Release the Power of Rejected Signatures: An Efficient Side-Channel Attack on Dilithium

Zheng Liu, An Wang, Congming Wei, Yaoling Ding, Jingqi Zhang, Annyu Liu, Liehuang Zhu

***Abstract*—The Module-Lattice-Based Digital Signature Standard (ML-DSA), formerly known as CRYSTALS-Dilithium, is a lattice-based post-quantum cryptographic scheme. In August 2024, the National Institute of Standards and Technology (NIST) officially standardized ML-DSA under FIPS 204. Dilithium generates one valid signature and multiple rejected signatures during the signing process. Most Side-Channel Attacks targeting Dilithium have focused solely on the valid signature, while neglecting the hints contained in rejected signatures. In this paper, we propose a method for recovering the private key by simultaneously leveraging side-channel leakages from both valid signatures and rejected signatures. This approach minimizes the number of signing attempts required for full key recovery. We construct a factor graph incorporating all relevant side-channel leakages and apply the Belief Propagation (BP) algorithm for private key recovery.**

We conducted a proof-of-concept experiment on a Cortex M4 core chip, where the results demonstrate that utilizing rejected signatures reduces the required number of traces by at least 42% for full key recovery. A minimum of a single trace can recover the private key with a success rate of 30%. Our findings highlight that protecting rejected signatures is crucial, as their leakage provides valuable side-channel information. We strongly recommend implementing countermeasures for rejected signatures during the signing process to mitigate potential threats.

***Index Terms*—Dilithium, ML-DSA, Side-Channel Attacks, rejected signatures, Belief Propagation.**

I. Introduction

To address the potential threats posed by large-scale quantum computers, NIST initiated the Post-Quantum Cryptography Standardization project in 2016. In 2025, this project entered its fourth round, and three digital signature algorithms have been standardized, namely ML-DSA (formerly known as CRYSTALS-Dilithium), FN-DSA (formerly known as FALCON), and SLH-DSA (formerly known as SPHINCS+). Among them, Dilithium stands out for its excellent overall performance. During the selection process of the project, NIST not only focuses on the performance of the algorithms but also on the minimum overhead required to defend against Side-Channel Attack (SCA). Therefore, side-channel attacks and defenses on Dilithium have been a focus of research in recent years.

SCA extracts secret information stored in cryptographic devices by analyzing side information leaked during their execution. Since Paul Kocher introduced SCA in [1], it has been widely applied to various cryptographic devices. Common types of side information include power traces, electromagnetic emissions, and timing variations.

SCAs are generally categorized into profiling attacks and non-profiling attacks. Profiling attacks assume an attacker with access to a profiling device identical to the victim device. Therefore, an attacker can train a model on the profiling device and then apply the model to the victim device. Through this approach, attackers often achieve relatively good performance in their attacks. Typical profiling attacks include Template Attack (TA), Stochastic Model, and Deep Learning (DL) Side-Channel Attack. Non-profiling attacks do not require attackers to have a profiling device in advance. Instead, attackers typically assume a leakage model of the victim device for the attack. As a result, non-profiling attacks generally offer greater flexibility but tend to have lower performance. Typical non-profiling attacks include Correlation Power Analysis (CPA) and Collision Attacks.

A. Related Work

Many profiling and non-profiling attacks targeting Dilithium have been proposed. In the non-profiling aspect, the first Differential Power Analysis (DPA) attack against Dilithium was proposed by Ravi et al. [2], who targeted the multiplication process of the challenge c and private key s_1 and provided simulation results. They also demonstrated that in the Dilithium algorithm, an attacker does not need to fully recover the private key, but recovering only s_1 is sufficient to forge signatures.

Subsequent researches [3]–[10] have made various improvements based on Ravi et al.'s work. Among them, a notable study by Chen et al. [10] conducted a detailed analysis of the multiplication between c and s_1, as well as the subsequent reduction process, and identified two positions susceptible to CPA. Qiao et al. [3] aimed to analyze the Number Theoretic Transform (NTT) result of the private key such as $\hat{s}_1 = \mathrm{NTT}(s_1)$, $\hat{s}_2 = \mathrm{NTT}(s_2)$. They represented the calculation process of $\hat{s}_1 = \mathrm{NTT}(s_1)$ as $As_1 = \hat{s}_1$, transforming the NTT domain recovery problem into a Small Integer Solution (SIS) problem, which was later resolved by the LLL algorithm or the BKZ algorithm.

Qiao et al. [11], [12] recovered partial bit information of y, constructed equations related to s_1, and solved for s_1 using the Least Squares Method (LSM). [11] and [12] are relatively unique in that they belong to non-profiling attacks but utilize profiling techniques, such as TA, during the attack process.

Zheng Liu, An Wang, Congming Wei, Yaoling Ding, Jingqi Zhang, Annyu Liu, Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: liuzheng98@bit.edu.cn; wangan@bit.edu.cn; weicm@bit.edu.cn; dy119@bit.edu.cn; zhangjq@bit.edu.cn; annvliu@bit.edu.cn; liehuangz@bit.edu.cn).

Corresponding author: Congming Wei.

TABLE I
SUMMARY OF EXISTING WORKS AND COMPARISON WITH THIS WORK

Work	Attack Target	Attack Type	Experiment Type	Signature Used	Main Method
[2]	cs_1	Non-Profiled	Simulation	Valid	DPA
[3]–[10]	cs_1	Non-Profiled	Practical	Valid	CPA
[11], [12]	y	Non-Profiled	Practical	Valid	LSM
[15]	w_0	Profiled	Practical	Valid	LSM
[13], [14]	y, cs_1	Profiled	Practical	Valid	ILP
[18]–[20]	s_1, s_2	Profiled	Practical	Valid	DL
[16]	z, c	Profiled	Practical	Rejected	ILP
[17]	y, z, cs_1	Profiled	Simulation	Valid or Rejected	BP
This Work	y, c, cs_1	Profiled	**Practical**	**Valid and Rejected**	BP

In the profiling aspect, early attacks focused on directly targeting operations related to s_1 and s_2, such as the unpacking process of s_1 and s_2, or transformations like NTT(s_1) and NTT(s_2). However, in practical deployment scenarios, such computations may only occur once or not at all, making these attack methods difficult to apply in real-world settings.

Subsequent researches generally follow the same approach: leveraging SCA to get hints about the private key, and using different methods to solve for the private key. [13]–[16] obtain numerical hints about s_1 and solve it using LSM or Integer Linear Programming (ILP). [17] obtains distributional hints about s_1 and solves it using Belief Propagation (BP).

B. Motivation

Dilithium follows the Fiat-Shamir paradigm, generating one valid signature along with multiple rejected (invalid) signatures. In our experiments, we analyzed the number of rejected signatures during Dilithium's signing process. We found that approximately 77% of signing attempts generate at least one rejected signature, as shown in Table II. Relying solely on valid signatures for key recovery results in a significant waste of information contained in rejected signatures. Although some prior works [16], [17] have attempted to exploit rejected signature information for private key recovery, their approaches suffer from low efficiency. Therefore, this paper focuses on how to effectively utilize both valid and rejected signatures to achieve efficient private key recovery.

Despite BP being widely applied in the field of post-quantum cryptography, its application to Dilithium has not been fully exploited. In BP, the flexibility of the factor graph makes BP particularly well-suited for scenarios where multiple hints from different sources are combined to recover the private key. Additionally, BP's probabilistic inference mechanism provides high error tolerance, making it ideal for recovering the private key from a large number of noisy hints. This is especially beneficial in our case, where hints extracted from rejected signatures may contain more noise compared to those from valid signatures.

Given these motivations, this paper proposes a BP-based method to efficiently utilize both valid and rejected signatures to recover the private key and provides practical experimental results to demonstrate its effectiveness.

TABLE II
STATISTICS ON THE NUMBER OF REJECTED SIGNATURES IN DILITHIUM

Rejection	0	1	2	3	4	5	>5
Ratio(%)	23.0	17.8	13.7	10.7	8.0	6.3	20.5

C. Contribution

The contributions of this paper are as follows.

- To the best of our knowledge, we are the first work that efficiently leverages both valid and rejected signatures for private key recovery in Dilithium. Since each valid signature is accompanied by approximately three rejected signatures on average, our method can theoretically reduce the required number of signing attempts by up to 75% compared to approaches that utilize only hints from valid signatures. Our experimental results show that our method reduces the number of signing attempts by at least 42%. Our approach requires a minimum of a single signing attempt and a maximum of 4 signing attempts to fully recover the private key.
- We propose a generalized factor graph structure for Dilithium, designed to maximize the utilization of all available hints generated during a single signing attempt for private key recovery. With this factor graph structure, an attacker can simultaneously leverage hints extracted from both valid and rejected signatures to recover the private key, thereby minimizing the number of signing attempts required for full key recovery. Additionally, the proposed factor graph offers high flexibility—the attacker can choose to use all hints obtained from SCA for key recovery or selectively utilize only the most accurate hints to maximize the success rate of full key recovery.
- We provide the missing details from previous studies on applying BP to Dilithium, offering a detailed and complete account of the entire attack process—from side-channel leakage extraction to the construction of private key hints. In addition, we introduce a series of computational optimizations, including an improved message propagation strategy and a carefully constrained message size, to enhance the efficiency and practicality of BP.

