Rejected Challenges Pose New Challenges: Key Recovery of CRYSTALS-Dilithium via Side-Channel Attacks

Yuanyuan Zhou[1], Weijia Wang[2,3], Yiteng Sun[2,3] and Yu Yu[4,5]

[1] SGS Brightsight BV, Delft, The Netherlands zhou.yuanyuan@gmail.com
[2] Shandong University, School of Cyber Science and Technology, Qingdao, China
wjwang@sdu.edu.cn,sunyiteng@mail.sdu.edu.cn
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao, China
[4] Shanghai Jiao Tong University, Shanghai 200240, China yuyu@yuyu.hk
[5] Shanghai Qi Zhi Institute, 701 Yunjin Road, Shanghai 200232, China

Abstract. Rejection sampling is a crucial security mechanism in lattice-based signature schemes that follow the Fiat-Shamir with aborts paradigm, such as ML-DSA/CRYSTALS-Dilithium. This technique transforms secret-dependent signature samples into ones that are statistically close to a secret-independent distribution (in the random oracle model). While many side-channel attacks have directly targeted sensitive data such as nonces, secret keys, and decomposed commitments, fewer studies have explored the potential leakage associated with rejection sampling. Notably, Karabulut et al. showed that leakage from rejected challenges can undermine, but not entirely break, the security of the Dilithium scheme.

Motivated by the above, we convert the problem of key recovery (from the leakage of rejection sampling) to an integer linear programming problem (ILP), where rejected responses of unique Hamming weights set upper/lower constraints of the product between the challenge and the private key. We formally study the worst-case complexity of the problem as well as empirically confirm the practicality of the rejected challenge attack. For all three security levels of Dilithium-2/3/5, our attack recovers the private key in seconds or minutes with a 100% Success Rate (SR).

Our attack leverages knowledge of the rejected challenge and response, and thus we propose methods to extract this information by exploiting side-channel leakage from Number Theoretic Transform (NTT) operations. We demonstrate the practicality of this rejected challenge attack by using real side-channel leakage on a Dilithium-2 implementation running on an ARM Cortex-M4 microcontroller. To the best of our knowledge, it is the first efficient side-channel key recovery attack on ML-DSA/Dilithium that targets the rejection sampling procedure. Furthermore, we discuss some countermeasures to mitigate this security issue.

Keywords: Dilithium · ML-DSA · Side-channel attacks · Rejection sampling · Integer Linear Programming.

1 Introduction

1.1 Side-channel attacks on Dilithium/ML-DSA

Side-channel attacks (SCA) [Koc96, KJJ99] exploit physical information leaked during the executions of cryptographic algorithm implementations. Unlike classical cryptanalytic techniques, which focus on the mathematical structure of the algorithms, SCA takes

advantage of unintended physical emissions such as power consumption, electromagnetic radiation, or timing information. Since most cryptographic algorithms were not designed with physical information leakage in mind, SCA can be more effective and powerful. For example, it has shown that SCA can successfully compromise the implementations of almost all well-known cryptographic algorithms.

The National Institute of Standards and Technology (NIST) has recently standardized CRYSTALS-Dilithium (or Dilithium for short) [DKL+18] and has named it the Module-Lattice-Based Digital Signature Algorithm (ML-DSA) [NIS24]. Its security has been thoroughly studied from reductionist and cryptanalytic perspectives [DKL+18, KLS18, LZ19, DFMS19, BDK+20, DFPS23, BBD+23, JMW24], particularly in the (classical or quantum) Random Oracle Model (ROM or QROM). As expected, Dilithium is vulnerable to side-channel attacks, given additional information about the secret signing key.

Algorithm 1 Dilithium Sign [DKL+18] (Simplified)

Input: sk=$(\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$, message M, $pk = (\rho, \mathbf{t}_1)$
Output: $(\mathbf{z}, c)$

1: $\mathbf{z} \leftarrow \perp$
2: **while** $\mathbf{z} = \perp$ **do**
3: $\quad \mathbf{y} \xleftarrow{\$} \widetilde{S}^{\ell}_{\gamma_1}$
4: $\quad \mathbf{w} \leftarrow \mathbf{A}\mathbf{y}$
5: $\quad \mathbf{w}_1 \leftarrow \mathsf{HighBits}_q(\mathbf{w})$ ▷ $\mathbf{w} = \mathbf{w}_1 \cdot 2\gamma_2 + \mathbf{w}_0$
6: $\quad c \leftarrow \mathsf{H}(\mu \| \mathsf{w1Encode}(\mathbf{w}_1))$
7: $\quad \mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_1$ ▷ Compute $c\mathbf{s}_1$ as $\mathsf{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{s}_1})$
8: $\quad$ **if** $\mathrm{CheckBound}_1(\mathbf{z}) ==$ False or $\mathrm{CheckBound}_2(\mathbf{w} - c\mathbf{s}_2) ==$ False **then**
9: $\quad\quad \mathbf{z} \leftarrow \perp$
10: **return** $\sigma = (\mathbf{z}, c)$

Algorithm 1 provides an overview of the Dilithium signing process, where the subroutines $\mathrm{CheckBound}_1$ and $\mathrm{CheckBound}_2$ verify whether or not $\mathbf{z}$ and $\mathbf{w} - c\mathbf{s}_2$ meet the respective constraints and output the corresponding Boolean values. The 'keep sampling until the constraints are met' process, known as rejection sampling, is essential for ensuring the correctness and security of the scheme. Most SCAs on Dilithium focus on the leakage of its signing process. Table 1 provides a summary of existing SCAs on Dilithium, categorized by attack type and the intermediate values they target.

Ravi et al. first published a SCA attack on Dilithium [RJH+18]. They specifically targeted the leakage during the point-wise multiplication of $c\mathbf{s}_1$ and recovered the secret $\mathbf{s}_1$ using differential power analysis. They further demonstrated that recovering $\mathbf{s}_1$ enables the forgery of signatures. This approach of exploiting point-wise multiplication has since been extended to other components, such as $c\mathbf{s}_2$ and $c\mathbf{t}_0$, and has been further optimized in subsequent literature [CKA+21, SLKG23, WGL+24, LLZ+24, QLZ+23b, TS24, TMS24].

Another attack strategy directly recovers the secrets $\mathbf{s}_1$ and $\mathbf{s}_2$. Kim et al. demonstrated that they could recover the secret by constructing a deep learning-based template [KLH+20]. However, in many cases, side-channel analysis only provides partial information about the secret due to noise, making direct recovery of the secret challenging. A more practical approach is to recover sensitive intermediate values being more susceptible to side-channel leakage first and then use multiple such intermediates to reconstruct the secret key. For example, Marzougui et al. [UMTS22] proposed a method to distinguish signatures in which one coefficient in $\mathbf{y}$ is 0 based on side-channel leakage. By obtaining multiple such signatures, they were able to recover the secret. Qiao et al. demonstrated that one bit in coefficients of $\mathbf{y}$ is sufficient for the key recovery [QLZ+23a], at the cost of using many more signatures. Berzati et al. [BVC+23] extended this strategy by focusing on the leakage of $\mathbf{w}_0$.

Table 1: Summary of the types and targets of side-channel attacks on Dilithium, where NTT refers to the Number Theoretic Transform, and SPA/CPA refers to the Simple/Correlation Power Analysis, respectively.

	Target	Attack Type
This work	c & $\mathbf{z}$	Profiled SPA
[BAE+24]	$\mathbf{y}$ & $c\mathbf{s}_1$	Simulation
[BVC+23]	$\mathbf{w}_0$	Profiled SPA
[LZS+21, UMTS22]	$\mathbf{y}$	Profiled SPA
[QLZ+23a, QLZ+24]	$\mathbf{y}$ & $c\mathbf{s}_1$	Profiled SPA
[RJH+18]	$c\mathbf{s}_1$	CPA
[KLH+20, CKA+21, QLZ+23b, SLKG23, WGL+24, LLZ+24, TS24, TMS24]	$\mathsf{NTT}(c\mathbf{s}_1), \mathsf{NTT}(c\mathbf{s}_2)$	CPA
[WNGD23]	$\mathbf{s}_1$, $\mathbf{s}_2$	Profiled SPA

However, in the literature, little attention has been paid to the leakage of public data, such as the rejected challenge c. It is worth noting that in [KAA21] Karabulut et al. targeted challenge c and recovered the signs of its nonzero coefficients. They concluded that this did not compromise the overall security of Dilithium but resulted in a reduction of τ bits of entropy in c, where τ is the number of nonzero coefficients in c. Moreover, Bronchain et al. explored attacks targeting the rejected response $\mathbf{z}$ and $\mathbf{y}$ in [BAE+24]. Based on simulation experiments, they show that key recovery of $\mathbf{s}_1$ is feasible assuming the knowledge of 32 Hamming weight (HW) classes of all coefficients of $\mathbf{y}$ and the index of the rejected coefficient of $\mathbf{z}$ in an Early-Abort setting. It requires about 10^7 rejected $\mathbf{z}$ values to recover $\mathbf{s}_1$ for Dilithium-2 with a relatively high signal-to-noise ratio (SNR) of 1 and an even higher SNR of 10 for Dilithium-3/5.

In the Dilithium scheme, challenge c is generally regarded as public randomness (produced by a random oracle) and is thus considered benign from a leakage perspective. As a result, no specific side-channel countermeasures have been recommended [ABC+23], nor have any been implemented in open-source libraries such as Dilithium reference implementation [KPR+], Botan library's [Bot24] Dilithium implementation, or the ML-DSA implementation in the OQS (Open Quantum Safe) library (a provider of the widely used OpenSSL library) [Lib24]. Besides, it is well-known that the rejected response $\mathbf{z}$ can exhibit a bias related to the private key $\mathbf{s}_1$, as discussed in the specification document [DKL+18, NIS24]. It is recommended in [ABC+23] to keep $\mathbf{z}$ protected until it passes the bound check to mitigate its potential SCA leakage. However, no published SCA attacks have specifically targeted $\mathbf{z}$. Furthermore, the proposed masking countermeasure [ABC+23, CGL+24] to protect $\mathbf{z}$ is rather costly, with first-order masking that slows down operations by approximately 40 times according to the state-of-the-art results [CGL+24], which is particularly challenging for embedded devices. In contrast, developers are well aware that they must adequately protect other targeted variables $\mathbf{y}, \mathbf{w}_0, \mathbf{s}_1, \mathbf{s}_2, c\mathbf{s}_1, c\mathbf{s}_2, \mathsf{NTT}(c\mathbf{s}_1)$, and $\mathsf{NTT}(c\mathbf{s}_2)$ in the literature as shown in Table 1 against SCA attacks [DKL+18, NIS24, ABC+23, RCDB24].

1.2 Our contribution

In this work, we aim to answer the question: *how and to what extent can we recover the Dilithium private key by combining the implementation leakages of rejected challenge c and its corresponding response $\mathbf{z}$ theoretically and empirically?* Our contributions in this context are summarized as follows.

First, we formalize the problem of key recovery from the leakages of rejected challenge and response. Specifically, we observe that rejections are caused by out-of-bound coefficients in $\mathbf{z}$. Therefore, we use rejected $\mathbf{z}$ (especially those with unique Hamming weights favored by side-channel attacks) and the corresponding challenge to construct a system of inequalities involving the private key $\mathbf{s}_1$. We then study the worst-case complexity of the key recovery problem given a linear number of these inequalities (signature generations). We also

empirically determine the required number of signatures and the bounds for all three NIST security levels 2, 3, and 5 for the proposed rejected challenge attack. We show that the 3×10^6 (resp., 2.3×10^7 and 1.3×10^7) signature generation trials are sufficient for our attack to recover $\mathbf{s}_1$ in seconds or minutes for security level 2 (resp., 3 and 5) with a 100% SR.

