit of the key is '0'.

The rest of this paper is structured as follows. In Section II, we present the preliminaries for this paper. In Section III, we present our power analysis attacks to a Dilithium implementation. In Section IV, we present experimental results and discussions. Finally, conclusion is given in Section V.

II. Background

A. Notations

In Dilithium, matrices are represented with bold capital letters, such as $\mathbf{A}$, and vectors with bold lowercase letters, like $\mathbf{s}$. The components of these matrices and vectors typically belong to the polynomial ring $R_q = \mathbb{Z}_q[X]/(X^n+1)$. The notation $\rho(f(A,B), Traces)$ represents the Pearson correlation coefficient (PCC), calculated between the intermediate value $f(A,B)$ and the corresponding power traces. Additionally, the Most Significant p-Bit of A is indicated as $A_{\mathrm{MSB}(p)}$, and the Least Significant p-Bit as $A_{\mathrm{LSB}(p)}$. Finally, the p-th bit from the end of A is represented as A_p.

B. CRYSTALS-Dilithium

Dilithium, selected in 2022 as one of the four PQC algorithms for standardization by the NIST, is a lattice-based signature algorithm. Its security is based on the hardness of the Module Learning With Errors (Module-LWE) and Module Short Integer Solution (Module-SIS) problems. Designed over the fixed base ring $R_q = \mathbb{Z}_q[x]/(x^{256}+1)$, where $q = 8380417$, it offers flexibility by supporting different module parameters (k, ℓ).

Algorithm 1 Dilithium.Gen()

```
$\zeta \leftarrow \{0,1\}^{256}$
$(\rho, \varsigma, K) \in \{0,1\}^{256\times 3} := H(\varsigma)$
$(\mathbf{s_1}, \mathbf{s_2}) \in S_\eta^t \times S_\eta^k := H(\varsigma)$
$\mathbf{A} \in R_q^{k\times \ell} := \mathbf{ExpandA}(\rho)$
$\mathbf{t} := \mathbf{As_1} + \mathbf{s_2}$
$(\mathbf{t_1}, \mathbf{t_0}) := \mathbf{Power2Round_q}(\mathbf{t}, d)$
$tr \in \{0,1\}^{384} := \mathbf{CRH}(\rho \parallel \mathbf{t_1})$
return $pk = (\rho, \mathbf{t_1}), sk = (\rho, K, tr, \mathbf{s_1}, \mathbf{s_2}, \mathbf{t_0})$
```

Key Generation: The key generation algorithm is presented in Alg. 1. First, the algorithm samples the random private key vector $\mathbf{s_1}$ and $\mathbf{s_2}$. Each coefficient of $\mathbf{s_1}$ and $\mathbf{s_2}$ is an element of R_q with small coefficients of size at most η. Secondly, the algorithm a $k \times \ell$ matrix $\mathbf{A}$ each of whose entries is a polynomial in the ring R_q. Finally, a part of the public key and private key is computed as $\mathbf{t} = \mathbf{As}_1 + \mathbf{s_2}$. As previously mentioned, we will always have $q = 2^{23} - 2^{13} + 1$, $N = 256$ and $R_q = \mathbb{Z}_q[X]/(X^n+1)$. During key generation, all algebraic operations are assumed to be performed in the polynomial ring R_q.