To validate our approach, we performed experiments on a practical dataset, thoroughly evaluating both the time and memory overhead. The results demonstrate that our method is not only effective but also efficient: it achieves successful private key recovery within 300 seconds, with memory consumption remaining under 250 MB.

All code and datasets used in this paper are publicly available at

https://github.com/AIGIUS/BP-On-Dilithium

II. Background

A. Notations

First, we define some notations. We adopt the notation from [21]. Let $\mathbb{Z}$ denote the ring of integers, $\mathbb{Z}_q$ denote the ring of integers modulo q, $\mathbb{Z}_q[X]$ denote the ring of polynomials modulo q, $\mathbb{R}_q = \mathbb{Z}_q[X]/(X^n+1)$ denote the quotient ring of polynomials modulo X^n+1. We use $[i,j]$ to represent the set $\{m \mid i \leq m \leq j, m \in \mathbb{Z}\}$. We use boldface letters to represent vectors or matrices, and superscripts to indicate their dimensions. We use subscripts to represent the coefficients of a polynomial or an element of a vector. We use square brackets in subscripts to represent certain bits of an integer in its binary representation. For $\forall r \in \mathbb{Z}$, $r^{'} = r \ mod^{\pm} \ q$ represents $r^{'} \equiv r \ mod \ q$ where $r^{'} \in (-q/2, q/2]$. For $w \in \mathbb{Z}$, let $||w||_\infty = |w \ mod^{\pm} \ q|$. For $w \in \mathbb{R}_q$, let $||w||_\infty = max||w[i]||_\infty$. For $\mathbf{w} \in \mathbb{R}_q^k$, let $||\mathbf{w}||_\infty = max||w_i||_\infty$. S_η denotes the set $\{w \mid w \in \mathbb{R}_q, ||w||_\infty \leq \eta\}$. B_τ denotes the set of all elements in $\mathbb{R}_q$ that have exactly τ coefficients equal to 1 or -1, with the remaining coefficients being 0. The NTT representation of a polynomial $w \in \mathbb{R}_q$ is $\hat{w} = \text{NTT}(w)$. The NTT representation of a vector or matrix is obtained by applying the NTT to each of its elements. We use $\cdot$ to denote the vector dot product, and $\circ$ to denote convolution. We use $\overline{z}$ and $\overline{c}$ to represent the rejected signatures, while z and c denote the valid signatures in Dilithium.

B. DILITHIUM

Dilithium is one of the three digital signature algorithms standardized by NIST. Its security is primarily based on the lattice-based Module Learning with Errors (M-LWE) problem and the Module Short Integer Solution (M-SIS) problem. Dilithium consists of three parts: key generation, signing, and verification. In this paper, we mainly explain the key generation and signing parts. The verification part is generally not targeted for SCA, so it is not covered. For the sake of brevity, we present only the algorithm templates and the necessary details.

Algorithm 1 Dilithium key generation algorithm

$\mathbf{A} \leftarrow \mathbb{R}_q^{k \times l}$
$(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^l \times S_\eta^k$
$\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
return $(pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2))$

Algorithm 2 Dilithium signing algorithm

$\mathbf{z} = \perp$
while $\mathbf{z} = \perp$ **do**
  $\mathbf{y} \leftarrow S_{\gamma_1 - 1}^l$
  $\mathbf{w} = \text{NTT}^{-1}(\hat{\mathbf{A}} \cdot \text{NTT}(\mathbf{y}))$
  $\mathbf{w}_1 = \text{HighBits}(\mathbf{w}, 2\gamma_2)$
  $c \in B_\tau = \text{H}(M||\mathbf{w}_1)$
  $\mathbf{z} = \mathbf{y} + \text{NTT}^{-1}(\text{NTT}(c) \cdot \hat{\mathbf{s}}_1)$
  if $||z||_\infty \geq \gamma_1 - \beta$ or $||\text{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)||_\infty \geq \gamma_2 - \beta$ **then**
    $\mathbf{z} = \perp$
  end if
end while
return $\sigma = (\mathbf{z}, c)$

C. Template Attack

Template attack [22] is one of the most commonly used methods for profiling Side-Channel Attacks. TA consists of two phases: the profiling phase and the attack phase. In the profiling phase, the profiling device runs n times, recording the input $\mathbf{p}_i$, the corresponding output σ_i, and the associated power trace $\mathbf{t}_i$, where $i \in [0, n-1]$. Using the key, inputs, and corresponding output, the intermediate values $\mathbf{v}_i$ that the attacker aims to attack can be computed. The power traces are then categorized based on $\mathbf{v}_i$.

Assuming $\mathbf{v}_i$ can be divided into d different categories. Let the power trace belonging to the j-th category be denoted as $\mathbf{t}_{j,i}$, where $i \in [0, \mathbf{d}_j - 1]$, $\mathbf{d}_j$ represents the number of traces in the j-th category. The mean vector $\mathbf{m}_j$ and the covariance matrix $\mathbf{C}_j$ for the j-th category can be calculated using the following formulas:

$$\mathbf{m}_j = \frac{1}{\mathbf{d}_j} \sum_{i=0}^{\mathbf{d}_j - 1} \mathbf{t}_{j,i},$$

$$\mathbf{C}_j = \frac{1}{\mathbf{d}_j} \sum_{i=0}^{\mathbf{d}_j - 1} (\mathbf{t}_{j,i} - \mathbf{m}_j)^T (\mathbf{t}_{j,i} - \mathbf{m}_j).$$

In the attacking phase, the attacker collects $n^{'}$ power traces $\mathbf{t}_i^{'}$ when running the victim device, where $i \in [0, n^{'} - 1]$. The probability that $\mathbf{t}_i^{'}$ belongs to the j-th template can be calculated using the following formula:

$$P_{i,j}(\mathbf{t}_i^{'}; (\mathbf{m}_j, \mathbf{C}_j)) = \frac{exp(-\frac{1}{2}(\mathbf{t}_i^{'} - \mathbf{m}_j)\mathbf{C}_j^{-1}(\mathbf{t}_i^{'} - \mathbf{m}_j)^T)}{\sqrt{(2\pi)^o \cdot det(\mathbf{C}_j)}},$$

where o is the dimension of $\mathbf{t}_i$ and $\mathbf{t}_i^{'}$.

D. Number Theoretic Transform

The Number Theoretic Transform (NTT) is a discrete Fourier transform (DFT) variant that operates over finite fields [23], making it particularly useful in cryptographic applications, such as lattice-based cryptography.

Given a polynomial $a \in \mathbb{R}_q$, its NTT transformation is defined as $\hat{a} = \text{NTT}(a)$, as shown in Equation 1, where ω is the primitive n-th root of unity modulo q.

$$\hat{a}_j = \sum_{i=0}^{n-1} \omega^{ij} a_i \ mod \ q \tag{1}$$

The inverse NTT (INTT) is similarly defined as $a = \text{INTT}(\hat{a})$, as shown in Equation 2, where n^{-1} is called the scaling factor.

$$a_i = n^{-1} \sum_{j=0}^{n-1} \omega^{-ij} \hat{a}_j \ mod \ q \tag{2}$$

In practice, the Cooley-Tukey (CT) algorithm is commonly used for efficient computation of the NTT. The fundamental computational unit in the CT algorithm is the CT butterfly unit. As shown in Figure 1, a CT butterfly unit performs the computation described by Equation 3. By configuring multiple such units in a structured manner, a complete NTT computation can be efficiently executed.

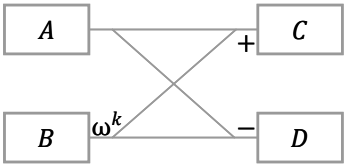


Fig. 1. The Cooley-Tukey butterfly unit.

$$\begin{aligned} C &= A + B \times \omega^k \\ D &= A - B \times \omega^k \end{aligned} \tag{3}$$

E. Belief Propagation

BP is a probabilistic inference algorithm widely used in graphical models, particularly in factor graphs, to compute marginal distributions of variables. A factor graph consists of factor nodes and variable nodes. Variable nodes represent the variables, while factor nodes represent the constraints imposed on these variables. BP operates by iteratively passing messages between variable nodes and factor nodes in a factor graph. These messages represent probability distributions, allowing the algorithm to update its belief about each variable until convergence is achieved.