Second, since the rejected challenge attack requires the knowledge of the rejected challenge c, we propose methods to extract those information. Our attack on rejected challenge focuses on NTT operations that are particularly vulnerable to side-channel attacks due to the intensive use of sensitive values. We experimentally verify the feasibility and validity of our rejected challenge key attack using deep learning-based side-channel leakage from a typical real-world target, namely, a reference Dilithium-2 implementation on a 32-bit STM32F microcontroller. The experimental results demonstrate the successful recovery of both the rejected challenge and the response with a 100% accuracy rate.

As a takeaway message, our work highlights that existing countermeasures for side-channel leakages in other operations are inadequate. We must focus on rejection sampling procedures as well. We discuss some potential countermeasures, such as masking and hiding techniques.

1.3 Related works

Azevedo-Oliveira et al. recently proposed a fault attack against Dilithium [AOVCG25] that builds on the premise that the bound check could be skipped (by, e.g., injecting faults), enabling the recovery of the secret with a sufficient number of samples. Despite the shared similarity with this concurrent work, our attack is non-invasive and exploits only the power leakages corresponding to the challenge and response. Furthermore, we empirically validate the practicality of our attack. Given the ubiquity of power leakage, we believe our attack underscores the critical importance of thoroughly protecting the challenge and response in the implementation of Dilithium.

1.4 Organization of the paper

The rest of this paper is structured as follows. Section 2 introduces background information about rejection sampling and Dilithium. Section 3 formulates the rejected challenge key recovery attack on Dilithium, and provides a formal analysis. Section 4 experimentally verifies the effectiveness and efficiency of this approach. Finally, Section 5 demonstrates the practicality of the rejected challenge attack by leveraging side-channel leakages from a typical real-world implementation of Dilithium.

2 Preliminaries

2.1 Notations

We adhere to the conventions defined in the NIST FIPS 204 [NIS24] wherever applicable. Let $[a, b]$ denote the intervals of integers $\{a, a+1, \ldots, b\}$ given $a, b \in \mathbb{Z}$. We denote by $\mathcal{N}(0, \sigma^2)$ the normal distribution with mean 0 and standard deviation σ. We denote polynomials by regular lowercase letters (e.g., c), vectors of polynomials by lowercase letters in bold (e.g., $\mathbf{z}$) and matrices of polynomials by uppercase letters in bold (e.g., $\mathbf{A}$). We define the polynomial ring $R_q \stackrel{\text{def}}{=} \mathbb{Z}_q[X]/\langle X^{n+1} \rangle$ with $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, in which, q is an odd prime number and $n \in \mathbb{N}$, they are set to 8380417 and 256 respectively in the ML-DSA/Dilithium scheme. We represent a polynomial $c = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1} \in R_q$ as a vector $(c_0, \ldots, c_{n-1}) \in \mathbb{Z}_q^n$. Given $r \in \mathbb{Z}_q$, $r \mod^{\pm} q$ is defined to be the unique element $r' \in \mathbb{Z}$ such that $-(q-1)/2 \leq r' \leq (q-1)/2$ and $r' \equiv r \mod q$. For any $c \in R_q$,

we define $|c|_i \stackrel{\text{def}}{=} |c_i \mod^{\pm} q|$ for all $i \in \{0, 1, \ldots, n-1\}$ and $\|c\|_\infty \stackrel{\text{def}}{=} \max_i |c|_i$. Similarly, we define $\|\mathbf{z}\|_\infty \stackrel{\text{def}}{=} \max_{i\in[1,\ell]} \|\mathbf{z}_i\|_\infty$ for any $\mathbf{z} \in R_q^\ell$. A capital alphabet in italics (e.g., S) refers to a finite nonempty set, $s \leftarrow S$ denotes an assignment of an element s from S. Let S_η denote the set $\{s \in R_q \mid \|s\|_\infty \leq \eta\}$ with $\eta \in \mathbb{N}$. Similarly, we denote B_τ as the set $\{b \in R_q \mid \|b\|_\infty = 1\}$ and b has exactly τ nonzero coefficients with $\tau \in \mathbb{N}$, so, $|B_\tau| = 2^\tau \binom{n}{\tau}$. By 'log' we denote the base 2 logarithm.

2.2 Rejection sampling in Lyubashevsky's signature scheme

Schnorr [Sch89] employed the Fiat-Shamir transformation [FS86] to transform a constructed identification scheme into a signature scheme. Lyubashevsky introduced lattice-based signature schemes in [Lyu09, Lyu12] by adapting Schnorr's approach and incorporating aborts during signature generation. This "Fiat-Shamir with aborts" paradigm aims to abort/discard signatures that could potentially leak secret key information. This rejection sampling [Lyu09] is a cornerstone used to construct many lattice-based signature schemes [Lyu12, BG14], including Dilithium/ML-DSA [DKL+18, NIS24]. It produces a distribution that is statistically independent of the secret key in the random oracle model, and thus eliminates the leakage of secret key information.

2.3 Dilithium

Algorithm 2 KeyGen

1: $\zeta \leftarrow \{0,1\}^{256}$
2: $(\rho, \rho', K) \in \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} \leftarrow \mathsf{H}(\zeta, 1024)$ ▷ H is instantiated as SHAKE-256 throughout
3: $\mathbf{A} \in R_q^{k\times\ell} \leftarrow \mathsf{ExpandA}(\rho)$ ▷ $\mathbf{A}$ is generated and stored in NTT representation as $\hat{\mathbf{A}}$
4: $(\mathbf{s}_1, \mathbf{s}_2) \in S_\eta^\ell \times S_\eta^k \leftarrow \mathsf{ExpandS}(\rho')$
5: $\mathbf{t} \leftarrow \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ ▷ Compute $\mathbf{A}\mathbf{s}_1$ as $\mathsf{NTT}^{-1}(\hat{\mathbf{A}} \cdot \mathsf{NTT}(\mathbf{s}_1))$
6: $(\mathbf{t}_1, \mathbf{t}_0) \leftarrow \mathsf{Power2Round}_q(\mathbf{t}, d)$
7: $tr \in \{0,1\}^{256} \leftarrow \mathsf{H}(\rho \| \mathbf{t}_1, 256)$
8: **return** $(\mathrm{pk} = (\rho, \mathbf{t}_1), \mathrm{sk} = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0))$

Dilithium includes three basic procedures: key generation, signature generation, and signature verification. In this subsection, we provide a brief overview of these processes, and refer the readers to official documentation for more details about the scheme. Algorithm 2 illustrates the key generation process of Dilithium. One should pay particular attention to the secret polynomials $\mathbf{s}_1$ and $\mathbf{s}_2$. These polynomials have a small norm such that $||\mathbf{s}_1||_\infty \leq \eta$ and $||\mathbf{s}_2||_\infty \leq \eta$. All coefficients in these polynomials are considered uniformly and independently drawn (using $\mathsf{ExpandS}$).

The signing process depicted in Algorithm 3 generates a masking vector of polynomials $\mathbf{y}$ with coefficients bounded by γ_1. The signer then computes $\mathbf{w} = \mathbf{A}\mathbf{y}$ and extracts $\mathbf{w}_1$, representing the "high-order" bits of the coefficients in this vector. Specifically, each coefficient in $\mathbf{w}$ can be decomposed canonically as $\mathbf{w} = \mathbf{w}_1 \cdot 2\gamma_2 + \mathbf{w}_0$, where $\|\mathbf{w}_0\|_\infty \leq \gamma_2$. Subsequently, a challenge c is generated as the hash of the message and $\mathbf{w}_1$. The resulting c is a polynomial in R_q with exactly τ coefficients of ± 1's, and the rest being 0. This distribution is designed in such a way that c has a small norm while still coming from a domain with a size greater than 2^{256}. Finally, the signer computes $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$ as the potential signature.

Rejection sampling is used to eliminate the dependency of $\mathbf{z}$ on the secret key. The parameter β is the maximum absolute value of any coefficient in $c\mathbf{s}_i$. Given that c has τ coefficients of ± 1's and the maximum absolute value of any coefficient in $\mathbf{s}_i$ is η, we have $\|c\mathbf{s}_i\|_\infty \leq \beta = \tau\eta$. If the absolute value of any coefficient of $\mathbf{z}$ exceeds $\gamma_1 - \beta$, the signing

Algorithm 3 $\mathsf{Sign}(sk, M)$

$\mathbf{A} \in R_q^{k\times\ell} \leftarrow \mathsf{ExpandA}(\rho)$ ▷ $\mathbf{A}$ is generated and stored in NTT representation as $\hat{\mathbf{A}}$
$\mu \in \{0,1\}^{512} \leftarrow \mathsf{H}(tr\|M, 512)$
$\kappa \leftarrow 0$, $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
$\rho' \in \{0,1\}^{512} \leftarrow \mathsf{H}(K\|\mu, 512)$ ▷ or $\rho' \leftarrow \{0,1\}^{512}$ for randomized signing
while $(\mathbf{z}, \mathbf{h}) = \perp$ **do** ▷ Pre-compute $\hat{\mathbf{s}}_1 \stackrel{\text{def}}{=} \mathsf{NTT}(\mathbf{s}_1), \hat{\mathbf{s}}_2 \stackrel{\text{def}}{=} \mathsf{NTT}(\mathbf{s}_2)$, and $\hat{\mathbf{t}}_0 \stackrel{\text{def}}{=} \mathsf{NTT}(\mathbf{t}_0)$
  $\mathbf{y} \in \widetilde{S}_{\gamma_1}^{\ell} \leftarrow \mathsf{ExpandMask}(\rho', \kappa)$
  $\mathbf{w} \leftarrow \mathbf{A}\mathbf{y}$ ▷ or $\mathbf{w} \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{A}} \cdot \mathsf{NTT}(\mathbf{y}))$
  $\mathbf{w}_1 \leftarrow \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
  $\widetilde{c} \in \{0,1\}^{256} \leftarrow \mathsf{H}(\mu\|\mathbf{w}_1, 256)$
  $c \in B_\tau \leftarrow \mathsf{SampleInBall}(\widetilde{c})$ ▷ Store c in NTT representation as $\hat{c} \stackrel{\text{def}}{=} \mathsf{NTT}(c)$
  $\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_1$ ▷ Compute $c\mathbf{s}_1$ as $\mathsf{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{s}_1})$
  $\mathbf{r}_0 \leftarrow \mathsf{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$ ▷ Compute $c\mathbf{s}_2$ as $\mathsf{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{s}_2})$
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then**
    $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
  else
    $\mathbf{h} \leftarrow \mathsf{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$ ▷ Compute $c\mathbf{t}_0$ as $\mathsf{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{t}_0})$
    if $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$ or the number of 1's in $\mathbf{h}$ is greater than ω **then**
      $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
  $\kappa \leftarrow \kappa + \ell$
return $\sigma = (\widetilde{c}, \mathbf{z}, \mathbf{h})$

procedure is rejected and is restarted. Likewise, if the absolute value of any coefficient of "low-order" bits of $\mathbf{A}\mathbf{z} - c\mathbf{t}$ exceeds $\gamma_2 - \beta$, the procedure is also restarted. The first check is necessary for security, and the second is essential for security and correctness. The signing repeats until both conditions are satisfied.