Algorithm 2 Dilithium.Signature()

```
$\mathbf{A} \in R_q^{k\times l} := \mathbf{ExpandA}(\rho)$
$\mu \in \{0,1\}^{384} := \mathbf{CRH}(tr \| M)$
$\kappa := 0, (\mathbf{z}, \mathbf{h}) := \perp$
$\rho' \in \{0,1\}^{384} := \mathbf{CRH}(K \parallel \mu) \pmod{\rho'} \leftarrow \{0,1\}^{384}$
while $\mathbf{do}(\mathbf{z}, \mathbf{h}) := \perp$
    $\mathbf{y} \in \widetilde{S}_{\gamma_1}^l := \mathbf{ExpandMassk}(\rho', \kappa)$
    $\mathbf{w} := \mathbf{Ay}$
    $\mathbf{w_1} := \mathbf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
    $\mathbf{c} \in B_\tau := \mathbf{H}(\mu \parallel \mathbf{w_1})$
    $\mathbf{z} := \mathbf{y} + \mathbf{cs_1}$
    $\mathbf{r_0} := \mathbf{LowBits_q}(\mathbf{w} - \mathbf{cs_2}, 2\gamma_2)$
    if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathbf{r_0}\|_\infty \geq \gamma_2 - \beta$ then $(\mathbf{z}, \mathbf{h}) := \perp$
    else
        $\mathbf{v_t} := \mathbf{INTT}(\hat{\mathbf{c}} \cdot \hat{\mathbf{t}}_0)$
        $\mathbf{h} := \mathbf{MakeHint_q}(-\mathbf{v_t}, \mathbf{w} - \mathbf{v_s} + \mathbf{v_t}, 2\gamma_2)$
        if $\|\mathbf{v_t}\|_\infty \geq \gamma_2$ or $[\mathbf{h}] > w$ then $(\mathbf{z}, \mathbf{h}) := \perp$
        end if
    end if
    $\kappa := \kappa + \ell$
end while
return $\rho = (\tilde{\mathbf{c}}, \mathbf{z}, \mathbf{h})$
```

Signature Generation: The signature generation algorithm is presented in Alg. 2. First, the random polynomial $\mathbf{y}$ is sampled by the function $ExpandMask$. Then calculate $\mathbf{w} = \mathbf{Ay}$. After obtaining challenge $\mathbf{c}$, calculate $\mathbf{z} = \mathbf{cs_1} + \mathbf{y}$. If $\mathbf{z}$ were directly output at this point, then the signature scheme would be insecure due to the fact that the private key would be leaked. Rejection sampling is used to avoid $\mathbf{z}$'s dependence on the private key. If any coefficient of $\mathbf{z}$ is larger than $\gamma_1 - \beta$, then reject and restart the signing procedure. The while loop continues throughout signature generation until the previous condition is satisfied. The different security levels of Dilithium are given in Table I. In this paper, we focus on Dilithium at Security Level 2, i.e. For Dilithium at other security levels, our approaches work as well.

TABLE I: Parameters of Dilithium.

NIST Security Level	2	3	5
Parameters			
q [modulus]	8380417	8380417	8380417
d [dropped bits from $\mathbf{t}$]	13	13	13
τ [# of ± 1's in c]	39	49	60
γ_1 [$\mathbf{y}$ coefficient range]	2^{17}	2^{19}	2^{19}
γ_2 [low-order rounding range]	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
(k, ℓ) [dimensions of $\mathbf{A}$]	(4,4)	(6,5)	(8,7)
η [private key range]	2	4	2
β [$\tau \cdot \eta$]	78	196	120

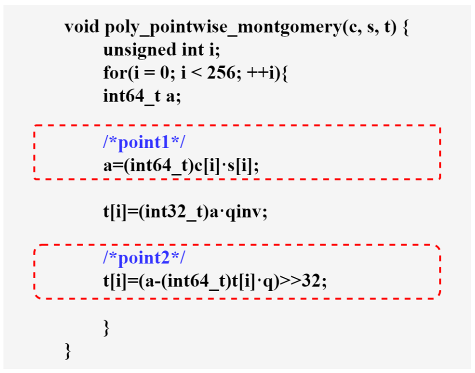
C. Partial Correlation

Tunstall *et al.* points out that partial correlation exists if the correlation coefficient of n independent bits amongst m is calculated [15]. This is given as:

$$\rho(f(A,B), Traces)_{n/m} = \rho(f(A,B), Traces)\sqrt{\frac{n}{m}} \quad (1)$$

where n bits from an m-bit word are known.