The message sent from a factor node a to a variable node i is computed as the product of all incoming messages from a's neighboring nodes (excluding i), multiplied by the factor function of a, and then marginalized over i. This is formally expressed in Equation 4, where $N(a)$ denotes the set of neighboring nodes of a.

$$m_{a \to i}(x_i) = \sum_{x_{N(a)\setminus\{i\}}} f_a(x_i, x_{N(a)\setminus\{i\}}) \prod_{j \in N(a)\setminus\{i\}} m_{j \to a}(x_j) \tag{4}$$

The message sent from a variable node i to a factor node a is computed as the product of all incoming messages from i's neighboring nodes (excluding a). This process is illustrated in Equation 5.

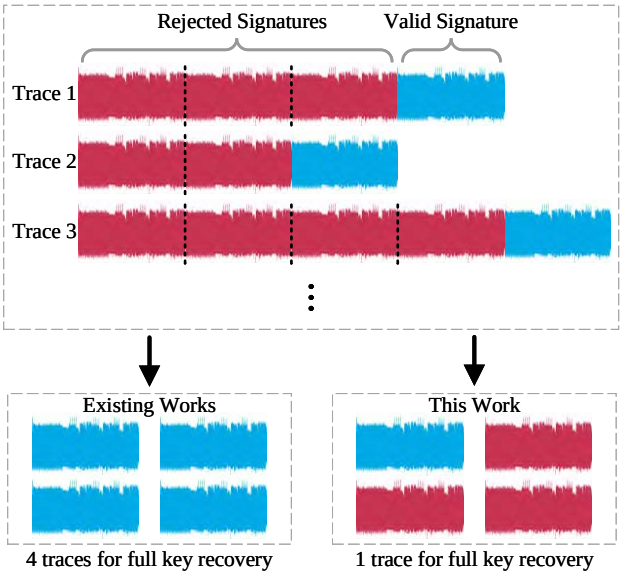


Fig. 2. Main idea of the attack.

$$m_{i \to a}(x_i) = \prod_{b \in N(i)\setminus\{a\}} m_{b \to i}(x_i) \tag{5}$$

BP was introduced into SCA in [24]. In this context, the results of SCA are typically referred to as hints. These hints are used to initialize the factor nodes in the factor graph, after which the BP algorithm is executed to solve for the private key.

III. General Attack Framework

The core idea of this paper is to maximize the utilization of all side-channel leakages within a single signing attempt to recover the private key as efficiently as possible. For each Dilithium signing attempt, we can capture a corresponding side-channel trace. Each trace contains leakages from one valid signature and multiple rejected signatures, as illustrated in Figure 2. Assuming that four signatures are sufficient to recover the private key, an attacker using only valid signatures would require four traces for key recovery. However, by leveraging both valid and rejected signatures, the attacker has the opportunity to recover the private key using fewer traces, and in some cases, may even achieve full key recovery from a single trace, as illustrated in Figure 2.

A. Overview of the Attack Strategy

This section outlines the overall attack framework proposed in this paper. The framework consists of four main processes, as illustrated in Figure 3. Typically, profiled SCA consists of two phases, which are the profiling phase and the attack phase. For the sake of clarity, we only present the attack phase, omitting the profiling phase. We describe the four stages as follows:

- **Side-Channel Leakage Collection**: The attacker collects side-channel information from the victim's device. Common types of side-channel information include power

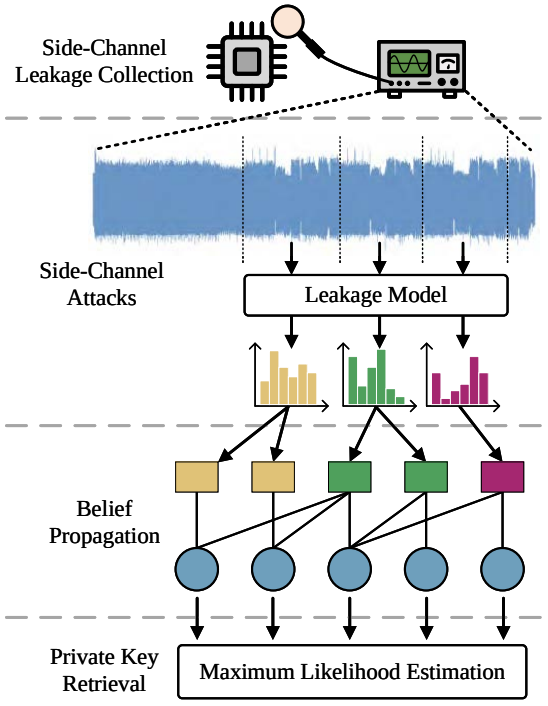


Fig. 3. General attack framework.

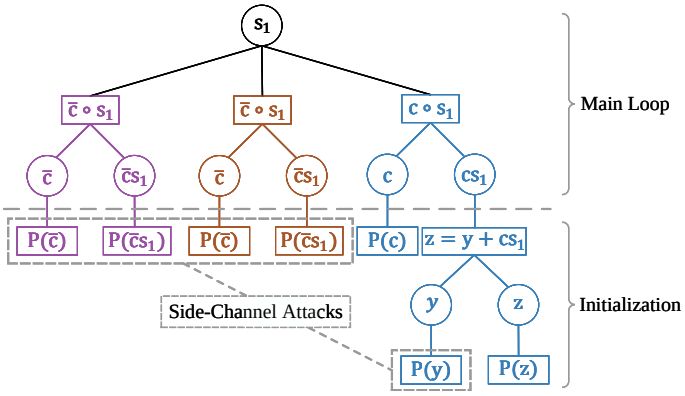


Fig. 4. The factor graph used in this paper. $\overline{c}$ denotes rejected signature and c denotes valid signature. Different colors represent different signature instances. $\mathrm{P}(\overline{c})$, $\mathrm{P}(\overline{c}s_1)$, and $\mathrm{P}(y)$ represent hints obtained through SCA.

traces, EM traces. In this work, we choose to collect EM traces and specifically measure the EM emissions near the voltage regulator pin of the victim device to achieve a higher signal-to-noise ratio (SNR).

- **Side-Channel Attacks**: The attacker identifies the positions of rejected signatures and valid signatures within EM traces, and classifies them using the leakage model obtained from the profiling device. The classification process yields hints of the target variables. Specifically, these variables are as follows:
  - Valid Signatures: y, cs_1
  - Rejected Signatures: $\overline{c}$, $\overline{c}s_1$
- **Belief Propagation**: The attacker constructs a factor graph, initializing it with hints obtained from the leakage model. The BP algorithm is then executed iteratively until the messages in the factor graph converge.
- **Private Key Retrieval**: Once the BP algorithm converges, the attacker can recover the private key from variable nodes. A common method for this is maximum likelihood estimation.

The following subsections will discuss specific details and issues related to the general attack framework.

B. Choices in Side-Channel Attacks

Common techniques for constructing leakage models include TA, stochastic models, and DL. In this work, we employ TA. While DL may potentially yield more accurate leakage models, TA is chosen for its advantages, including parameter-free implementation and consistently reliable performance. This makes TA well-suited for proof-of-concept validation.

This work targets $\overline{c}s_1$ instead of $\overline{z}$, for existing research has shown that the hint from $\overline{z}$ is too limited [16], [17]. Incorporating $\overline{z}$ might introduce additional noise, degrading performance. In addition, SCA on $\overline{z}$ is not the primary focus of this paper. Nevertheless, we demonstrate that $\overline{z}$ can be easily integrated into the proposed attack framework in Section III-C. The details of the SCA on y, cs_1 (or $\overline{c}s_1$), and $\overline{c}$ will be described in Section IV.

C. Factor Graph Construction

As shown in Figure 4, the factor graph used in this paper is illustrated. Since c and z (as part of the valid signature) are known to the attacker, they could be integrated into the factor node $z = y + cs_1$. However, we still represent $\mathrm{P}(c)$ and $\mathrm{P}(z)$ as independent factor nodes in the factor graph, where only one value has a probability of 1, and all others have a probability of 0. This enhances the generality of the proposed factor graph. If an attacker can obtain a sufficiently accurate hint about the rejected z, it can be incorporated into our factor graph simply by replacing z with $\overline{z}$.