Algorithm 4 $\mathsf{Verify}(pk, M, \sigma)$

$\mathbf{A} \in R_q^{k\times\ell} \leftarrow \mathsf{ExpandA}(\rho)$ ▷ $\mathbf{A}$ is generated and stored in NTT representation as $\hat{\mathbf{A}}$
$\mu \in \{0,1\}^{512} \leftarrow \mathsf{H}(\mathsf{H}(\rho\|\mathbf{t}_1)\|M, 512)$
$c \leftarrow \mathsf{SampleInBall}(\tilde{c})$
$\mathbf{w}_1' \leftarrow \mathsf{UseHint}_q(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)$ ▷ Compute as $\mathsf{NTT}^{-1}(\hat{\mathbf{A}} \cdot \mathsf{NTT}(\mathbf{z}) - \mathsf{NTT}(c) \cdot \mathsf{NTT}(\mathbf{t}_1 \cdot 2^d))$
return $[\|\mathbf{z}\|_\infty < \gamma_1 - \beta]$ and $[\tilde{c} = \mathsf{H}(\mu\|\mathbf{w}_1')]$ and [# of 1's in $\mathbf{h}$ is $\leq \omega$]

Dilithium's verification process is depicted in Algorithm 4. It computes $\mathbf{w}_1'$, which represents the "high-order" bits of $\mathbf{A}\mathbf{z} - c\mathbf{t}$. The verifier accepts the signature if all coefficients of $\mathbf{z}$ fall within the range of $[-\gamma_1 + \beta + 1, \gamma_1 - \beta - 1]$ and if c matches the hash of the message and $\mathbf{w}_1'$. Since $\mathbf{A}\mathbf{z} - c\mathbf{t} = \mathbf{A}\mathbf{y} - c\mathbf{s}_2$, it suffices to show that $\mathsf{HighBits}_q(\mathbf{A}\mathbf{y}, 2\gamma_2) = \mathsf{HighBits}_q(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)$. The reason is that a valid signature ensures $\|\mathsf{LowBits}_q(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)\|_\infty < \gamma_2 - \beta$. Given that the coefficients of $c\mathbf{s}_2$ are upper-bounded by β, adding $c\mathbf{s}_2$ cannot cause any low-order coefficient to exceed γ_2 by introducing carry bits. Thus, the above equations and inequalities hold, and the signature is verified correctly.

Dilithium has three security levels, namely, Dilithium-2/3/5 [BDK+20], which correspond to the ML-DSA security levels ML-DSA-44/65/87 [NIS24]. Table 2 summarizes the parameter choices related to this attack for each security level.

2.4 Integer Linear Programming (ILP)

Following Lenstra's work [Jr.83], we consider a variant of the ILP problem as follows: for integral matrix $\mathbf{A} \in \mathbb{Z}^{m\times n}$ and vector $\mathbf{b} \in \mathbb{Z}^m$, find a vector $\mathbf{x} \in \mathbb{Z}^n$ that satisfies the system of m inequalities $\mathbf{A}\mathbf{x} \leq \mathbf{b}$.

Table 2: Relevant ML-DSA/Dilithium parameter choices [BDK+20, NIS24]

Scheme/NIST security level	(k,ℓ)	η	τ	$\beta(=\eta\cdot\tau)$	γ_1	#Reps
ML-DSA-44 (L2)	(4,4)	2	39	78	2^{17}	4.25
ML-DSA-65 (L3)	(6,5)	4	49	196	2^{19}	5.1
ML-DSA-87 (L5)	(8,7)	2	60	120	2^{19}	3.85

In general, the ILP problem is *NP-complete* [GJ79, vzGS78]. Lenstra [Jr.83] showed that this problem is polynomially solvable with a time complexity of $2^{O(n^3)}\cdot(m\cdot\log V)^{O(1)}$ when the number of variables n is a constant, where V represents the maximum absolute value of the coefficients in $\mathbf{A}$ and $\mathbf{b}$. More recently, Reis et al. improved this time complexity to $(\log n)^{O(n)}\cdot(m\cdot\log V)^{O(1)}$ in [RR23, Theorem 41].

3 Rejected challenge attack on Dilithium

As discussed in Section 2.2, the rejection sampling of Dilithium guarantees that those $\mathbf{z}$ that leak substantial information about the private key are rejected and not produced as output. In the context of side-channel attack, our objective is to recover the private key $\mathbf{s}_1$ using implementation leakages of the rejected $\mathbf{z}$ and the corresponding c.

As shown in Algorithm 4, $\mathbf{z}=\mathbf{y}+c\mathbf{s}_1$ and the coefficients of the mask polynomial vector $\mathbf{y}$ are uniformly distributed in the interval $[-\gamma_1+1,\gamma_1]$, the challenge polynomial c has precisely τ nonzero coefficients (-1 or 1), the secret key polynomial vector has a small norm such that $\|\mathbf{s}_1\|_\infty\le\eta$ with each coefficient drawn from a discrete form of the Gaussian distribution $\mathcal{N}(0,\frac{\eta\cdot(\eta+1)}{3})$. Moreover, we recall a basic property of polynomial multiplication modulo X^n+1. Denoted by c_i the $(i+1)$-th row of the coefficient matrix as Equation (1), which is given by $\mathsf{rot}(c,i)=c\cdot X^i \mod X^n+1$. Then, denoted by $\mathbf{x}_i$ the $(i+1)$-th coefficient of the product $c\mathbf{s}_1$, which is given by the inner product $\mathbf{x}_i=\langle c_i,\mathbf{s}_1\rangle$.

$$\begin{bmatrix}\mathbf{x}_0\\ \mathbf{x}_1\\ \mathbf{x}_2\\ \vdots\\ \mathbf{x}_{n-2}\\ \mathbf{x}_{n-1}\end{bmatrix}\stackrel{\text{def}}{=}c\mathbf{s}_1=\begin{bmatrix}c_0 & -c_{n-1} & -c_{n-2} & \cdots & -c_2 & -c_1\\ c_1 & c_0 & -c_{n-1} & \cdots & -c_3 & -c_2\\ c_2 & c_1 & c_0 & \cdots & -c_4 & -c_3\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ c_{n-2} & c_{n-3} & c_{n-4} & \cdots & c_0 & -c_{n-1}\\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots & c_1 & c_0\end{bmatrix}\begin{bmatrix}\mathbf{s}_{1_0}\\ \mathbf{s}_{1_1}\\ \mathbf{s}_{1_2}\\ \vdots\\ \mathbf{s}_{1_{n-2}}\\ \mathbf{s}_{1_{n-1}}\end{bmatrix}\tag{1}$$

By the Central Limit Theorem, the distribution of each $\mathbf{x}_i$ approximates $\mathcal{N}(0,\frac{\tau\cdot\eta\cdot(\eta+1)}{3})$, taking into account that c has exactly τ non-zero coefficients. This equation is straightforward to solve to recover $\mathbf{s}_1$ with known $c\mathbf{s}_1$ and its corresponding challenge c. However, as mentioned above, known SCA countermeasures already suggest protecting secret-involved calculations such as $c\mathbf{s}_1$ and the mask polynomial vector $\mathbf{y}$. We address the problem of recovering $\mathbf{s}_1$ using rejection information alone (without leakage about multiplication $c\mathbf{s}_1$ or mask polynomial vector $\mathbf{y}$).

The probability of $\mathbf{z}$ been rejected is $\Pr\left[\|\mathbf{z}\|_\infty\ge\gamma_1-\beta\right]=1-(1-\frac{\beta}{\gamma_1-0.5})^{\ell n}\approx 1-e^{-\beta\ell n/\gamma_1}$ according to the Dilithium documentation [DKL+18]. Given a rejected $\mathbf{z}$, at least one of its coefficients $\mathbf{z}_i$ will be in the interval $[-\gamma_1+1-\beta,-\gamma_1+\beta]\cup[\gamma_1-\beta,\gamma_1+\beta]$, i.e., $4\beta+1$ possible integral values $\mathbf{z}_i$. These $4\beta+1$ possible $\mathbf{z}_i$ values can confine the corresponding $\mathbf{y}_i$ to a much smaller interval than the original $[-\gamma_1+1,\gamma_1]$ and thus further limit the range of $\mathbf{x}_i$. For instance,

- if $\mathbf{z}_i=-\gamma_1$, then $-\gamma_1+1\le\mathbf{y}_i\le-\gamma_1+\beta$ and $-\beta\le\mathbf{x}_i\le-1$;
- if $\mathbf{z}_i=-\gamma_1-1$, then $-\gamma_1+1\le\mathbf{y}_i\le-\gamma_1-1+\beta$ and $-\beta\le\mathbf{x}_i\le-2$;

- if $\mathbf{z}_i = \gamma_1 - 1$, then $\gamma_1 - 1 - \beta \leq \mathbf{y}_i \leq \gamma_1$ and $-1 \leq \mathbf{x}_i \leq \beta$;
- if $\mathbf{z}_i = \gamma_1$, then $\gamma_1 - \beta \leq \mathbf{y}_i \leq \gamma_1$ and $0 \leq \mathbf{x}_i \leq \beta$;
- if $\mathbf{z}_i = -\gamma_1 + 1$, then $-\gamma_1 + 1 \leq \mathbf{y}_i \leq -\gamma_1 - 1 + \beta$ and $-\beta \leq \mathbf{x}_i \leq 0$;
- if $\mathbf{z}_i = \gamma_1 - 2$, then $\gamma_1 - 1 - \beta \leq \mathbf{y}_i \leq \gamma_1$ and $-2 \leq \mathbf{x}_i \leq \beta$.

With the aforementioned upper/lower constraints of $\mathbf{x}_i$, we can construct an ILP instance of $\mathbf{x}_i$ with (ℓn) variables (the polynomial degree parameter n is 256 and the ℓ parameter of $\mathbf{s}_1$ set to 4/5/7 for security level 2/3/5 in ML-DSA/Dilithium respectively). By solving this ILP problem, we can recover $\mathbf{s}_1$ given linearly many pairs of rejected ($\mathbf{z}_i$, c).

Our main result of the rejected challenge attack on $\mathbf{s}_1$ is as follows:

Definition 1 (Rejected coefficient of a rejected challenge)**.** For a rejected challenge c, its rejected coefficient is a coefficient in c's corresponding $\mathbf{z}$ such that $|\mathbf{z}_i| \geq \gamma_1 - \beta$. Moreover, i is referred to as the position of the rejected coefficient.

Theorem 1 (ILP-based Dilithium key recovery)**.** *For the Dilithium signing (Algorithm 3), given $m > \ell n$ rejected challenges c and the corresponding values and positions of their rejected coefficients, an adversary can recover $\mathbf{s}_1$ with probability of at least $1 - \ell n \cdot e^{-\frac{\tau m}{\ell n}}$ and a time complexity of $(\log \ell n)^{O(\ell n)} \cdot (m \cdot \log \beta)^{O(1)}$.*

Proof. Each rejected coefficient $\mathbf{z}_i$ will assume one of $4\beta+1$ values in the set defined as $S_z \stackrel{\text{def}}{=} \{-\gamma_1+1-\beta, -\gamma_1+2-\beta, \ldots, -\gamma_1-1+\beta, -\gamma_1+\beta\} \cup \{\gamma_1-\beta, \gamma_1-\beta+1, \ldots, \gamma_1-1+\beta, \gamma_1+\beta\}$. For each rejected coefficient $\mathbf{z}_i$, one can construct two inequalities about the unknown $\mathbf{s}_1$. For example, if the rejected $\mathbf{z}_i = -\gamma_1$, we get two inequalities $-\beta \leq \langle c_i, \mathbf{s}_1 \rangle \leq -1$ according to 3, which can be further written as:

$$\begin{bmatrix} -\mathsf{rot}(c,i) \\ \mathsf{rot}(c,i) \end{bmatrix} \mathbf{s}_1 \leq \begin{bmatrix} \beta \\ -1 \end{bmatrix}$$

Taking into account the m rejected coefficients, we can construct an ILP instance $\mathbf{C}\mathbf{s}_1 \leq \mathbf{b}$ for a $2m \times \ell n$ matrix $\mathbf{C}$ and a $2m \times 1$ vector $\mathbf{b}$, where the number of variables in $\mathbf{s}_1$ is ℓn, and the maximum absolute value of the coefficients in $\mathbf{C}$ and $\mathbf{b}$ is β. According to [RR23, Theorem 41], one can solve this ILP problem in time $(\log \ell n)^{O(\ell n)} \cdot (m \cdot \log \beta)^{O(1)}$ to recover the secret key $\mathbf{s}_1$.