This concept introduces a novel attack strategy. An attacker can initially compromise a portion of the key's bits and then use this information to attack the remaining bits. This process is iterative, with each phase of the attack building upon the knowledge gained from the previous one, until the entire key is successfully recovered.



```
void poly_pointwise_montgomery(c, s, t) {
        unsigned int i;
        for(i = 0; i < 256; ++i){
        int64_t a;

        /*point1*/
        a=(int64_t)c[i]·s[i];

        t[i]=(int32_t)a·qinv;

        /*point2*/
        t[i]=(a-(int64_t)t[i]·q)>>32;

        }
}
```

Fig. 1: Polynomial multiplication at Dilithium.

III. A Novel Power Analysis Attack

A. Locating the Vulnerability in Dilithium

To identify potential vulnerabilities, we focus on operations involving the private key. As indicated in Alg. 2, $\mathbf{cs_1}$, $\mathbf{ct_0}$, and $\mathbf{cs_2}$ are identified as potential weak points. Given the similarity in their calculation processes and private key structures, our analysis will concentrate on $\mathbf{cs_1}$. In the Dilithium2, $\mathbf{s_1}$ is a vector of $\ell = 4$ polynomials, each consisting of $N = 256$ coefficients. The challenge $\mathbf{c}$ is also a polynomial with $N = 256$ coefficients. Each polynomial of $\mathbf{s_1}$ performs operations with challenge $\mathbf{c}$ separately. It can be seen that we can perform an attack on each polynomial of $\mathbf{s_1}$ separately, and then obtain $\mathbf{s_1}$. As depicted in Fig. 1, we illustrate the operational process between a single polynomial of $\mathbf{s_1}$ and the challenge $\mathbf{c}$. Here, $c[i]$ and $s[i]$ represent the i-th coefficients of $\mathbf{c}$ and the private key polynomials, respectively. The sequential nature of operations involving $c[i]$ and $s[i]$ allows for a step-by-step recovery of each coefficient, significantly reducing the complexity of the attack. It's important to note that these operations are conducted in the NTT domain, and constants such as q and $qniv$ are involved. With $\mathbf{c}$ being known, calculating the intermediate values of $c[i]$ and $s[i]$ becomes feasible, paving the way for the attack. As shown in Fig. 1, the attack positions that can be exploited are point1 and point2, which have been confirmed by Chen *et al.* [14].

B. CPA on a single bit

Inspired by the methodology proposed by Tunstall *et al.* [15], we can segment the key and approach it with sequential attacks. Consider, for instance, the multiplication of two 32-bit values in a 32-bit ALU, where Y is known, and X is the key. The attack is conducted using CPA and utilizes the Hamming weight model. Let the 32-bit key X be divided into four bytes X_{MSB}, X_{MMSB}, X_{MLSB}, and X_{LSB}. An attacker can perform a CPA attack on each byte in turn. The attack begins with a CPA on X_{LSB}, given that there are only 256 (2^8) possible values for this segment. As discussed in Section II-C,this will produce a correlation coefficient of $\sqrt{\frac{1}{4}}$ the correlation produced if all the bits were predicted. Once X_{LSB} is compromised, the attack proceeds to the next 8 bits. At this stage, the effort remains similar, as the previously acquired 8 bits reduce the complexity. This will produce a correlation coefficient of $\sqrt{\frac{1}{2}}$ the correlation produced if all the bits were predicted. Following this pattern, the attack can extend to X_{MMSB} and X_{MSB}, with each successive stage building upon the data acquired from the previous. This segmented approach is substantially more efficient than a direct CPA attack on the entire 32-bit key X. By dividing the key into segments and attacking sequentially, the guess space effectively reduces from 2^{32} to $2^8 \cdot 4$, significantly lowering the attack's complexity.