The factor graph includes two types of constraints in factor nodes: polynomial multiplication ($\overline{c} \circ s_1$, $c \circ s_1$) and polynomial addition ($z = y + cs_1$). Constraints $c \circ s_1$ and $\overline{c} \circ s_1$ follow the Equation 6, where $\zeta = 0$ when $j \leq i$, $\zeta = 1$ when $j > i$, $i \in [0, 255]$. Constraint $z = y + cs_1$ follows the Equation 7, where $i \in [0, 255]$.

$$(cs_1)_i = \sum_{j=0}^{255} (-1)^{\zeta} c_{(i-j) \bmod 256} \times (s_1)_j \tag{6}$$

$$z_i = y_i + (cs_1)_i \tag{7}$$

Compared to the factor graph in existing work [17], the factor graph proposed in this paper represents c (as well as $\overline{c}$) and z as probability distributions, enabling the utilization of both valid and rejected signatures. Additionally, the factor graph in this paper explicitly represents the constraint $z = y + cs_1$, allowing s_1 to be directly recovered from SCA results. In contrast, existing work needs to derive the distribution of cs_1 from y before incorporating it into the factor graph. Our work is more automated, reducing manual processing steps.

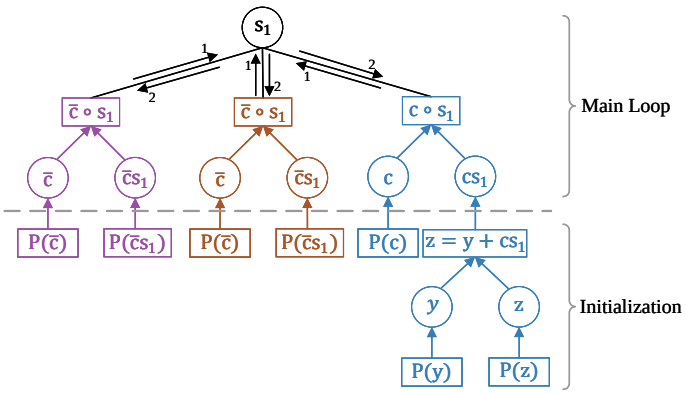


Fig. 5. Message passing path in the factor graph.

D. Efficient BP computation

In the factor graph used in this paper, the BP computational overhead mainly comes from three sources: the iterative message passing process, the message size, and the computation process of the constraints. The following content will discuss these three parts separately.

In a standard BP algorithm, each iteration requires bidirectional message passing along every edge in the factor graph. However, in the factor graph proposed in this paper, this process can be pruned to reduce computational complexity. This paper adopts the message passing path as shown in Figure 5.

As shown in Figure 5, the initialization part only consists of values obtained from SCA results or known values. Consequently, messages in this part remain unchanged throughout the BP process, propagating only upward and requiring computation only once during the first iteration of BP.

In the main loop part, only the edges directly connected to s_1 require bidirectional message passing, while all other edges propagate messages upward only. At a higher level, the factor graph of the main loop forms a tree-like structure, thus, the edges directly connected to s_1 do not require parallel bidirectional updates. Instead, the message passing follows a top-down and bottom-up iterative process [25].

Considering message size, the main loop contains three types of messages: s_1, c (or $\overline{c}$), and cs_1 (or $\overline{c}s_1$). Since these variables have relatively small value ranges, their impact on BP computational complexity is minimal. In contrast, in the initialization phase, y has a much larger value range of 2^{17} (or 2^{19} for Dilithium3 and Dilithium5). This results in a larger message size, which not only increases memory overhead but also prolongs constraint computations in factor nodes.

To control the message size of y, we analyze the $z = y + cs_1$ process and limit the bits used for y, thereby reducing computational complexity. The details of this optimization are provided in Section IV-A.

Regarding the computational cost of the constraints, Equation 6 and Equation 7 involve two fundamental distribution operations: the sum of two distributions $\mathrm{P}(A + B)$ and the product of two distributions $\mathrm{P}(A \times B)$. Since $\mathrm{P}(A + B) = \mathrm{P}(A) \circ \mathrm{P}(B)$ when A and B are independent, the computation can be accelerated using the NTT. Additionally, since the bits used for y have been optimized, Equation 7 can also be further refined. The details are provided in Section IV-A. For $\mathrm{P}(A \times B)$, since the coefficients of c are restricted to $\{-1, 0, 1\}$, $\mathrm{P}(A \times B)$ equals $\mathrm{P}(B)$, $\mathrm{P}(-B)$, or 0, which can be obtained by simply flipping the distribution of B.

IV. Side-Channel Attacks on Dilithium

This section will describe how to perform SCA on the target variables y, cs_1 (or $\overline{c}s_1$), and $\overline{c}$, respectively. For y, we primarily limit the number of bits recovered through SCA to reduce the message size in the initialization part of the factor graph, thereby lowering the computational overhead of BP. For $\overline{c}$ and cs_1 (or $\overline{c}s_1$), we describe their leakage and the corresponding SCA, ensuring the effective utilization of rejected signatures.

A. Side-Channel Attack on y

In this section, we present how to perform SCA on y. We directly attack the generation process of y, which corresponds to line 3 of Algorithm 2. Since each coefficient of y is bounded by γ_1 (2^{17} for Dilithium2 and 2^{19} for Dilithium3 and Dilithium5), modeling all bits of y's coefficients not only complicates the profiling process but also significantly increases the overhead of message passing in BP and the computation cost of the constraints. Therefore, minimizing the number of bits of y to be modeled is a primary consideration. Although some studies have discussed this issue [11]–[14], the question of "what is the minimum number of bits of y that must be modeled to accurately recover the distribution of cs_1" has not yet been fully answered. To address this issue, we propose and prove the following proposition in this section.

Proposition 1. *Given $|cs_1| < 2^\tau$, at least $\tau + 2$ bits of y are required to accurately reconstruct the distribution of cs_1, where $z = y + cs_1$, and z, y, c, s_1 refer to the coefficients of the corresponding polynomial.*

Proof. On a typical 32-bit processor, z, y, and cs_1 are stored and computed in the form of 32-bit two's complement integers. We can express y, z, and cs_1 as the sum of their higher $32-\tau$ bits and lower τ bits, as shown in Equation 8.

$$\begin{aligned} z_{[31:\tau]} \times 2^\tau + z_{[\tau-1:0]} = y_{[31:\tau]} \times 2^\tau + y_{[\tau-1:0]} \\ +(cs_1)_{[31:\tau]} \times 2^\tau + (cs_1)_{[\tau-1:0]} \end{aligned} \tag{8}$$

Given $|cs_1| < 2^\tau$, $(cs_1)_{[31:\tau]}$ can only take the value 0 or -1, depending on the sign of cs_1. Then the relationship between $z_{[31:\tau]}$ and $y_{[31:\tau]}$ has only three possible forms: $z_{[31:\tau]} = y_{[31:\tau]}$, $z_{[31:\tau]} = y_{[31:\tau]} - 1$, or $z_{[31:\tau]} = y_{[31:\tau]} + 1$, depending on whether a carry occurring in $y_{[\tau-1:0]} + (cs_1)_{[\tau-1:0]}$.

Combing Equation 8, four scenarios about $(cs_1)_{[31:0]}$ can arise:

1) With carry, $cs \geq 0$:

$$z_{[31:\tau]} = y_{[31:\tau]} + 1, (cs_1)_{[31:0]} = z_{[\tau-1:0]} - y_{[\tau-1:0]} + 2^\tau$$

2) Without carry, $cs \geq 0$:

$$z_{[31:\tau]} = y_{[31:\tau]}, (cs_1)_{[31:0]} = z_{[\tau-1:0]} - y_{[\tau-1:0]}$$

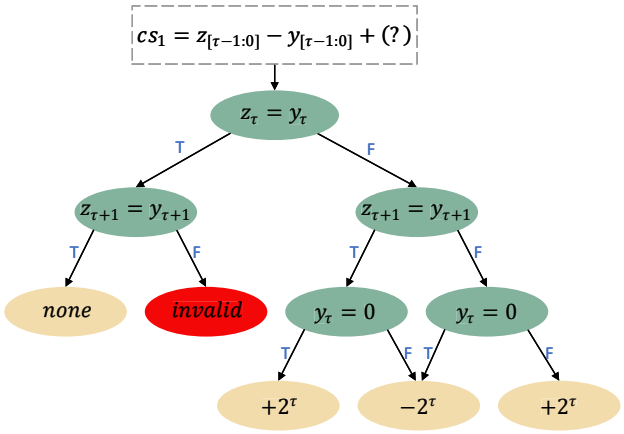


Fig. 6. The decision tree used for the relationship construction between cs_1, $z_{[\tau+1:0]}$, and $y_{[\tau+1:0]}$.