For each rejected $\mathbf{z}_i$, only τ non-zero coefficients in the corresponding c are multiplied with the corresponding $\mathbf{s}_1$ as shown in Equation (1). Thus, each rejected $\mathbf{z}_i$ can only add constraints to the constructed ILP instance about τ coefficients of $\mathbf{s}_1$. To fully recover $\mathbf{s}_1$, all the ℓn coefficients of $\mathbf{s}_1$ must be covered by the rejected coefficients $\mathbf{z}_i$. We denote $\mathrm{E}_j (j \in [1, \ell n])$ as the event that those m rejected coefficients $\mathbf{z}_i$ have not bounded the j-th coefficient of $\mathbf{s}_1$. The probability that those m rejected coefficients $\mathbf{z}_i$ have not bounded at least one coefficient of $\mathbf{s}_1$ is $\Pr\left[\cup_{j=1}^{\ell n} \mathrm{E}_j\right] \leq \sum_{j=1}^{\ell n} \Pr\left[\mathrm{E}_j\right] = \ell n \cdot (1-p)^m \approx \ell n \cdot e^{-pm}$, in which, $p = \frac{\tau}{\ell n}$ corresponds to the probability of any coefficient of $\mathbf{s}_1$ bounded by any rejected $\mathbf{z}_i$. Putting all together, the probability that all coefficients of $\mathbf{s}_1$ bounded by that m rejected $\mathbf{z}_i$ is $1 - \Pr\left[\cup_{j=1}^{\ell n} \mathrm{E}_j\right] \geq 1 - \ell n \cdot e^{-\frac{\tau m}{\ell n}}$. This completes the proof. □

In practice, the coefficient $\mathbf{z}_i$ is typically a 32-bit signed integer using two's complement representation. It is very difficult to recover each and every $4\beta + 1$ values in S_z with 100% SR. Targeting HW values of the rejected coefficients $\mathbf{z}_i$ can significantly reduce the number of classes to identify. There are four particular values $-\gamma_1$, $-\gamma_1 - 1$, $\gamma_1 - 1$, and γ_1 in S_z that correspond to unique HW values (detailed in Appendix A), which are easier to extract via SCA leakages. We thus define the following rejected challenge that we will exploit towards a more useful and practical attack.

Definition 2 (Rejected challenge of interest - RCoI)**.** For the Dilithium signing (Algorithm 3), a challenge c is a rejected challenge of interest, if c is a rejected challenge and c's rejected coefficient equals $-\gamma_1$, $-\gamma_1 - 1$, $\gamma_1 - 1$ or γ_1.

Lemma 1. *For each generation of the challenge c in Dilithium signing (Algorithm 3), the probability that c is a rejected challenge of interest is* $\dfrac{1-(1-\frac{\beta}{\gamma_1-0.5})^{\ell n}}{\beta+0.25}$.

Proof. Conditioned on the event $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$, which occurs with probability

$$\Pr\left[\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta\right] = 1 - (1 - \frac{\beta}{\gamma_1 - 0.5})^{\ell n} \ ,$$

at least one coefficient of $\mathbf{z}$ is uniform over S_z. It thus hits the four values (as in Definition 2) with probability $\frac{4}{4\beta+1} = \frac{1}{\beta+0.25}$, which completes the proof. □

Taking into account Lemma 1, we can deduce a special case of Theorem 1 as below based on rejected challenge of interest,

Theorem 2 (Key recovery with RCoI)**.** *For M executions of the Dilithium signing (Algorithm 3) with $M > \ell nu$ and $u = \dfrac{2\beta+0.5}{1-(1-\frac{\beta}{\gamma_1-0.5})^{\ell n}}$, an adversary who obtains the rejected challenges of interest and the corresponding values and positions of their rejected coefficients can recover $\mathbf{s}_1$ with a probability of at least $(1 - \ell n e^{-\frac{\tau M}{\ell n u}} - e^{-\frac{M}{4u}})$ and a time complexity of $(\log \ell n)^{O(\ell n)} \cdot (\frac{M \log \beta}{u})^{O(1)}$.*

Proof. According to Lemma 1, the expected number of rejected challenges of interest among M signing executions is $M \cdot \dfrac{1-(1-\frac{\beta}{\gamma_1-0.5})^{\ell n}}{\beta+0.25} = \frac{2M}{u}$. We have by a Chernoff bound that the number of rejected challenges of interest is at least, $\frac{M}{u}$ except with a probability of $e^{-\frac{M}{4u}}$. Then, it is natural to replace $m = \frac{M}{u}$ in Theorem 1, which leads to recovering $\mathbf{s}_1$ with probability of at least $(1 - \ell n e^{-\frac{\tau M}{\ell n u}} - e^{-\frac{M}{4u}})$ and a time complexity of $(\log \ell n)^{O(\ell n)} \cdot (\frac{M \log \beta}{u})^{O(1)}$. □

4 Experimental results

The time complexity given in Theorem 2 is not of much practical significance because it corresponds to the worst-case scenario and contains a huge exponential factor. To assess the effectiveness of the proposed rejected challenge attack, we first evaluate it to recover the entire private key $\mathbf{s}_1$ considering Dilithium-2/3/5 parameter sets. In our experiments, instead of implementing the algorithm proposed in [RR23], we choose the built-in `branch and bound` [LMSK63] exact algorithm from `GLPK` MILP solver of SAGE to solve the above ILP. This exact algorithm turns out to be very efficient in practice, especially with effective bounding techniques one can prune away large sections of the tree, dramatically reducing the actual computation time. Our experimental results are in line with this intuition, i.e., the constructed ILP instances are solved in seconds or minutes.

Since the coefficients of $\mathbf{s}_1$ follow the discreet Gaussian distribution with a mean of 0, we set the objective function of the constructed ILP instance to minimize the sum of all $\mathbf{s}_1$ coefficients (i.e., the variables of the constructed ILP instance). This is equivalent to minimizing the mean value of all coefficients of $\mathbf{s}_1$, which is expected to be close to 0. We emulate the rejected challenge attack by sampling random pairs of Dilithium keys, generating corresponding signatures, and recording only those rejected

$\mathbf{z}$ with $\mathbf{z}_i \in \{-\gamma_1, -\gamma_1 - 1, \gamma_1 - 1, \gamma_1\}$ and the corresponding c, where '$\mathbf{i}$' is the index of the coefficient of $\mathbf{z}$ that fails the bound check. Our experiments consider the bound check of $\mathbf{z}$ from the Early-Abort implementation, which is regarded as the worst-case scenario from the attacker perspective. This is because, in the Early-Abort implementation, only one $\mathbf{z}_i$ can be exploited for each rejected $\mathbf{z}$ (before reaching an early abort). In contrast, in other (for example, constant-timing) implementations, multiple interesting $\mathbf{z}_i$ values may be found in a single rejected $\mathbf{z}$. We summarize the experimental rejected challenge attack results in Table 3. The experiments are repeated 100 times for each setting on randomly generated key pairs, so the success rate "SR" column of full key recovery of $\mathbf{s}_1$ corresponds to the average of 100 experiments. We implemented the experiments in SAGE 10.3 (Ubuntu 22.04.4) with an Intel® Core™ i5-7500 CPU 3.40GHz. All experiments used the default `GLPK` MILP solver of SAGE with a timeout of 1 hour. Each successful full key recovery of $\mathbf{s}_1$ takes a matter of seconds or minutes.

The second column of Table 3 contains two numbers. The first number indicates the number of signatures generated for each random key pair. The second number corresponds to the expected number of rejected signatures. Interestingly, the results suggest that the complexity of this attack does not directly depend on the security levels. On the one hand, higher security levels require more signatures. The SR reaches 100% for Dilithium-2/3/5. On the other hand, Dilithium-3 requires 10^7 more signatures than Dilithium-5 (with 7×10^6 more signatures than for Dilithium-5, 17 out of 100 experiments failed to recover the full $\mathbf{s}_1$ with timeout error). The fact that the expected number of rejected four particular $\mathbf{z}_i$ values depends on the parameters ℓ and β explains this. That is, the probability of at least one (considering the Early-Abort setting in the bound check of the reference implementation) $\mathbf{z}_i$ coefficient rejected is $\Pr\left[\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta\right] = 1 - (1 - \frac{\beta}{\gamma_1 - 0.5})^{\ell n}$, approximately, $45.64\%, 38.04\%, 33.65\%$ for Dilithium-2/3/5, respectively. With higher security levels, the expected number of rejected $\mathbf{z}$ is getting smaller, given the same amount of signature generation trials. However, the probability of the rejected $\mathbf{z}_i$ coefficient being one of those four values is about $\frac{1}{4\beta+1}$, i.e., $\frac{1}{313}$, $\frac{1}{785}$, $\frac{1}{481}$ for Dilithium-2/3/5, respectively. This explains the fact that Dilithium-3 needs more signature generations to get sufficient rejected $\mathbf{z}_i$ compared to Dilithium-5.

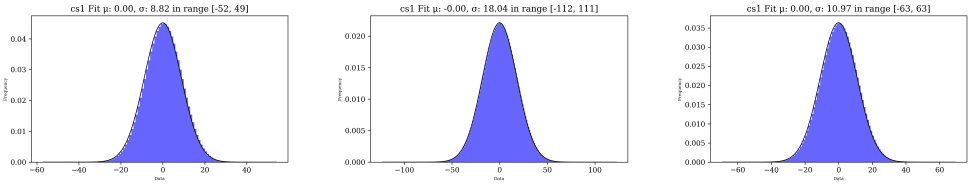
Table 3: Rejected challenge attack results using 4 rejected values of $\mathbf{z}$'s coefficients

	#Sign (#Rejected z)	SR	ILP constraint bounds	Running time (s)
ML-DSA-44 (L2)	3×10^6 (1.37×10^6)	100%	[−52,−1]; [−52,−2]; [−1,49]; [0,49]	104
ML-DSA-65 (L3)	2.3×10^7 (8.75×10^6)	100%	[−112,−1]; [−112,−2]; [−1,111]; [0,111]	201
ML-DSA-87 (L5)	1.3×10^7 (4.38×10^6)	100%	[−63,−1]; [−63,−2]; [−1,63]; [0,63]	75

The fourth column of Table 3 consists of four pairs of constraints, represented by interval [lower-bound,upper-bound], which correspond to the four respective rejected $\mathbf{z}_i$ values, namely $\mathbf{z}_i = -\gamma_1, \mathbf{z}_i = -\gamma_1 - 1, \mathbf{z}_i = \gamma_1 - 1$, or $\mathbf{z}_i = \gamma_1$. As discussed, we could have just put the standard constraints [$-\beta$,−1], [$-\beta$,−2], [−1,β] and [0,β] to represent the respective inequalities for the four cases $-\beta \leq \mathbf{x}_i \leq -1$, $-\beta \leq \mathbf{x}_i \leq -2$, $-1 \leq \mathbf{x}_i \leq \beta$, and $0 \leq \mathbf{x}_i \leq \beta$, where β is 78, 196 or 120 for Dilithium-2/3/5 respectively. For further optimization, we observe that smaller absolute values can be used in place of above upper/lower bound $\pm\beta$ without affecting the accuracy. We statistically analyze the distribution of $\mathbf{x}_i$ of a rejected $\mathbf{z}_i$. As depicted in the histogram of Figure 1a (resp., Figure 1b and Figure 1c), we execute 10^9 (resp., 2.5×10^9 and 2×10^9) times of Dilithium-2 (resp., Dilithium-3 and Dilithium-5) signing process, and record 458508787 (resp., 952794746 and 675233816) rejected $\mathbf{z}_i$ and the corresponding values of $\mathbf{x}_i$. As can be observed from the experimental results, the lower bound of $-\beta = -78$ (resp., $-\beta = -196$ and $-\beta = -120$) can be reduced to -52 (resp., -112 and -63) for Dilithium-2 (resp., Dilithium-3 and Dilithium-5), which already bounds all values in our experiments. The case for upper bound β is likewise.

We mention that the above empirical optimization has theoretical support. We can

observe that $\mathbf{x}_i$ approximates the $\mathcal{N}(0, \sigma^2 = \frac{\tau \cdot \eta \cdot (\eta+1)}{3})$ normal distribution as outlined in black in Figures 1a, 1b and 1c. The probability that a value drawn from this distribution falls in the interval $[-5.3\sigma, 5.3\sigma]$, is at least 99.99999% (see e.g., [FSK03]) by substituting concrete values into the parameters. The last column of Table 3 gives the average running time of full key recovery of $\mathbf{s}_1$, which is less than 4 minutes for all three security levels.