When the key X is segmented into 32 individual parts, with each segment comprising a single bit, the CPA attack can be executed on each bit independently. For instance, when targeting the least significant bit $X_{\text{LSB}(1)}$, the guess space for the attacker is only 2^1, encompassing the two binary possibilities, '1' and '0'. This approach, which begins with targeting $X_{\text{LSB}(1)}$ and then progressively attacks the remaining bits, proved to be more effective than directly performing a CPA attack on the entire 32-bit key X due to its reduced computational complexity and more focused guess space.

C. Optimized Fast Two-stage Approach

Chen *et al.* introduced two methodologies: a conservative approach and a fast two-stage approach [14]. The latter notably reduces the attack running time compared to the former. Nevertheless, it is essential to consider the large number of coefficients in $\mathbf{s_1}$, which contributes to the time-consuming nature of the attack process. This section aims to further optimize the fast two-stage approach, enhancing its efficiency and reducing the overall attack running time. Here we review the procedure of the fast two-stage approach:

- **Stage1:** In Fig. 1, an attacker first performs CPA on the LSB-p of the $s[i]$ at point1. The purpose of this stage is to select a few highly correlated LSB-p candidates to eliminate erroneous key guesses.
- **Stage2:** In this stage, an attacker utilizes the candidates identified in Stage1 to create a preliminary guess list for $s[i]_{\text{LSB}(p)}$. Next, the attacker carries out a CPA targeting point2, which aims to recover the complete secret coefficient $s[i]$. This method builds upon the initial information gleaned in Stage1, allowing for a more focused and effective CPA in the subsequent stage.

In the second stage of the fast two-stage approach, the guess space for each candidate value is 2^{27-p}, and the total guess space is $(2^{27-p}) \cdot \text{n}_{cdd}$ (the guess space for $\mathbf{s_1}$ in the NTT domain is close to 2^{27}, where n_{cdd} denotes the number of candidate values). When p is small, the execution of the CPA attack in Stage2 becomes more time-consuming. $s[i]$ and $c[i]$ perform multiplication operations at point1. Therefore, after obtaining $s[i]_{\text{LSB}(p)}$, the remaining $(27-p)$ bits can be split into individual bits, and then a CPA attack can be performed on each bit separately. The entire attack flow is as follows:

- **Stage1:** In this stage, we reproduce the first stage of the fast two-stage approach.
- **Stage2:** First,an attacker utilizes the candidates identified in Stage1 to create a preliminary guess list for $s[i]_{\text{LSB}(p)}$. For each candidate within the guess list, a sequential CPA attack is conducted on the remaining $(27-p)$ bits. Taking a candidate, denoted as A, as an example: the process begins with a CPA attack on the bit A_{p+1}, followed by an attack on A_{p+2}, and so on. This approach continues iteratively, with a CPA attack on each subsequent bit, concluding with A_{27}. This method is replicated for each candidate in the list, thereby generating a new, refined set of guesses. Finally, the PCC of each candidate in this updated guess list is calculated and the candidate with the highest PCC value is selected as the most probable value for $s[i]$.

The primary distinction between the fast two-stage approach and its optimized variant lies in the execution of the second stage. In the optimized approach, this stage entails splitting the bits and conducting a CPA attack on each bit individually. This strategy effectively reduces the guess space in the second stage from $2^{27-p} \cdot \text{n}_{cdd}$ to $2 \cdot (27-p) \cdot \text{n}_{cdd}$. It is important to note that when the value of p is large, the efficiency gains of the optimized fast two-stage approach compared to the original approach become less pronounced. However, higher values of p typically increase the attack overhead in the first stage, which is generally undesirable. Consequently, the optimized approach can lead to a significant reduction in overall attack running time, especially when p is small.