Listing 1 The C code corresponding to the final step of INTT.

```
for(j = 0; j < N; ++j) {
  a[j] = montgomery_reduce((int64_t)f * a[j]);
}
```

3) With carry, $cs < 0$:

$$z_{[31:\tau]} = y_{[31:\tau]}, (cs_1)_{[31:0]} = z_{[\tau-1:0]} - y_{[\tau-1:0]}$$

4) Without carry, $cs < 0$:

$$z_{[31:\tau]} = y_{[31:\tau]} - 1, (cs_1)_{[31:0]} = z_{[\tau-1:0]} - y_{[\tau-1:0]} - 2^\tau$$

Summarizing the four possible cases, it can be observed that $(cs_1)_{[31:0]}$ depends only on $z_{[\tau-1:0]} - y_{[\tau-1:0]}$ and the relationship between $z_{[31:\tau]}$ and $y_{[31:\tau]}$. Since the relationship between $z_{[31:\tau]}$ and $y_{[31:\tau]}$ only has three possible cases, recovering the lower 2 bits of $y_{[31:\tau]}$ is sufficient to capture this relationship. Therefore, recovering the lower $\tau+2$ bits of each y is sufficient to accurately reconstruct the distribution of cs_1. □

Furthermore, we present a method for computing cs_1 from Equation 7 when only $y_{[\tau+1:0]}$ is recovered, as illustrated in Figure 6. Specifically, the relationship between $z_{[\tau+1:\tau]}$ and $y_{[\tau+1:\tau]}$ is used to determine the relationship between cs_1 and $z_{[\tau-1:0]} - y_{[\tau-1:0]}$, which then enables the computation of cs_1.

B. Side-Channel Attack on cs_1

In this section, we present how to perform SCA on cs_1 (as well as $\bar{c}s_1$). We focus on the final step of INTT($\hat{c}\hat{s}_1$) as shown in Listing 1, which corresponds to the multiplication by n^{-1} in Equation 2. Since this operation involves a multiplication combined with Montgomery reduction, it naturally exhibits a high SNR in SCA. We directly model the lower 7 bits of the coefficients of cs_1, for $||cs_1||_\infty$ is less than 2^6 with nearly 100% probability according to a previous study [13].

C. Side-Channel Attack on $\bar{c}$

In this section, we present how to perform SCA to recover the rejected signature $\bar{c}$. We recover $\bar{c}$ by analyzing the process of NTT($\bar{c}$). SCA on the NTT process has been extensively discussed in several studies [16], [26], [27]. In this work, we apply a layer-by-layer recovery method that neither relies on BP nor requires a full analysis of the NTT process.

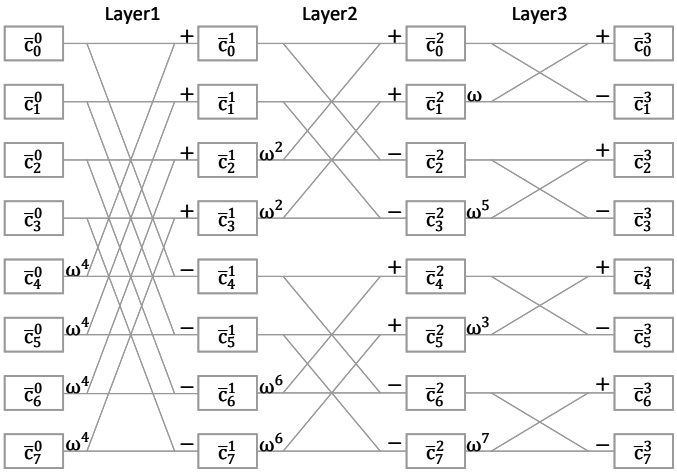
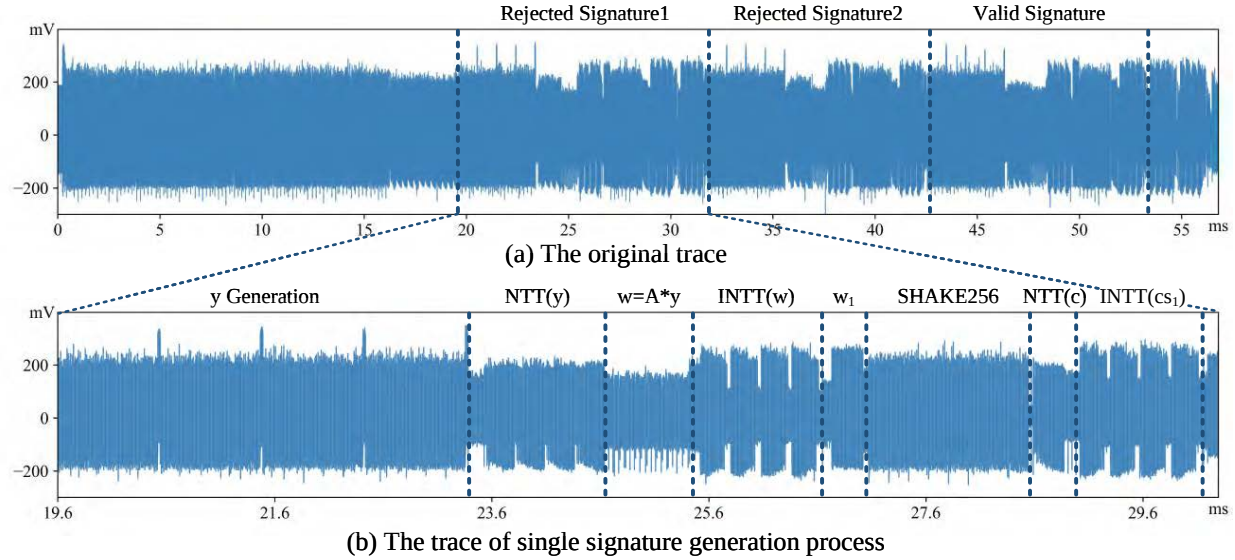


Fig. 7. Simplified NTT process.

Rather than analyzing the entire NTT process, we focus on attacking the first few layers of NTT. First, we analyze the first layer of the NTT. A simplified illustration of the NTT($\bar{c}$) process is shown in Figure 7. Since $\bar{c}_i^0 \in \{-1, 0, 1\}$, there are only 9 possible input combinations for each butterfly unit in the first layer. Therefore, 9 templates are required for the attack. Considering each butterfly unit in the figure, the values of $\{\bar{c}_i^0 | i \in [4, 7]\}$ significantly influence the hamming weight (HW) of the butterfly unit's output. Specifically, when $\bar{c}_i^0$ takes values in $\{-1, 0, 1\}$, the hamming weight of the output corresponds to HW($\bar{c}_{i-4}^0 - \omega^4$), HW($\bar{c}_{i-4}^0$), or HW($\bar{c}_{i-4}^0 + \omega^4$) separately. Through similar analysis, it can be observed that the values of $\{\bar{c}_i^0 | i \in [0, 3]\}$ have a smaller impact on the hamming weight of the butterfly unit's output. Consequently, the recovery accuracy for $\{\bar{c}_i^0 | i \in [4, 7]\}$ is high, while $\{\bar{c}_i^0 | i \in [0, 3]\}$ may contain errors.

To correct these errors, the attacker can proceed to attack the second layer. Leveraging the fact that $\{\bar{c}_i^0 | i \in [4, 7]\}$ has already been recovered, this constrains $\bar{c}_i^1$ to only three possible values. Similar to the first layer, in each butterfly unit, the second input value is easier to recover, while the first input value is more error-prone. By iteratively attacking subsequent layers, the attacker can progressively refine and correct errors from previous layers, ultimately improving the overall attack accuracy.

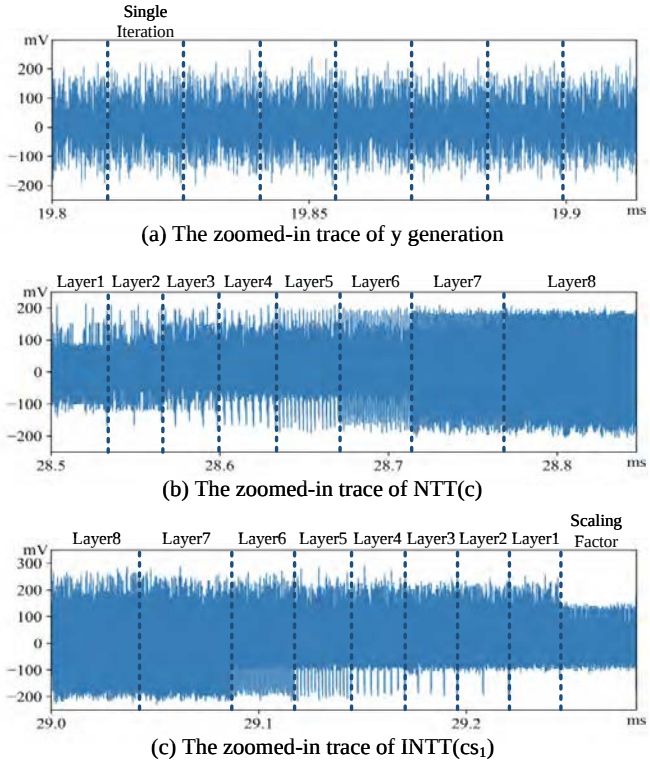
An attacker can always target more layers to achieve higher accuracy for $\bar{c}_i^0$; however, the number of templates required per layer increases. For example, the first layer requires 9 templates, while the second layer requires 81 templates (although only 9 of them may be used). This represents a trade-off between cost and performance. In this work, thanks to the high error tolerance of BP, our experiments show that attacking only the first layer is sufficient for the private key recovery.