(a) Dilithium-2. (b) Dilithium-3. (c) Dilithium-5.

Figure 1: Distributions of $\mathbf{x}_i$ for Dilithium at different security levels.

In all Dilithium-2/3/5 cases, we verified the effectiveness of the proposed rejected challenge attack using those four particular values of $\mathbf{z}$'s rejected coefficient with the corresponding challenge c to disclose the entire private key $\mathbf{s}_1$. Next, we will demonstrate utilizing this rejected challenge attack using the obtained side-channel leakages of $\mathbf{z}$'s coefficients and its corresponding c via profiled attacks in a realistic context.

5 A use case of proposed attack from real SCA leakage

In this section, we validate the practicality and effectiveness of our attack. Specifically, we propose a deep learning-based method to recover the rejected coefficient of $\mathbf{z}$ and the corresponding challenge c from side-channel leakage. By leveraging the proposed rejected challenge attack, we achieve a successful side-channel key-recovery attack on real implementations.

5.1 Deep learning profiled attack

Since the introduction of template attacks [CRR02], the SCA community has regarded profiled attacks as the most powerful type of side-channel attacks. Profiled attacks consist of two phases, i.e., profiling and attacking phases. In the profiling phase, an attacker uses a profiling device, where he controls or at least knows the key, to model the leakage characteristics of key-dependent sensitive data (e.g., the coefficients of the rejected challenge c in this work). In the attacking phase, the attacker recovers sensitive information from the target device using side-channel leakage and the model built during the profiling phase.

We employ Deep Learning (DL) profiled attacks to experimentally evaluate the effectiveness of our proposed rejected challenge attack. Specifically, we use a Multi-Layer Perceptron (MLP) neural network model for these DL profiled attacks. Despite its simple and shallow structure, the MLP model demonstrated strong performance. Unlike the classical profiled attacks (e.g., template attacks), DL profiled attacks do not assume any specific leakage characteristics. They exploit features (sample points from side-channel traces) via neural networks to classify labels (sensitive data in the SCA context). The training process of neural networks in the profiling phase aims to construct a classifier function $F(.) : \mathbb{R}^d \rightarrow \mathbb{R}^{|S|}$, mapping the input trace $\mathbf{l} \in R^d$ to an output vector $\mathbf{p} \in R^{|S|}$, representing classification scores. During training, the backpropagation method [Kel60, GBC16] is applied to each training batch, updating the trainable parameters of the neural network model to minimize the loss function, which quantifies classification errors. In the attacking phase, the trained model (i.e., $F(.)$ with all the final updated parameters) is used to classify

each attack trace and produce a probability vector $\mathbf{p}[s_g]$. We detail the parameter settings for our neural networks in Section 5.2.

5.2 Attack setup

Our target is the reference Dilithium implementation [1] on a 32-bit STM32F microcontroller running at 8 MHz. During the execution of the signing process in Dilithium (before obtaining the signature σ), the values of c and $\mathbf{z}$, whether $\mathbf{z}$ is rejected in the while loop, are unknown to the attacker. We need to recover the corresponding intermediate values through SCA. The following are the leakage points for the corresponding intermediate values.

To implement the attack, we acquired power consumption traces using a Picoscope 5244D oscilloscope and set the sampling rate to 62.5 MHz. We need to distinguish between the various types of leakage discussed above, concretely differentiating the values of the coefficients of c, distinguishing some specific values of the coefficients of $\mathbf{z}$, and determining whether the coefficients of $\mathbf{z}$ will be rejected. This task involves a one-dimensional data classification problem. Given that each type of leakage corresponds to a limited number of categories, we found that the representational capacity of an MLP is sufficient. Consequently, we designed an MLP model to perform this classification. Table 4 summarizes the details of the used MLP model and the corresponding hyper-parameters.

Table 4: MLP Model Architecture

Layer Type	**Output Size**	**Details**
Input Layer	(input size,)	-
Dense Layer	128	Activation: ReLU, Regularization: L2 (0.01)
Dropout	128	Dropout Rate: 0.2
Output Layer	classes num	Activation: Softmax, Optimizer: Adam

It is a simple MLP structure, where the Dense layer (fully connected layer) is the core component. Each neuron connects to all neurons in the previous layer. We use the ReLU activation function to introduce non-linearity and accelerate the training process. We add L2 regularization to constrain the weight values to reduce the risk of overfitting, along with Dropout, which randomly "disables" a portion of the neurons during training to enhance the model's generalization ability.

In terms of data preprocessing, all profiling and attacking traces are normalized using the *StandardScalar* function from the Scikit-learn [PVG+11] library by removing the mean and scaling to unit variance. For each data point x, the standardized data is $x_{\text{scaled}} = \frac{x-\mu}{\sigma}$, where μ is the mean of the sample data, and σ is the standard deviation of the sample data.

There is an implicit assumption about the knowledge of the Points of Interest (POIs), i.e., the attackers/evaluators can determine the rough timing intervals of each coefficient processing of the rejected challenge c and the corresponding $\mathbf{z}_i$ in the side-channel traces. It is feasible for most of security products in a grey-box testing context shown in [KAA21, QLZ+23a, QLZ+24].

5.3 Recovering c

c is transformed into its NTT form by employing the Cooley-Tukey (CT) Algorithm. Figure 2 illustrates the NTT code provided by the Dilithium's reference implementation [2].

[1] The specific implementation is detailed in the Dilithium repository on GitHub: https://github.com/pq-crystals/dilithium

[2] The reference implementation is available at https://github.com/pq-crystals/dilithium.

The leakage generated by the butterfly operation, highlighted in red, can help us recover c. For the polynomial c with n coefficients $c_0, \ldots, c_{n-1}$, this process consists of $\log n$ layers, with $n/2$ CT Butterfly Units (BFUs) repeated for all coefficients in each layer.

```
1       void ntt(int32_t a[N]) {
2           unsigned int len, start, j, k=0; int32_t zeta, t;
3           for(len = 128; len > 0; len >>= 1) {
4               for(start = 0; start < N; start = j + len) {
5                   zeta = zetas[++k];
6                   for(j = start; j < start + len; ++j) {
7                       /*Attack Position*/
8                       t = montgomery_reduce((int64_t)zeta * a[j + len]);
9                       a[j + len] = a[j] - t;
10                      a[j] = a[j] + t;
11                  }
12              }
13          }
14      }
```

Figure 2: C Implementation of NTT.

Each CT BFU operates on two coefficients, referred to as *upper and lower coefficients*, and produces two outputs involving a twiddle factor. Specifically, the product between the upper coefficient and the twiddle factor is added to (or subtracted from) the lower coefficient to generate the results.

Figure 3 provides a brief summary of the NTT process with 16 coefficients. Our attack is to recover the entire challenge c by exploiting the leakage from NTT. In the rest of this subsection, we denote each coefficient in the input and output of the NTT as c_i and $\hat{c}_i$, respectively, with $i \in [0, n-1]$, and we call them as *input or output coefficients*. Additionally, we denote each coefficient in the j^{th} layer as $c_{i,j}$ with $i \in [0, n-1]$ and we call them as *inner coefficients, or, coefficients for the sake of brevity*. Moreover, for an upper coefficient $c_{i,j}$ in the j^{th} layer, we denote the corresponding product of $c_{i,j}$ and the associated twiddle factor as $c'_{i,j} = c_{i,j} \cdot \zeta$.

5.3.1 Attacking the first layer

First, we can recover the upper input coefficient of each BFU in the first layer with perfect accuracy. Because the possible values of the upper input coefficient, namely -1, 0, and 1, produce significantly distinct leakage patterns. These patterns are related to the leakages of the upper input coefficient, denoted as c_i and $c_i \cdot \zeta$, where $\frac{n}{2} \leq i \leq n-1$. We focus on the Hamming weight leakage and present the Hamming weights of c_i and $c_i \cdot \zeta$ below.

- When $c_i = -1$, the Hamming weights are 32 and 14, respectively.
- When $c_i = 0$, the Hamming weights are 0 and 0, respectively.
- When $c_i = 1$, the Hamming weights are 1 and 19, respectively.

Additionally, we can distinguish each lower input coefficient value of -1 from 0 and 1 in the first layer with perfect accuracy. Because the Hamming weight of -1 differs significantly from those of 0 and 1.

Figure 4 presents the attack results for both the upper and lower input coefficients in the first layer of the NTT. Figures Figure 4a and Figure 4c illustrate a relationship between the model's (epochs=4, batch size=512) prediction accuracy and the number of profiling traces for the upper and lower input coefficients, respectively. For the upper input coefficients, the accuracy is measured by distinguishing between -1, 0, and 1. At the same time, for the lower input coefficients, the task is to distinguish -1 from $\{0, 1\}$.

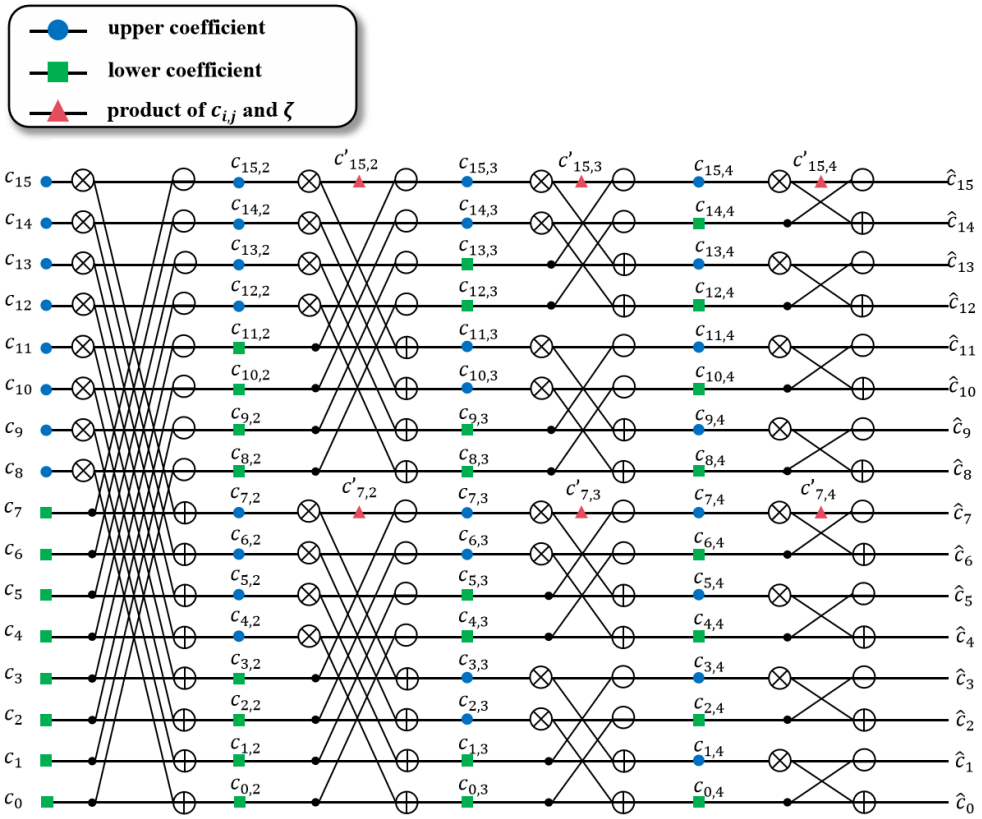


Figure 3: 16×16 BFU construction.

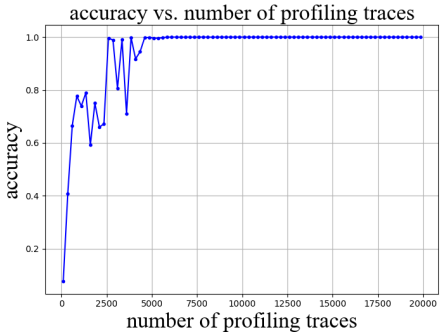
We also provide the confusion matrix to describe the accuracy. Each element in the confusion matrix represents the model's prediction and the actual class matching for a specific category. We can observe from Figures 4b and 4d that the prediction accuracy has reached 100%.