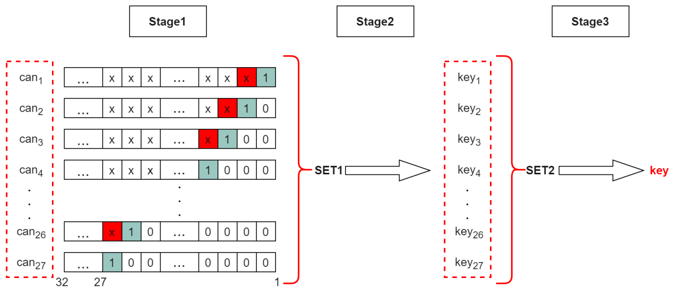


Fig. 2: The process of single-bit approach.

D. Single-Bit Approach

The fast two-stage approach significantly reduces the attack running time but is constrained by certain limitations. As a probabilistic approach, its success largely depends on the outcomes of the first stage. This dependency also extends to the optimized fast two-stage approach, which fails to recover $s[i]$ if the first stage is unsuccessful. As depicted in Fig. 1, point1 involves a direct multiplication operation between $s[i]$ and $c[i]$. This interaction is prone to generating false positives. For example, $s[i]$ is hex(0000FF00), and the key guess is hex(00007F80). The power model is challenging to differentiate because, after multiplying by the same $c[i]$, they consistently exhibit similar Hamming weights. Although Chen *et al.* proposed a hybrid approach to address the shortcomings of the fast two-stage approach, the attack running time is still extended due to the large number of coefficients in $\mathbf{s_1}$. In this subsection, we explore the acquisition of $s[i]$ using a single-bit approach.

In the optimized fast two-stage approach, the CPA attack is executed on each bit of $s[i]$ individually, starting from $s[i]_{p+1}$. This follows after obtaining the p least significant bits of $s[i]$ in Stage1. Conversely, in the single-bit approach, the CPA attack commences from $s[i]_1$, targeting each bit separately. However, in single-bit approach, when $s[i]_{\text{LSB}(p)}$ is '0', the CPA attack cannot be performed on $s[i]_1$. Details of the issue are as follows. $s[i]_1$ has only two possible bit guesses, '1' and '0'. When $s[i]_{\text{LSB}(p)}$ is '0', $HW((s[i] \cdot c[i])_{\text{LSB}(1)})$ is '0'. This renders the power model indistinguishable between the correct bit and the incorrect bit. Thus, initiating a CPA attack directly from $s[i]_1$ becomes unfeasible in the single-bit approach. The optimized fast two-stage approach is able to perform the CPA attack on each bit starting from $s[i]_{p+1}$ in the second stage by selecting several LSB-p with high correlation to avoid this problem. We'll discuss how to solve this problem next. To ensure that the $s[i]$ can be successfully recovered by single-bit approach. This means that $s[i]_{\text{LSB}(p)}$ must not be '0' when the CPA is executed from $s[i]_{p+1}$. Therefore, as long as at least one of the p bits of $s[i]_{\text{LSB}(p)}$ is '1', the above problem can be solved. Then we can quickly recover the $s[i]$ by single-bit approach. For example, when $s[i]_{\text{LSB}(p)}$ is '0' and $s[i]_{p+1}$ is '1', then $s[i]_{\text{LSB}(p+1)}$ is not '0'. Therefore, we can perform CPA starting from $s[i]_{p+2}$, and we can recover the correct $s[i]$. However, this approach faces challenges when $s[i]_{\text{LSB}(p+1)}$ is '0', as initiating CPA from $s[i]_{p+2}$ would still encounter the same issue. In order to recover the $s[i]$ successfully, as shown in Fig. 2, we will consider all possible cases where $s[i]_{\text{LSB}(p)}$ is '0'. Make it form a set SET1, perform an attack on each candidate value of SET1, and then combine it with point2 to recover $s[i]$ (excluding the case where $s[i]$ is entirely '0'). Therefore, as shown in Fig. 2, the attack process of the single-bit approach is as follows:

- **Stage1:** The creation of the candidate set SET1 is a critical step, involving the consideration of all scenarios where $s[i]_{\text{LSB}(p)}$ is '0'. The size of SET1 is contingent upon the range of $s[i]$, with each candidate in SET1 representing a unique case where $s[i]_{\text{LSB}(p)}$ is '0'. This approach aims to ensure a high success rate for the attack, as the attacker lacks prior knowledge of the current $s[i]_{\text{LSB}(p)}$ value during an actual attack scenario. For example, as shown in Fig. 2, when $s[i]_{\text{LSB}(1)}$ is '0' and $s[i]_2$ is '1', it is only necessary to perform the CPA from $s[i]_3$ to recover the $s[i]$. If the attack goes well, $can1$ is the correct private key coefficient. However, it is unknown for an attacker what the exact value of $s[i]_{\text{LSB}(p)}$ is and from where the CPA is executed. Consequently, while attacking most candidate values in SET1 may seem redundant, it is necessary to consider all cases where $s[i]_{\text{LSB}(p)}$ is '0' to accurately recover each coefficient of $\mathbf{s_1}$.

- **Stage2:** At this stage, the focus is on conducting an attack on each candidate value within SET1, leading to the formation of a new set, SET2. Following the methodology outlined in Section III-B,as shown in Fig. 2,the attack begins from a designated starting point (marked by a red square) for each candidate in SET1, targeting individual bits in succession. Given that each bit is hypothesized as either '1' or '0', the attack process is relatively rapid. In a scenario where the attack is executed successfully, the correct $s[i]$ is guaranteed to be a member of SET2. This assurance stems from the comprehensive consideration of all instances where $s[i]_{\text{LSB}(p)}$ is '0', thus encompassing every potential outcome.
- **Stage3**: In this stage, the attacker employs point2 to sift through SET2, effectively discarding incorrect candidate values to isolate the correct $s[i]$. The presence of numerous incorrect candidates within SET2 is an anticipated outcome, attributable to the exhaustive nature of the attack strategy that considers all possible scenarios from the outset.

The overall running time of the attack is the cumulative sum of the running time for attacking each candidate in SET1. The number of guesses for each candidate is $(27-p+1)\cdot 2$ (For each candidate of SET1, p is initially the position of the red square). Therefore, for the single-bit approach, the total guess space for attacking a coefficient is $((27-p+1)\cdot 2)\cdot 27 = 703$. It is important to note that not every coefficient of $\mathbf{s_1}$ necessarily encounters the scenario where LSB-p is '0'. Ignoring this situation theoretically allows for the recovery of $s[i]$ within a guess space of $27 \cdot 2$. However, this approach would only recover approximately half of the coefficients of $\mathbf{s_1}$ due to its uniform distribution. To achieve a high success rate, the attack strategy considers the worst-case scenario for each $s[i]$, which, while increasing the computational cost, ensures a success rate of 99.99%. Despite this increased cost, the attack running time is still significantly lower than that of the conservative approach or the fast two-stage approach.

IV. Experimental Results

A. Setup

In our practical attack, we focus on Dilithium, an open-source reference implementation submitted for consideration in the NIST's 3rd round PQC Competition. The subsequent discussion outlines our experimental setup. The primary target device in our experiment is a Chipwhisperer UFO development board, which is powered by a standard power supply. This board features an STM32F405 chip, executing the Dilithium implementation code. The chip operates at 168Hz, utilizing an ARM Cortex-M4 core. For power trace acquisition, we employed a PicoScope 3206D oscilloscope. This device is capable of sampling at a rate of 125 million samples per second and offers a bandwidth of 200MHz. We connected two probes to the target device's pins: one for data collection and the other serving as a trigger mechanism. The gathered data was transferred to a desktop computer via a USB data cable. The desktop is equipped with an Intel(R) Core(TM) i7-9700 CPU, 32GB of RAM, and runs on the Windows 11 operating system. The CPA programme runs on Matlab 2018b.

TABLE II: A performance comparison between the fast two-stage approach and its optimized variant.