(a) The original trace

(b) The trace of single signature generation process

Fig. 8. EM traces of the reference implementation of Dilithium on STM32F407 with optimization O3.



(a) The zoomed-in trace of y generation

(b) The zoomed-in trace of NTT(c)

(c) The zoomed-in trace of INTT(cs_1)

Fig. 9. Zoomed-in traces of y generation, NTT(c), and INTT(cs_1).

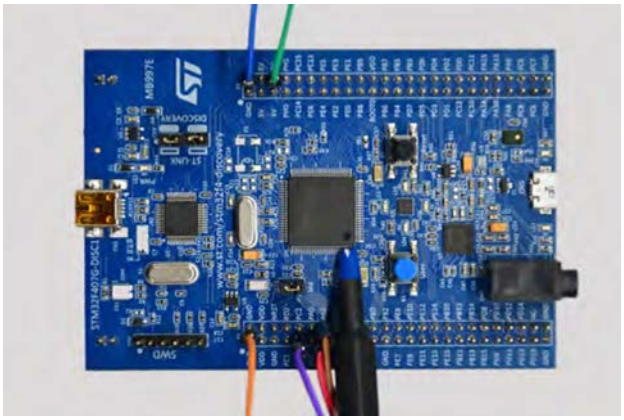


Fig. 10. Experimental environment.

V. Experiment Results

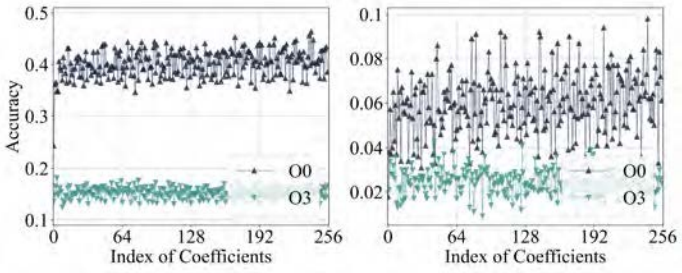
A. Experimental Setup

The experimental setup is shown in Figure 10. We downloaded the reference implementation of Dilithium into an STM32F407 Discovery development board. EM traces were collected using a Langer LF-U 2.5 probe, amplified by a Langer PA 303 amplifier, and then captured by a PICO 3203D oscilloscope. The traces were then transferred to a laptop equipped with an Intel i9-14900HX processor and 64GB RAM for further analysis. We conducted a proof-of-concept experiment on the reference implementation of Dilithium under both O0 and O3 optimizations.

B. Side-Channel Attacks Results

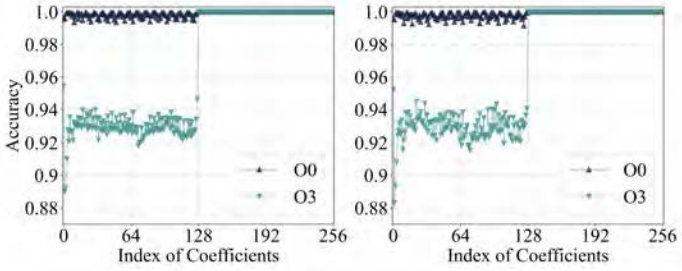
Figure 8 shows the EM traces of the reference implementation of Dilithium on STM32F407 with optimization O3. Figure 8a presents the complete trace of a single signing process, which includes two rejected signatures and one valid signature. Figure 8b shows a zoomed-in view of the single signature generation process from Figure 8a. For ease of description, we do not distinguish between c and $\overline{c}$, nor between cs_1 and $\overline{c}s_1$ here. We have marked the main operations during a single signature generation in the Figure 8b, corresponding to lines 3 to 7 of Algorithm 2. In this trace, the generation of $\mathbf{y}$, the NTT(c) process, and the INTT($c\mathbf{s}_1$) process are the primary targets. As clearly shown in Figure 8b, both the $\mathbf{y}$ generation

process and the INTT($c\mathbf{s}_1$) process contain four major loops, since both $\mathbf{y}$ and $c\mathbf{s}_1$ are elements of $\mathbb{R}_q^k$, and the operations on them are performed separately on each polynomial component.



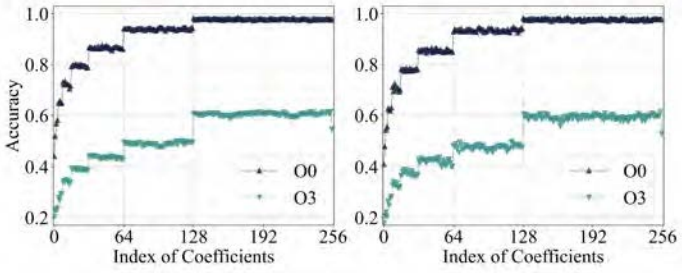
(a) Train set (b) Test set

Fig. 11. Side-Channel Attack Results of y.



(a) Train set (b) Test set

Fig. 12. Side-Channel Attack Results of NTT(c).



(a) Train set (b) Test set

Fig. 13. Side-Channel Attack Results of INTT(cs_1).

Figure 9a provides a further zoomed-in view of y. With further zooming, we can observe the trace corresponding to a single iteration of the y generation. Figure 9b presents a zoomed-in view of the NTT(c) process, where the 8 layers of the NTT can be identified. In the figure, the distinct downward peaks correspond to butterfly units with different twiddle factors. The peaks become denser in the later layers, as these layers involve a greater number of twiddle factors, as illustrated in Figure 7. We apply SCA on the first layer of NTT(c). Figure 9c shows a zoomed-in view of the INTT(cs_1) process, which is roughly the reverse of the process shown in Figure 9b. The main difference is that, following the first layer, there is an additional multiplication by the scaling factor, corresponding to Listing 1. This multiplication step is also the target of SCA.

TABLE III
TIME AND MEMORY OVERHEAD OF THE BP ALGORITHM

Number of Signatures	10	100	1000	5000	10000
Time Per Iteration (s)	1.5	14.6	148	743	1495
Memory (MB)	231	339	1440	6323	12426

We collected traces from $10,000$ signing attempts, using $9,000$ traces as the training set and $1,000$ traces as the test set. We use TA combined with Linear Discriminant Analysis (LDA) as the SCA method. The results for y, c, and cs_1 are shown in Figures 11, 12, and 13, respectively. In these figures, the horizontal axis represents the polynomial coefficient index, and the vertical axis represents the accuracy. We perform TA separately on the different coefficients of the polynomials, as the attack accuracy varies across coefficients, as shown in Figures 11, 12, and 13. Using the training set, we identify the coefficients with higher accuracy, and in the test phase, we prioritize the use of these high-accuracy coefficients to improve the overall effectiveness of the attack.

C. BP Results

As shown in Figures 16 and 17, the results of the BP algorithm under different number of traces and optimization levels are presented. The horizontal axis represents the number of traces required for the attack, while the vertical axis represents either the success rate or the number of recovered key coefficients. For each trace count, we repeated the experiment 10 times and reported the average results.

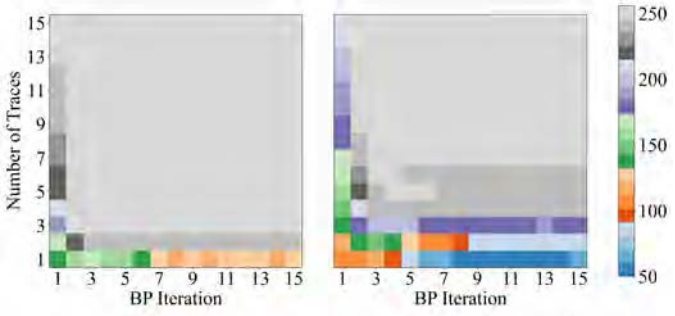
The results indicate that under O0 optimization, using all signatures (both valid and rejected), we require a minimum of 1 and a maximum of 3 traces for full key recovery. In contrast, using only valid signatures, it takes a minimum of 3 and a maximum of 7 traces for full key recovery. The use of all signatures decreases the number of traces by approximately 57% compared to using only valid signatures.

Under O3 optimization, using all signatures, the attack requires a minimum of 2 and a maximum of 4 traces for full key recovery. When using only valid signatures, the attack requires a minimum of 2 and a maximum of 7 traces. In this case, leveraging all signatures decreases the number of traces by approximately 42% over using only valid signatures.