In the rest of this subsection, let B denote the input coefficients directly recovered by the first-layer leakage, and thus all the input upper input coefficients are contained in B, i.e., $\bigcup_{k=1}^{\frac{n}{2^j}} c_{n-k} \subseteq B$. However, since the Hamming weight leakage of 0 and 1 are very similar, it is challenging to distinguish the lower input coefficient value of 1 from 0. To address this, we need to consider further leakages from other layers.

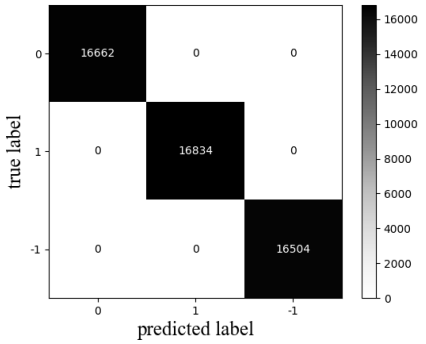
5.3.2 Attacking the other layers

Given all the lower input coefficients with values of -1 and the values of upper input coefficients in the first layer, we propose the method to distinguish the remaining lower input coefficients whose values are within $\{0, 1\}$. Through the propagation of BFU in NTT, the differences in Hamming weights will be amplified, especially before and after the multiplication. Our attack exploits the leakages of $c_{i,j}$ and $c'_{i,j}$.

Through side-channel leakage, we can recover approximately the Hamming weight of each lower coefficient and its product with the corresponding twiddle factor in the 2^{nd} to $(\log n)^{\text{th}}$ layers. We mean by 'approximately' that a tolerance ϵ is allowed between the actual Hamming weight and the predicted Hamming weight. In other words, through leakage, in 2^{nd} to $(\log n)^{\text{th}}$ layers, we can recover that the Hamming weight of a lower coefficient (say, $c_{i,j}$) lies between $\text{HW}(c_{i,j})-\epsilon$ to $\text{HW}(c_{i,j})+\epsilon$, as well as the Hamming weight

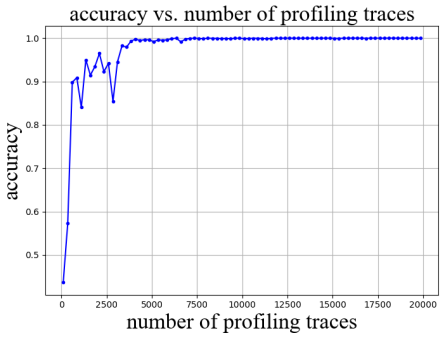


(a) Profiling set size vs. attack accuracy of an upper input coefficient.

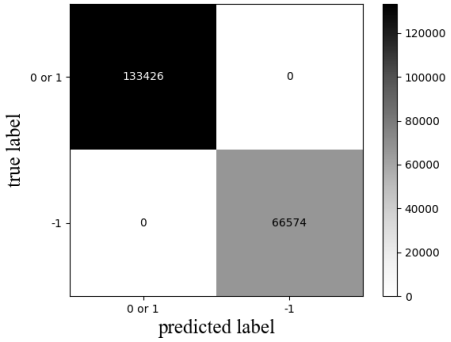


(b) The confusion matrix of the upper input coefficient for distinguishing among the classes $\{0, 1, -1\}$

.



(c) Profiling set size vs. attack accuracy of a lower input coefficient.



(d) The confusion matrix of the lower input coefficient for distinguishing between the class $\{-1\}$ and $\{0, 1\}$.

Figure 4: Attack results of the first-layer NTT.

of $c'_{i,j}$ lies between $\mathrm{HW}(c'_{i,j})-\epsilon$ to $\mathrm{HW}(c'_{i,j})+\epsilon$. Specifically, we denote the approximate Hamming weight of a coefficient $c_{i,j}$ (resp., $c'_{i,j}$) with tolerance ϵ as $\mathrm{HW}_\epsilon(c_{i,j})$ (resp., $\mathrm{HW}_\epsilon(c'_{i,j})$).

The side-channel prediction accuracy of this approximation depends on the value of ϵ. Figure 5 illustrates the prediction accuracy relative to ϵ. We observe that when $\epsilon \geq 3$, the prediction accuracy reaches 100% (The model has epochs set to 20 and a batch size of 512). In the rest of this subsection, we present the recovery of lower input coefficients $c_0, \ldots, c_{\frac{n}{2}-1}$ using the approximate Hamming weights.

Upper coefficients $C_{i,j} \stackrel{\text{def}}{=} \left\{c_{f(i,j,k),j}, c'_{f(i,j,k),j} \mid k = 0, 1, 2, \ldots, 2^{j-1}-1,\ n-\frac{n}{2^j} \leq i < n\right\}$, where $f(i,j,k) \stackrel{\text{def}}{=} i-\frac{k\cdot n}{2^{j-1}}$. These coefficients in the j^{th} layer are determined by the input coefficients $A_{i,j} \stackrel{\text{def}}{=} \left\{c_{i-\frac{k\cdot n}{2^{j-1}}} \mid k = 0, 1, \ldots, 2^{j-1}-1\right\}$. Let $A_j \stackrel{\text{def}}{=} A_{j-1} \cup \bigcup_{k=1}^{\frac{n}{2^j}} A_{n-k,j}$, so we have $A_{\log n} = \{c_1, c_2, \ldots, c_{n-1}\}$, and $\{c_{i-\frac{(2^j-1)\cdot n}{2^j}}\} = A_{i,j}/A_{j-1}$.

Assuming A_{j-1} is known, it is possible to resolve the input coefficients $c_{g(i,j)} = A_{i,j}/A_{j-1}$, with $g(i,j) \stackrel{\text{def}}{=} i-\frac{(2^{j-1}-1)\cdot n}{2^{j-1}}$ by a probability. It can be achieved by exhaustively guessing the value of $c_{g(i,j)}$, calculating the corresponding Hamming weights of the coefficients in $C_{i,j}$, and verifying whether the computed Hamming weights fall within the expected

Algorithm 5 Attack with the approximate Hamming weights

Input:
  Side-channel leakage of NTT, MLP model, tolerance ϵ.
Output:
  All input coefficients $\{c_0, \ldots, c_{255}\}$ or Unsuccess.
$B \overset{MLP}{\longleftarrow}$ **Side-channel leakage**
  ▷ B represents the input coefficients recovered by leakage of the first layer.
for $j \leftarrow 2$ **to** $\log n$ **do** ▷ j represents the j^{th} layer in the NTT.
  for $i \leftarrow n - \frac{n}{2^j}$ **to** $n-1$ **do**
    $f(i,j,k) \overset{\text{def}}{=} i - \frac{k \cdot n}{2^{j-1}}$, $g(i,j) \overset{\text{def}}{=} i - \frac{(2^{j-1}-1)\cdot n}{2^{j-1}}$
    $C_{i,j} \overset{\text{def}}{=} \left\{c_{f(i,j,k),j}, c'_{f(i,j,k),j} \mid k = 0, 1, 2, \ldots, 2^{j-1}-1\right\}$
    $A_{i,j} \overset{\text{def}}{=} \{c_{f(i,j,k)} \mid k = 0, 1, 2, \ldots, 2^{j-1}-1\}$ ▷ Related input coefficients.
    $c_{g(i,j)} = A_{i,j} / A_{j-1}$ ▷ $c_{g(i,j)}$ is the input coefficient to be recovered.
    if $c_{g(i,j)} \in B$ **then** ▷ Check if $c_{g(i,j)}$ is recovered in the first layer.
      $c_{g(i,j)} \leftarrow -1$ **and** break
    $mark_1 \leftarrow \{False\}^2$ ▷ Indicate whether $\tilde{c}_{g(i,j)}$ can be recovered.
    $mark_2 \leftarrow \{False\}^{2^{j-1}}$ ▷ Indicate whether $c_{f(i,j,k),j}$ can be recovered.
    for $\tilde{c}_{g(i,j)} \leftarrow 0$ **to** 1 **do**
      $\tilde{C}_{i,j} \overset{\text{def}}{=} \left\{\tilde{c}_{f(i,j,k),j}, \tilde{c}'_{f(i,j,k),j} \mid k = 0, 1, 2, \ldots, 2^{j-1}-1\right\}$
        ▷ Coefficients derived from $\tilde{c}_{g(i,j)}$ and A_{j-1} in j^{th} layer.
      for $k \leftarrow 0$ **to** $2^{j-1}-1$ **do**
        $\text{HW}_\epsilon(c_{f(i,j,k),j})$, $\text{HW}_\epsilon(c'_{f(i,j,k),j}) \overset{MLP}{\longleftarrow}$ **Side-channel leakage**
        $\text{HW}(\tilde{c}_{f(i,j,k),j})$, $\text{HW}(\tilde{c}'_{f(i,j,k),j}) \overset{\tilde{c}_{g(i,j)}, A_{j-1}}{\longleftarrow}$ **Deduction**
          ▷ Derive HW from guessed $\tilde{c}_{g(i,j)}$ and previously recovered A_{j-1}.
        if $|\text{HW}(\tilde{c}_{f(i,j,k),j}) - \text{HW}_\epsilon(c_{f(i,j,k),j})| \leq \epsilon$
          and $|\text{HW}(\tilde{c'}_{f(i,j,k),j}) - \text{HW}_\epsilon(c'_{f(i,j,k),j})| \leq \epsilon$ **then**
            ▷ Check if HW is in the range of HW_ϵ
          $mark_2[k] \leftarrow True$
      if $False \in mark_2$ **then**
        ▷ If at least one HW is not in the range of HW_ϵ.
        return Unsuccess
      else
        $mark_1[\tilde{c}_{g(i,j)}] \leftarrow True$
    if $mark_1[0] == mark_1[1]$ **then** ▷ There exist conflicting results.
      return Unsuccess
    else
      $c_{g(i,j)} \leftarrow (mark_1[0] == True)\,?\,0 : 1$
  $A_j \overset{\text{def}}{=} A_{j-1} \cup \bigcup_{k=1}^{\frac{n}{2^j}} A_{n-k,j}$ ▷ The set of recovered coefficients.
    ▷ $A_{\log n} = \{c_1, c_2, \ldots, c_{n-1}\}$
if $c_0 \in B$ **then** ▷ Check if c_0 is recovered in the first layer.
  $c_0 \leftarrow -1$
else
  $c_0 \leftarrow (A_{\log n}$ contains τ nonzero input coefficients$)\,?\,0 : 1$
return $A_{\log n} \cup \{c_0\}$

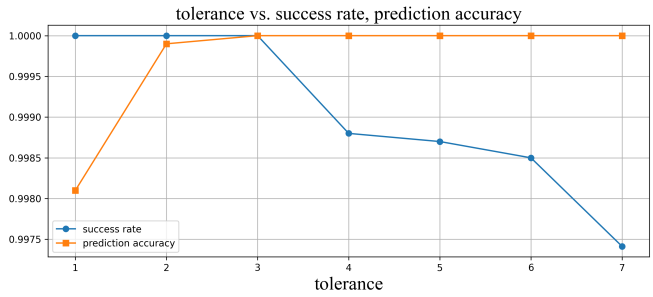


Figure 5: The relation between the calculated SR, prediction accuracy, and tolerance ϵ.

range of their approximated Hamming weights. Therefore, we can execute the attack by leveraging the leakage from each successive layer. Algorithm Algorithm 5 describes the process of this attack. In addition, we give an example of the attack in Section 5.3.3 for better understanding.