Approach	#Traces	n_{cdd}	p	Success Rate[1]	Running Time[2](s)
Fast Two-Stage	3,000	40	14	99.99%	965.61
	5,000	24	13	99.61%	1135.03
	10,000	12	12	97.26%	1137
Optimized Fast Two-Stage	3,000	40	14	98.04%	19.83
	5,000	24	13	98.83%	14.99
	10,000	**12**	**12**	**96.87%**	**12.96**

[1]Success Rate: The success rate of a polynomial of the attack $\mathbf{s}_1$.
[2]Running Time: The time for recovering one coefficient (in second).
Note: The running times in the table are measured in the same experimental setup.

TABLE III: Experimental results of the single-bit approach.

Approach	Scheme	#Traces	Success Rate[1]
Single-Bit	Dilithium	1,500	98.04%
		2,000	99.61%
		3000	**99.99%**
		5,000	99.99%

[1]Success Rate: The success rate of a polynomial of the attack $\mathbf{s}_1$.

B. Performance of our Optimized Fast Two-stage Approach

In this section, we present a comparative analysis between the fast two-stage approach and its optimized variant. The fast two-stage approach was replicated under the identical experimental setup for a fair comparison. To ensure the validity of our experimental results, we adhered to the experimental parameter settings suggested by Chen *et al.* [14], where n_{cdd} represents the number of candidate values selected, and p indicates the number of bits targeted in the first stage. The experimental results are shown in Table II, the parameter set $\{Traces, \text{n}_{cdd}, p\} = \{10000, 12, 12\}$ shows the best acceleration effect. The optimized approach achieves a speed advantage of about 88 times compared to the fast two-stage approach without incurring any additional overhead.

C. Performance of our Single-Bit Approach

We conducted an actual attack on Dilithium2 by single-bit approach. The results of this attack are detailed in Table III. Our findings confirm the effectiveness of the single-bit approach in recovering $\mathbf{s_1}$, which is involved in NTT polynomial multiplication. The success rate of the attack was evaluated by analyzing the coefficients of a polynomial, each comprising 256 coefficients. As indicated in Table III, a success rate of 98.04% was achieved using only 1,500 power traces. Increasing the number of power traces to 3,000 enabled the recovery of all coefficients.

D. Discussions

In this section, we engage in a detailed performance comparison between our proposed approaches and those presented by Chen *et al.* [14]. It is important to emphasize that the

TABLE IV: Performance comparison of our approaches and the approaches proposed in ICCD 2021

Approach	#Traces	Success Rate[1]	Running Time[2]
Conservative [14]	157	99.99%	477
Fast Two-Stage [14]	10,000	97.26%	81
Optimized Fast Two-Stage	**10,000**	**96.84%**	**0.92**
Single-Bit	**3,000**	**99.99%**	**1.30**

[1]Success Rate: The success rate of a polynomial of the attack $\mathbf{s}_1$.
[2]Running Time: The running time of a polynomial of the attack $\mathbf{s}_1$ (in hour).
Note: The running times are measured in the same experimental setup.

experimental results showcased in Table IV for all approaches were obtained under identical experimental conditions.

The experimental outcomes, as summarized in Table IV, reveal that both the optimized fast two-stage approach and the single-bit approach significantly reduce the attack running time in comparison to the conservative and fast two-stage approaches, without incurring any additional overhead. Specifically, the optimized fast two-stage approach achieves an acceleration of 519 times and 88 times over the conservative and fast two-stage approaches, respectively. Similarly, the single-bit approach shows an acceleration of 365 times and 62 times when compared to the conservative and fast two-stage approaches, respectively. Notably, our approach is not limited to $\mathbf{s_1}$. It also successfully recovers $\mathbf{s_2}$ and $\mathbf{t}_0$, and is applicable to other security levels, including Dilithium3 and Dilithium5. This suggests the potential to recover the private key for these additional security levels using a similar number of power traces.