As shown in Figures 14 and 15, the variation in the number of recovered key coefficients during the BP iteration is depicted for different trace counts and optimization levels, where the color represents the average number of recovered key coefficients.

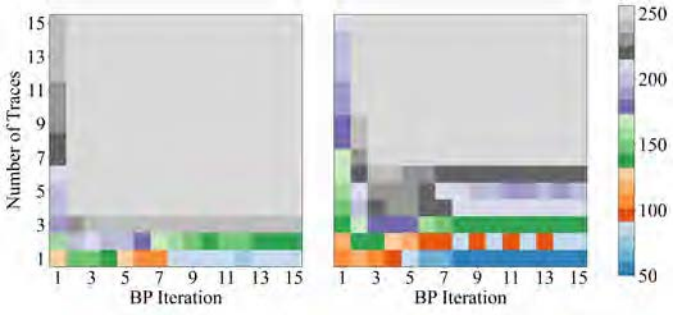
It can be observed that the BP algorithm converges around the 10-th iteration. If the key coefficients can be fully recovered, this typically occurs within a maximum of 3 iterations. However, BP does not always converge in a favorable direction. When fewer signing attempts are used in the attack, the number of recovered key coefficients may initially increase but then decrease and converge to a very low value, as illustrated in the case where only a single trace is used.

As shown in Table III, the time and memory overhead of the BP algorithm vary with the number of signatures used.



(a) All signatures (b) Valid signatures

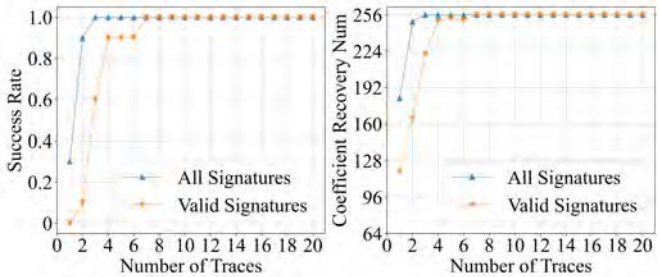
Fig. 14. Recovered Coefficients Number In BP Iteration With Optimization O0.



(a) All signatures (b) Valid signatures

Fig. 15. Recovered Coefficients Number In BP Iteration With Optimization O3.

On average, each signature requires approximately 1.5 seconds per BP iteration. Additionally, each additional signature incurs an extra memory overhead of approximately 1.2 MB. These data were collected under a single-core setup. With multi-core optimization, the time required per iteration can be further reduced. Considering that each signing attempt typically produces around four signatures, to fully recover the coefficients of s_1, the average BP iteration time required is approximately 216s for O0 and $288s$ for O3. The corresponding memory overhead is 234 MB for O0 and 240 MB for O3.

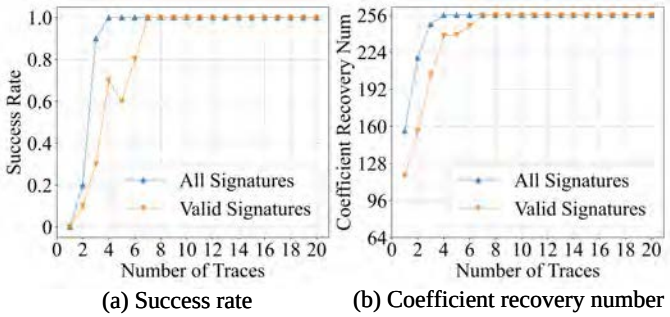


(a) Success rate (b) Coefficient recovery number

Fig. 16. BP Results With Optimization O0.

D. Comparison With Existing Works

As shown in Table IV, a comparison is provided between this work and existing studies. Existing research can be broadly classified into three categories. [18]–[20] target operations directly involving the private key, such as its unpacking process or NTT transformations. Benefiting from the high classification accuracy provided by DL, these approaches have achieved single-trace key recovery. However, in real-world deployment scenarios, these private key operations may occur only once at device startup or may never be executed at all (e.g., if the private key is stored directly in the NTT domain). This makes such methods difficult to apply in practical settings.



(a) Success rate (b) Coefficient recovery number

Fig. 17. BP Results With Optimization O3.

[13]–[15] obtain numerical hints about the private key and use ILP for key recovery. The best implementation has also achieved single trace key recovery. However, due to their limited error tolerance, they lose useful probabilistic hints when the correct classification does not have the highest probability. As a result, their effectiveness is also heavily dependent on the classification accuracy of DL models. Furthermore, their ability to simultaneously utilize both valid and rejected signatures for key recovery remains to be fully validated.

[16], [17] attempt to exploit rejected signatures to recover the private key. However, their methods of leveraging rejected signatures are inefficient, resulting in worse performance than attacks that use only valid signatures. This inefficiency limits their ability to effectively integrate hints from both valid and rejected signatures for key recovery.

This paper proposes a method for private key recovery that fully leverages side-channel leakages from both valid and rejected signatures. Unlike prior works such as [18]–[20], which are limited to specific scenarios, our approach targets the signature and corresponding computations, making it more broadly applicable across different scenarios. Benefiting from the high error tolerance of BP, our method does not rely on the high classification accuracy offered by DL, yet still achieves single-trace key recovery. Compared to existing studies [16], [17], which also attempt to utilize rejected signatures, our method extracts higher-quality hints from rejected signatures. This leads to a significant reduction in the number of traces required for key recovery when rejected signatures are used. Furthermore, our work addresses several missing details in applying BP to Dilithium [17] and provides practical BP experimental results. Additionally, we propose a more generalized factor graph structure to maximize the utilization of all side-channel leakages within a single signing attempt, and optimize BP computation to enhance practical applicability. To facilitate

TABLE IV
COMPARISON WITH EXISTING WORKS

Work	Applicable Scenarios	Experimental Type	Main Method	Utilizing Rejected Signatures	Error Tolerance	Number of Traces Required*
[17]	General	Simulation	TA, BP	Inefficient	High	10^5
[16]	General	Practical	DL, ILP	Inefficient	Medium	10^6
[13]	General	Practical	DL, ILP	No	Medium	$1(2)$
[14]	General	Practical	DL, ILP	No	Medium	10^5
[15]	General	Practical	TA, LSM	No	Medium	10^6
[18]	Constrained	Practical	DL	No	Low	$1(100)$
[19]	Constrained	Practical	DL	No	Low	$1(1)$
[20]	Constrained	Practical	DL	No	Low	$1(1)$
This Work	General	Practical	TA, BP	**Efficient**	**High**	$\mathbf{1(4)}$

* The value outside the parentheses represents the minimum required traces, while the value inside the parentheses indicates the number of traces required for a 100% success rate. For larger results, we only present their approximate order of magnitude.

further research, we open-source our experimental code and dataset, allowing future studies to build upon our findings.

VI. DISCUSSION ON MASKED IMPLEMENTATION

To the best of our knowledge, the state-of-the-art masking countermeasures for Dilithium do not protect rejected signatures [28], [29], meaning that the rejected signature c remains recoverable. Although some studies have proposed masked implementations of the NTT [30], the rejected signature c can still be recovered by analyzing the signature generation process [31], specifically Line 6 of Algorithm 2.

While y and s_1 are masked, the attacker can perform SCA separately on each share of y and cs_1. By summing the recovered shares, the attacker can eliminate the masking effect and continue utilizing the factor graph attack proposed in this paper. However, under masked implementations, the bit-length reduction strategy for y and cs_1 proposed in this work would no longer be sensible during the SCA phase. Instead, the attacker must recover all bits of each share of y and cs_1, significantly increasing the overhead of the SCA phase.

Despite this, in the BP phase, the attacker can still leverage the bit-length reduction strategy proposed in this paper. The BP algorithm can be executed using only a small portion of the recovered bits of y and cs_1, reducing computational complexity.

Overall, masking countermeasures increase the difficulty of SCA, but the BP-based key recovery process remains unchanged.

VII. CONCLUSION AND COUNTERMEASURE

This paper proposes an attack method against Dilithium, which simultaneously leverages hints from both valid and rejected signatures to recover the private key. We conducted experiments on a Cortex-M4 core chip, and the results demonstrate that utilizing rejected signature information can reduce the required number of traces by at least 42% for a full key recovery.

Our study highlights the necessity of protecting rejected signatures, particularly c, as their leakage significantly aids key recovery. We strongly recommend implementing masking for rejected signatures or introducing shuffle-based countermeasures during the signature generation process to mitigate this attack.