In the j^{th} iteration in the algorithm (lines 10 to 30), A_{j-1} is already recovered by previous layers. Then, for the lower input coefficient $c_{g(i,j)}$, if $c_{g(i,j)} = -1$, then $c_{g(i,j)} \in B$ is recovered in the first layer. Otherwise, we take a guess value $\tilde{c}_{g(i,j)} \in \{0, 1\}$, and calculate the coefficients in $C_{i,j}$ through the NTT route, denoted as $\tilde{C}_{i,j} \stackrel{\text{def}}{=} \{\tilde{c}_{f(i,j,k),j}, \tilde{c}'_{f(i,j,k),j} \mid k = 0, 1, 2, \ldots, 2^{j-1}-1\}$. If all the Hamming weights of the coefficients in $\tilde{C}_{i,j}$ fall within the range of approximated Hamming weights of the coefficients in $C_{i,j}$, the guessed value $\tilde{c}_{g(i,j)}$ may be correct. However, note that the two guess values 0 and 1 may both exhibit to be correct ones because of the too-large tolerate ϵ of the approximated Hamming weights. We regard this case as an unsuccessful attack.

At last, the only undetermined input coefficient is c_0, which can be obtained based on the value of τ. Concretely, since the number τ of nonzero input coefficients in $c_1, \ldots, c_{n-1}$ is known, we can directly ascertain whether or not c_0 is zero. Additionally, as we already know, if c_0 is -1 or not, its exact value can be determined.

The success rate of recovering all input coefficients using the above attack method depends on the value of the tolerance ϵ. To evaluate this, we perform the attack with different ϵ values and calculate the success rate by iterating the attack 40 million times for each ϵ value. Figure 5 shows the relationship between the success rate, the prediction accuracy of the SCA model, and the tolerance ϵ. As observed, both the attack success rate (of Algorithm 5) and the prediction accuracy of approximate Hamming weight reach 100% when $\epsilon = 3$.

5.3.3 Attack example

To better illustrate this algorithm, we provide an example based on NTT with 16 input coefficients, shown in Figure 3. Our example traces the process from the 1$^{\text{st}}$ to the final layer of the NTT.

- 1$^{\text{st}}$ **layer**: In conjunction with the description in Section 5.3.1, we can leverage the side-channel leakage from the first layer to recover the upper input coefficients of the first layer BFU (i.e., $\{c_8, c_9, \ldots, c_{15}\}$), as well as the lower input coefficients whose values are -1. It recovers input coefficients in the set B. The subsequent attacks are conducted based on the method described in Section 5.3.2.

- 2$^{\text{nd}}$ **layer**: Taking $i = 15$ as a case example, set $C_{15,2} = \{c_{15,2}, c'_{15,2}, c_{7,2}, c'_{7,2}\}$ is a function of input coefficients $A_{15,2} = \{c_{15}, c_7\}$, where $c_{15} \in A_1$ has already been determined during the attack process in the previous (i.e., the first) layer. If $c_7 = -1$, then $c_7 \in B$ is also known from the previous layer. If $c_7 \neq -1$, then we have

$A_{15,2}/A_1 = c_7$ and attempt to recover this input coefficient through the side-channel leakage of the second layer. For a hypothesis $\tilde{c}_7 \in \{0,1\}$ of c_7, we can compute the set $\tilde{C}_{15,2} = \{\tilde{c}_{15,2}, \tilde{c}'_{15,2}, \tilde{c}_{7,2}, \tilde{c}'_{7,2}\}$ by $\tilde{c}_7$ and c_{15} ($c_{15} \in A_1$) along the NTT path. Thus, we can compute the predicted Hamming weights of $\tilde{c}_{15,2}$, $\tilde{c}'_{15,2}$, $\tilde{c}_{7,2}$, and $\tilde{c}'_{7,2}$. Additionally, we can obtain the approximate Hamming weights of those four coefficients by exploiting the side-channel leakage. If and only if all predicted guessed Hamming weights match the corresponding approximate Hamming weights, the guess $\tilde{c}_7$ is deemed valid. Furthermore, the attack is successful if only one guessed value of c_7 is valid. If both hypotheses $\tilde{c}_7 = 0$ and $\tilde{c}_7 = 1$ are valid, this situation is considered an attack failure. Through the above attack scheme for the second layer attack, we can recover c_7, c_6, c_5, and c_4, so we have $A_2 = A_1 \cup A_{15,2} \cup A_{14,2} \cup A_{13,2} \cup A_{12,2}$, or equivalently, i.e., $A_2 = A_1 \cup \{c_7, c_6, c_5, c_4\}$.

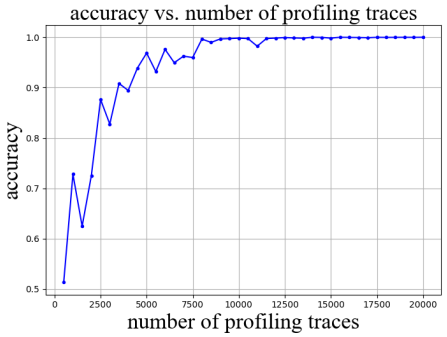
- 3^{rd} **layer**: Taking $i = 15$ as a case example, set $C_{15,3} = \left\{c_{f(15,k,3),3}, c'_{f(15,k,3),3} \mid k = 0,1,2,3\right\}$ is a function of input coefficients $A_{15,3} = \{c_{15}, c_{11}, c_7, c_3\}$, where $\{c_{15}, c_{11}, c_7\} \in A_2$ have been determined during the attack process in the second layer. If $c_3 = -1$, then $c_3 \in B$ is known in the first layer. If $c_3 \neq -1$, we have $A_{15,3}/A_2 = c_3$, and we attempt to recover this input coefficient through the side-channel leakage of the 3^{rd} layer. For a hypothesis $\tilde{c}_3 \in \{0,1\}$ of c_3, we can compute the set $\tilde{C}_{15,3} = \left\{\tilde{c}_{f(15,k,3),3}, \tilde{c}'_{f(15,k,3),3} \mid k = 0,1,2,3\right\}$ by $\tilde{c}_3$ and $\{c_{15}, c_{11}, c_7\} \in A_2$ along the NTT path. The subsequent attack process is similar to the description of attacking c_7 in the second layer. In the third layer attack, we can recover c_3 and c_2, so we have $A_3 = A_2 \cup \bigcup_{k=1}^{2} A_{n-k,3}$, or equivalently i.e. $A_2 \cup \{c_3, c_2\}$.

- 4^{th} **layer**: Taking $i = 15$ as a case example, set $C_{15,4} = \left\{c_{f(15,k,4),4}, c'_{f(15,k,4),4} \mid k = 0,1,2,\cdots,7\right\}$ is a function of input coefficients $A_{15,4} = \{c_{15}, c_{13}, c_{11}, c_9, c_7, c_5, c_3, c_1\}$, where $\{c_{15}, c_{13}, c_{11}, c_9, c_7, c_5, c_3\} \in A_3$ have been determined during the attack process in the third layer. If $c_1 = -1$, we know $c_3 \in B$ in the first layer. If $c_1 \neq -1$, we have $A_{15,4}/A_3 = c_1$ and attempt to recover this input coefficient through the side-channel leakage of the 4^{th} layer. For a hypothesis $\tilde{c}_1 \in \{0,1\}$ of c_1, we can compute the set $\tilde{C}_{15,4} = \left\{\tilde{c}_{f(15,k,4),4}, \tilde{c}'_{f(15,k,4),4} \mid k = 0,1,2,\cdots,7\right\}$ by $\tilde{c}_1$ and $\{c_{15}, c_{13}, c_{11}, c_9, c_7, c_5, c_3\} \in A_3$ along the NTT path. The subsequent attack process is similar to the description of attacking c_7 in the second layer. In the fourth layer attack, we can recover c_1, so we have $A_4 = A_3 \cup \bigcup_{k=1}^{1} A_{n-k,4}$, i.e. $A_3 \cup \{c_1\}$. At this point, we can recover all input coefficients involved in the multiplications within the NTT, with the only undetermined input coefficient being c_0. We can obtain its value through the τ value. If we have recovered the τ nonzero input coefficients in the polynomial c from $c_1, c_2, \ldots, c_{15}$, then $c_0 = 0$. Additionally, since we already know if c_0 is -1 or not, its exact value can be determined.

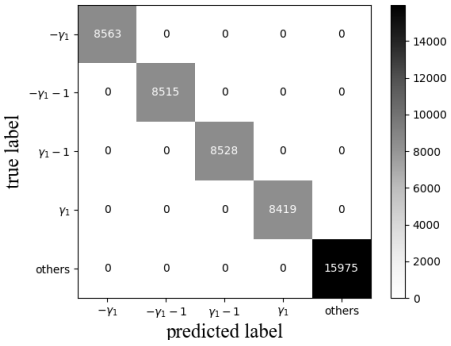
5.4 Recovering z

In Algorithm 3, line 11, $\mathbf{z}$ is calculated by $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$ in the polynomial ring modulo q. By exploiting the leakage of the $\mathbf{z}$ using DL profiled attacks, we can directly determine whether a given coefficient of $\mathbf{z}$ takes one of the following critical values: $-\gamma_1, -\gamma_1 - 1, \gamma_1 - 1, \gamma_1$, or any other value. Notably, a rejection occurs if any coefficient of $\mathbf{z}$ hits these values during the Dilithium signing process.

Figure 6a depicts the relationship between the profiling set size of this function and the attack success rate (The model has epochs set to 4 and a batch size of 512), and Figure 6b presents the confusion matrix when the attack set size is 50 000.



(a) Profiling set size vs. attack accuracy.



(b) The confusion matrix.

Figure 6: Attack result for distinguishing the coefficient values $-\gamma_1, -\gamma_1 - 1, \gamma_1 - 1, \gamma_1$, and other values.

5.5 Countermeasures

There are different potential countermeasures to eliminate or mitigate this threat. Masking countermeasures proposed in recent literature, including but not limited to [BBE+18, MGTF19, ABC+23, CGL+24], are expected to provide provably secure protection against this rejected challenge attack. However, as mentioned before, the disadvantage of masking countermeasures is the performance penalty.

Apart from the masking, one can also implement classical hiding countermeasures, e.g., increasing the noise level during the bound check of $\mathbf{z}$ and handling of c. They can be done by adding time noise (for instance, variable clock, jitters, random delay) or balancing power consumption while processing different target data ($\mathbf{z}$ and c in this case) values. A shuffling countermeasure can be another way to mitigate the attack [RPBC20, LHL+23], i.e., using a shuffled order to process $\mathbf{z}$ and c, an attacker then needs to additionally recover the shuffling sequence after recovering the correct values of the rejected $\mathbf{z}_i$ and c. In conclusion, a combination of those mitigation countermeasures can render the attack complexity much higher, while it can only mitigate, but might not fix, the potential security issue completely.

6 Conclusion and future work

Existing SCA research on ML-DSA/Dilithium mainly focuses on direct leakages of sensitive data, such as the nonce $\mathbf{y}$, polynomial multiplications $c\mathbf{s}_1$ and $c\mathbf{s}_2$, and the decomposed commitment $\mathbf{w}_0$. The rejection sampling mechanism prevents secret key leakage by discarding responses $\mathbf{z}$ that fail the bound check.

In this work, we introduced a rejected challenge attack to recover the private key $\mathbf{s}_1$, exploiting side-channel leakages from rejected $\mathbf{z}$ values and their corresponding challenges c. This is formulated as an ILP problem with bounded $c\mathbf{s}_1$ constraints derived from the rejected $\mathbf{z}$ and c.

Our experiments across all security levels of Dilithium-2/3/5 validate the effectiveness of this approach, demonstrating the feasibility of private key recovery with sufficient rejected $\mathbf{z}_i$ and challenge pairs. Real-world experiments on a Dilithium-2 implementation on a 32-bit ARM Cortex-M4 further confirm the practicality of this attack.

As a possible future direction, we plan to enhance the attack's tolerance to generic SCA errors in the recovered $\mathbf{z}_i$ and c coefficients, particularly in low-SNR scenarios, where only a single attack trace is available. Additionally, we should investigate countermeasures to mitigate the proposed attacks and strengthen the security of real-world implementations.

Acknowledgments

The authors would like to thank the Eurocrypt 2025 anonymous reviewers for their helpful feedback.