It is noteworthy that the single-bit approach exhibits a higher success rate than the optimized fast two-stage approach, albeit at the cost of a longer attack running time. A practical strategy for attackers could involve initially employing the optimized fast two-stage approach to target the private key coefficient. If this approach proves unsuccessful, the single-bit approach can then be utilized to complete the attack. The decision to switch approaches can be based on a predetermined threshold, which serves as an indicator of the success of the optimized fast two-stage approach. The methodology for establishing this threshold has been previously proposed and implemented in actual attacks by Chen *et al.* [14]. Thus, it is not the primary focus of this paper. This combined strategy enables a rapid completion of the attack while ensuring a high success rate.

V. Conclusion

In this paper, we have introduced an efficient single-bit approach for NTT polynomial multiplication and an optimized fast two-stage approach, specifically targeting the Dilithium algorithm on a microprocessor. Our experimental results demonstrate that these approaches significantly outperform previous approaches. Moreover, the methodologies we have proposed possess the versatility to be readily adapted to other NTT-based cryptographic implementations, including Kyber and NewHope, broadening their applicability in the field of post-quantum cryptography.

References

[1] Shor P W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer[J]. SIAM review, 1999, 41(2): 303-332.

[2] Kocher P C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems[C]//Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16. Springer Berlin Heidelberg, 1996: 104-113.

[3] Migliore V, Gérard B, Tibouchi M, et al. Masking Dilithium: efficient implementation and side-channel evaluation[C]//Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17. Springer International Publishing, 2019: 344-362.

[4] Fournaris A P, Dimopoulos C, Koufopavlou O. Profiling dilithium digital signature traces for correlation differential side channel attacks[C]//Embedded Computer Systems: Architectures, Modeling, and Simulation: 20th International Conference, SAMOS 2020, Samos, Greece, July 5–9, 2020, Proceedings. Cham: Springer International Publishing, 2020: 281-294.

[5] Chari S, Rao J R, Rohatgi P. Template attacks[C]//Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4. Springer Berlin Heidelberg, 2003: 13-28.

[6] Berzati A, Viera A C, Chartouni M, et al. A Practical Template Attack on CRYSTALS-Dilithium[J]. Cryptology ePrint Archive, 2023.

[7] Primas R, Pessl P, Mangard S. Single-trace side-channel attacks on masked lattice-based encryption[C]//Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Springer International Publishing, 2017: 513-533.

[8] Pessl P, Primas R. More practical single-trace attacks on the number theoretic transform[C]//Progress in Cryptology–LATINCRYPT 2019: 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2–4, 2019, Proceedings 6. Springer International Publishing, 2019: 130-149.

[9] Kim I J, Lee T H, Han J, et al. Novel single-trace ML profiling attacks on NIST 3 round candidate Dilithium[J]. Cryptology ePrint Archive, 2020.

[10] Qiao Z, Liu Y, Zhou Y, et al. Practical Public Template Attack Attacks on CRYSTALS-Dilithium With Randomness Leakages[J]. IEEE Transactions on Information Forensics and Security, 2022, 18: 1-14.

[11] Kocher P, Jaffe J, Jun B. Differential power analysis[C]//Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19. Springer Berlin Heidelberg, 1999: 388-397.

[12] Brier E, Clavier C, Olivier F. Correlation power analysis with a leakage model[C]//Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6. Springer Berlin Heidelberg, 2004: 16-29.

[13] Steffen H, Land G, Kogelheide L, et al. Breaking and Protecting the Crystal: Side-Channel Analysis of Dilithium in Hardware[J]. Cryptology ePrint Archive, 2022.

[14] Chen Z, Karabulut E, Aysu A, et al. An efficient non-profiled side-channel attack on the CRYSTALS-Dilithium post-quantum signature[C]//2021 IEEE 39th International Conference on Computer Design (ICCD). IEEE, 2021: 583-590.

[15] Tunstall M, Hanley N, McEvoy R P, et al. Correlation power analysis of large word sizes[C]//IET Irish Signals and Systems Conference (ISSC). 2007: 145-150.