REFERENCES

[1] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, ser. Lecture Notes in Computer Science, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 388–397. [Online]. Available: https://doi.org/10.1007/3-540-48405-1_25

[2] 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.*, p. 821, 2018. [Online]. Available: https://eprint.iacr.org/2018/821

[3] Z. Qiao, Y. Liu, Y. Zhou, M. Shao, and S. Sun, "When NTT meets SIS: efficient side-channel attacks on dilithium and kyber," *IACR Cryptol. ePrint Arch.*, p. 1866, 2023. [Online]. Available: https://eprint.iacr.org/2023/1866

[4] H. M. Steffen, G. Land, L. J. Kogelheide, and T. Güneysu, "Breaking and protecting the crystal: Side-channel analysis of dilithium in hardware," in *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023, College Park, MD, USA, August 16-18, 2023, Proceedings*, ser. Lecture Notes in Computer Science, T. Johansson and D. Smith-Tone, Eds., vol. 14154. Springer, 2023, pp. 688–711. [Online]. Available: https://doi.org/10.1007/978-3-031-40003-2_25

[5] H. Wang, Y. Gao, Y. Liu, Q. Zhang, and Y. Zhou, "In-depth correlation power analysis attacks on a hardware implementation of crystals-dilithium," *Cybersecur.*, vol. 7, no. 1, p. 21, 2024. [Online]. Available: https://doi.org/10.1186/s42400-024-00209-9

[6] Y. Liu, Y. Liu, Y. Zhou, Y. Gao, Z. Qiao, and H. Wang, "A novel power analysis attack against crystals-dilithium implementation," in *IEEE European Test Symposium, ETS 2024, The Hague, Netherlands, May 20-24, 2024*. IEEE, 2024, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ETS61313.2024.10567325

[7] T. Tosun and E. Savas, "Zero-value filtering for accelerating non-profiled side-channel attack on incomplete ntt-based implementations of lattice-based cryptography," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 3353–3365, 2024. [Online]. Available: https://doi.org/10.1109/TIFS.2024.3359890

[8] T. Tosun, A. Moradi, and E. Savas, "Exploiting the central reduction in lattice-based cryptography," *IEEE Access*, vol. 12, pp. 166 814–166 833, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3494593

[9] A. P. Fournaris, C. Dimopoulos, and O. G. Koufopavlou, "Profiling dilithium digital signature traces for correlation differential side channel attacks," in *Embedded Computer Systems: Architectures, Modeling, and Simulation - 20th International Conference, SAMOS 2020, Samos, Greece, July 5-9, 2020, Proceedings*, ser. Lecture Notes in Computer Science, A. Orailoglu, M. Jung, and M. Reichenbach, Eds., vol. 12471. Springer, 2020, pp. 281–294. [Online]. Available: https://doi.org/10.1007/978-3-030-60939-9_19

[10] 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 *39th IEEE International Conference on Computer Design, ICCD 2021, Storrs, CT, USA, October 24-27, 2021*. IEEE, 2021, pp. 583–590. [Online]. Available: https://doi.org/10.1109/ICCD53106.2021.00094

[11] Z. Qiao, Y. Liu, Y. Zhou, J. Ming, C. Jin, and H. Li, "Practical public template attack attacks on crystals-dilithium with randomness leakages," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 1–14, 2023. [Online]. Available: https://doi.org/10.1109/TIFS.2022.3215913

[12] Y. Liu, Y. Zhou, S. Sun, T. Wang, R. Zhang, and J. Ming, "On the security of lattice-based fiat-shamir signatures in the presence of randomness leakage," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1868–1879, 2021. [Online]. Available: https://doi.org/10.1109/TIFS.2020.3045904

[13] Z. Qiao, Y. Liu, Y. Zhou, Y. Zhao, and S. Chen, "Single trace is all it takes: Efficient side-channel attack on dilithium," *IACR Cryptol. ePrint Arch.*, p. 512, 2024. [Online]. Available: https://eprint.iacr.org/2024/512

[14] S. Marzougui, V. Ulitzsch, M. Tibouchi, and J. Seifert, "Profiling side-channel attacks on dilithium: A small bit-fiddling leak breaks it all," *IACR Cryptol. ePrint Arch.*, p. 106, 2022. [Online]. Available: https://eprint.iacr.org/2022/106

[15] 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 Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2023, no. 4, pp. 188–210, 2023. [Online]. Available: https://doi.org/10.46586/tches.v2023.i4.188-210

[16] Y. Zhou, W. Wang, Y. Sun, and Y. Yu, "Rejected challenges pose new challenges: Key recovery of CRYSTALS-dilithium via side-channel attacks," Cryptology ePrint Archive, Paper 2025/214, 2025. [Online]. Available: https://eprint.iacr.org/2025/214

[17] O. Bronchain, M. Azouaoui, M. ElGhamrawy, J. Renes, and T. Schneider, "Exploiting small-norm polynomial multiplication with physical attacks application to crystals-dilithium," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2024, no. 2, pp. 359–383, 2024. [Online]. Available: https://doi.org/10.46586/tches.v2024.i2.359-383

[18] R. Wang, K. Ngo, J. Gärtner, and E. Dubrova, "Single-trace side-channel attacks on crystals-dilithium: Myth or reality?" *IACR Cryptol. ePrint Arch.*, p. 1931, 2023. [Online]. Available: https://eprint.iacr.org/2023/1931

[19] J. Han, T. Lee, J. Kwon, J. Lee, I. Kim, J. Cho, D. Han, and B. 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. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3135600

[20] I. Kim, T. Lee, J. Han, B. Sim, and D. Han, "Novel single-trace ML profiling attacks on NIST 3 round candidate dilithium," *IACR Cryptol. ePrint Arch.*, p. 1383, 2020. [Online]. Available: https://eprint.iacr.org/2020/1383

[21] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, D. Stehlé, and S. Bai, "Crystals-dilithium," *Algorithm Specifications and Supporting Documentation*, 2020.

[22] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, ser. Lecture Notes in Computer Science, B. S. K. Jr., Ç. K. Koç, and C. Paar, Eds., vol. 2523. Springer, 2002, pp. 13–28. [Online]. Available: https://doi.org/10.1007/3-540-36400-5_3

[23] A. Satriawan and R. Mareta, "A complete beginner guide to the number theoretic transform (NTT)," *IACR Cryptol. ePrint Arch.*, p. 585, 2024. [Online]. Available: https://eprint.iacr.org/2024/585

[24] N. Veyrat-Charvillon, B. Gérard, and F. Standaert, "Soft analytical side-channel attacks," in *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, ser. Lecture Notes in Computer Science, P. Sarkar and T. Iwata, Eds., vol. 8873. Springer, 2014, pp. 282–296. [Online]. Available: https://doi.org/10.1007/978-3-662-45611-8_15

[25] C. Knoll, "Understanding the behavior of belief propagation," *CoRR*, vol. abs/2209.05464, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2209.05464

[26] R. Primas, P. Pessl, and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," in *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, ser. Lecture Notes in Computer Science, W. Fischer and N. Homma, Eds., vol. 10529. Springer, 2017, pp. 513–533. [Online]. Available: https://doi.org/10.1007/978-3-319-66787-4_25

[27] P. Pessl and R. Primas, "More practical single-trace attacks on the number theoretic transform," in *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*, ser. Lecture Notes in Computer Science, P. Schwabe and N. Thériault, Eds., vol. 11774. Springer, 2019, pp. 130–149. [Online]. Available: https://doi.org/10.1007/978-3-030-30530-7_7

[28] M. Azouaoui, O. Bronchain, G. Cassiers, C. Hoffmann, Y. Kuzovkova, J. Renes, T. Schneider, M. Schönauer, F. Standaert, and C. van Vredendaal, "Protecting dilithium against leakage revisited sensitivity analysis and improved implementations," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2023, no. 4, pp. 58–79, 2023. [Online]. Available: https://doi.org/10.46586/tches.v2023.i4.58-79

[29] J. Coron, F. Gérard, M. Trannoy, and R. Zeitoun, "Improved gadgets for the high-order masking of dilithium," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2023, no. 4, pp. 110–145, 2023. [Online]. Available: https://doi.org/10.46586/tches.v2023.i4.110-145

[30] R. C. Rodriguez, E. Valea, F. Bruguier, and P. Benoit, "Hardware implementation and security analysis of local-masked NTT for crystals-kyber," *IACR Cryptol. ePrint Arch.*, p. 1194, 2024. [Online]. Available: https://eprint.iacr.org/2024/1194

[31] E. Karabulut, E. Alkim, and A. Aysu, "Single-trace side-channel attacks on ω-small polynomial sampling: With applications to ntru, NTRU prime, and CRYSTALS-DILITHIUM," in *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2021, Tysons Corner, VA, USA, December 12-15, 2021*. IEEE, 2021, pp. 35–45. [Online]. Available: https://doi.org/10.1109/HOST49136.2021.9702284