References

[ABC+23] Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Tobias Schneider, Markus Schönauer, François-Xavier Standaert, and Christine van Vredendaal. Protecting dilithium against leakage: Revisited sensitivity analysis and improved implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023, Issue 4:58–79, 2023.

[AOVCG25] Paco Azevedo-Oliveira, Andersson Calle Viera, Benoît Cogliati, and Louis Goubin. Finding a polytope: A practical fault attack against dilithium. Cryptology ePrint Archive, Paper 2025/195, 2025.

[BAE+24] Olivier Bronchain, Melissa Azouaoui, Mohamed ElGhamrawy, Joost Renes, and Tobias Schneider. Exploiting small-norm polynomial multiplication with physical attacks: Application to crystals-dilithium. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 024 No. 2:359–383, 2024.

[BBD+23] Manuel Barbosa, Gilles Barthe, Christian Doczkal, Jelle Don, Serge Fehr, Benjamin Grégoire, Yu-Hsuan Huang, Andreas Hülsing, Yi Lee, and Xiaodi Wu. Fixing and mechanizing the security proof of fiat-shamir with aborts and dilithium. Springer-Verlag, 2023.

[BBE+18] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. Masking the glp lattice-based signature scheme at any order. In *EUROCRYPT (2)*, pages 354–384. Springer, 2018.

[BDK+20] Shi Bai, Láo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS–dilithium: Algorithm specification and supporting documentation (version 3.1). Round-3 :ubmission to the NIST Post-Quantum Cryptography Standardization Project, 2020. https://cryptojedi.org/papers/#dilithiumnistr3.

[BG14] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, volume 8366 of *Lecture Notes in Computer Science*, pages 28–47. Springer, 2014.

[Bot24] Botan. Botan, a Crypto and TLS for Modern C++ library, Version: 3.5.0. Available at https://github.com/randombit/botan/blob/master/src/lib/pubkey/dilithium/dilithium_common/dilithium_algos.cpp, 2024.

[BVC+23] Alexandre Berzati, Andersson Calle Viera, Maya Chartouny, Steven Madec, Damien Vergnaud, and David Vigilant. Exploiting intermediate value leakage in dilithium: A template-based approach. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023, Issue 4:188–210, 2023.

[CGL+24] Jean-Sébastien Coron, François Gérard, Tancrède Lepoint, Matthias Trannoy, and Rina Zeitoun. Improved high-order masked generation of masking vector and rejection sampling in dilithium. *IACR Cryptol. ePrint Arch.*, page 1149, 2024.

[CKA+21] Zhaohui Chen, Emre Karabulut, Aydin Aysu, Yuan Ma, and Jiwu 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*, pages 583–590. IEEE, 2021.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. 11693:356–383, 2019.

[DFPS23] Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A detailed analysis of fiat-shamir with aborts. Springer-Verlag, 2023.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *Transactions on Cryptographic Hardware and Embedded Systems*, 2018, Issue 1:238–268, 2018.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[FSK03] Daniel L. Fulks, Michael K. Staton, and Leonard J. Kazmier. *Business Statistics : Based on Schaum's Outline of Theory and Problems of Business Statistics*. McGraw-Hill, 2003.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[JMW24] Kelsey A. Jackson, Carl A. Miller, and Daochen Wang. Evaluating the security of crystals-dilithium in the quantum random oracle model. Springer-Verlag, 2024.

[Jr.83] Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

[KAA21] Emre Karabulut, Erdem Alkim, and Aydin 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*, pages 35–45. IEEE, 2021.

[Kel60] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KLH+20] Il-Ju Kim, Taeho Lee, Jaeseung Han, Bo-Yeon Sim, and Dong-Guk Han. Novel single-trace ML profiling attacks on NIST 3 round candidate dilithium. *IACR Cryptol. ePrint Arch.*, page 1383, 2020.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In *EUROCRYPT (3)*, pages 552–586. Springer, 2018.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[KPR+] Matthias J. Kannwischer, Richard Petri, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4. https://github.com/mupq/pqm4.

[LHL+23] Jonghyeok Lee, Jaeseung Han, Sangyub Lee, Jihoon Kwon, Keon-Hee Choi, Jae-Won Huh, Jihoon Cho, and Dong-Guk Han. Systematization of shuffling countermeasures: With an application to crystals-dilithium. *IEEE Access*, 11:142862–142873, 2023.

[Lib24] Liboqs. Liboqs, the open quantum safe quantum-safe cryptographic algorithms library, Version: 0.10.1. Available at https://github.com/open-quantum-safe/liboqs/blob/main/src/sig/ml_dsa/pqcrystals-dilithium-standard_ml-dsa-44-ipd_ref/poly.c, 2024.

[LLZ+24] Yong Liu, Yuejun Liu, Yongbin Zhou, Yiwen Gao, Zehua Qiao, and Huaxin Wang. A novel power analysis attack against crystals-dilithium implementation. In *IEEE European Test Symposium, ETS 2024, The Hague, Netherlands, May 20-24, 2024*, pages 1–6. IEEE, 2024.

[LMSK63] John D. C. Little, Katta G. Murty, Dura W. Sweeney, and Caroline Karel. An algorithm for the traveling salesman problem. *Operations Research*, 11(6):972–989, 1963.

[Lyu09] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, 2012.

[LZ19] Qipeng Liu and Mark Zhandry. Revisiting post-quantum fiat-shamir. 11693:326–355, 2019.

[LZS+21] Yuejun Liu, Yongbin Zhou, Shuo Sun, Tianyu Wang, Rui Zhang, and Jingdian Ming. On the security of lattice-based fiat-shamir signatures in the presence of randomness leakage. *IEEE Trans. Inf. Forensics Secur.*, 16:1868–1879, 2021.

[MGTF19] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking dilithium - efficient implementation and side-channel evaluation. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*, volume 11464 of *Lecture Notes in Computer Science*, pages 344–362. Springer, 2019.

[NIS24] NIST. FIPS 204, Module-Lattice-Based Digital Signature Standard (FIPS 204). Available at https://doi.org/10.6028/NIST.FIPS.204, 2024.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[QLZ+23a] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Jingdian Ming, Chengbin Jin, and Huizhong Li. Practical public template attack attacks on crystals-dilithium with randomness leakages. *IEEE Trans. Inf. Forensics Secur.*, 18:1–14, 2023.

[QLZ+23b] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Mingyao Shao, and Shuo Sun. When NTT meets SIS: efficient side-channel attacks on dilithium and kyber. *IACR Cryptol. ePrint Arch.*, page 1866, 2023.

[QLZ+24] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Yuhan Zhao, and Shuyi Chen. Single trace is all it takes: Efficient side-channel attack on dilithium. *IACR Cryptol. ePrint Arch.*, page 512, 2024.

[RCDB24] Prasanna Ravi, Anupam Chattopadhyay, Jan-Pieter D'Anvers, and Anubhab Baksi. Side-channel and fault-injection attacks over lattice-based post-quantum schemes (kyber, dilithium): Survey and new results. *ACM Trans. Embed. Comput. Syst.*, 23(2):35:1–35:54, 2024.

[RJH+18] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. Side-channel assisted existential forgery attack on dilithium - A NIST PQC candidate. *IACR Cryptol. ePrint Arch.*, page 821, 2018.

[RPBC20] Prasanna Ravi, Romain Poussier, Shivam Bhasin, and Anupam Chattopadhyay. On configurable SCA countermeasures against single trace attacks for the NTT - A performance evaluation study over kyber and dilithium on the ARM cortex-m4. In Lejla Batina, Stjepan Picek, and Mainack Mondal, editors, *Security, Privacy, and Applied Cryptography Engineering - 10th International Conference, SPACE 2020, Kolkata, India, December 17-21, 2020, Proceedings*, volume 12586 of *Lecture Notes in Computer Science*, pages 123–146. Springer, 2020.

[RR23] Victor Reis and Thomas Rothvoss. The subspace flatness conjecture and faster integer programming. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 974–988. IEEE, 2023.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[SLKG23] Hauke Malte Steffen, Georg Land, Lucie Johanna Kogelheide, and Tim Güneysu. Breaking and protecting the crystal: Side-channel analysis of dilithium in hardware. In Thomas Johansson and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023, College Park, MD, USA, August 16-18, 2023, Proceedings*, volume 14154 of *Lecture Notes in Computer Science*, pages 688–711. Springer, 2023.

[TMS24] Tolun Tosun, Amir Moradi, and Erkay Savas. Exploiting the central reduction in lattice-based cryptography. *IACR Cryptol. ePrint Arch.*, page 66, 2024.

[TS24] Tolun Tosun and Erkay 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.*, 19:3353–3365, 2024.

[UMTS22] Vincent Quentin Ulitzsch, Soundes Marzougui, Mehdi Tibouchi, and Jean-Pierre Seifert. Profiling side-channel attacks on dilithium - A small bit-fiddling leak breaks it all. In Benjamin Smith and Huapeng Wu, editors, *Selected Areas in Cryptography - 29th International Conference, SAC 2022, Windsor, ON, Canada, August 24-26, 2022, Revised Selected Papers*, volume 13742 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2022.

[vzGS78] Joachim von zur Gathen and Malte Sieveking. A bound on solutions of linear integer equalities and inequalities. *Proceedings of the American Mathematical Society*, 72(1):155–158, 1978.

[WGL+24] Huaxin Wang, Yiwen Gao, Yuejun Liu, Qian Zhang, and Yongbin Zhou. In-depth correlation power analysis attacks on a hardware implementation of crystals-dilithium. *Cybersecur.*, 7(1):21, 2024.

[WNGD23] Ruize Wang, Kalle Ngo, Joel Gärtner, and Elena Dubrova. Single-trace side-channel attacks on crystals-dilithium: Myth or reality? *IACR Cryptol. ePrint Arch.*, page 1931, 2023.

Appendix

A Concrete Hamming weights of rejected coefficient $\mathbf{z}_i$

For each security level of Dilithium-2/3/5, we analyzed the histogram of the Hamming weights of those $4\beta + 1$ possible values of rejected coefficient $\mathbf{z}_i$. For Dilithium-2, HW-15/31/21/25 are unique ones when the rejected $\mathbf{z}_i$ is in the interval of $[-\gamma_1+1-\beta, -\gamma_1+1+\beta]$, however, HW-21/25 were never observed during our analyses of the distribution of $\mathbf{x}_i$ shown in 1a. Thus, only HW-15/31 corresponding to $-\gamma_1$ and $-\gamma_1 - 1$ in this interval are considered for our rejected challenge attack in this work. Similarly, HW-17/1/7/11 are unique ones when the rejected $\mathbf{z}_i$ is in the interval of $[\gamma_1 - \beta, \gamma_1 + \beta]$, however, HW-7/11

were never observed during our analyses of the distribution of $\mathbf{x}_i$ shown in 1a. Thus, only HW-17/1 corresponding to $\gamma_1 - 1$ and γ_1 in this interval are considered for our rejected challenge attack in this work.

For Dilithium-3/5, only HW-13/31 corresponding to $-\gamma_1$ and $-\gamma_1 - 1$ in the interval of $[-\gamma_1 + 1 - \beta, -\gamma_1 + 1 + \beta]$, and HW-19/1 corresponding to $\gamma_1 - 1$ and γ_1 in the interval of $[\gamma_1 - \beta, \gamma_1 + \beta]$ are considered for our rejected challenge attack. Because they are the only ones with unique HW values in those two intervals, respectively.

We summarized the histogram of HW values of those $4\beta + 1$ possible values of rejected coefficient $\mathbf{z}_i$ in Tables 5 to 10.

Table 5: HW corresponding to 2β possible rejected $\mathbf{z}_i$ coefficient in the interval of $[-\gamma_1 + 1 - \beta, -\gamma_1 + 1 + \beta]$ for Dilithium-2

HW	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
Frequency	1	7	19	26	18	6	1	1	6	17	26	19	7	1

Table 6: HW corresponding to $2\beta + 1$ possible rejected $\mathbf{z}_i$ coefficient in the interval of $[\gamma_1 - \beta, \gamma_1 + \beta]$ for Dilithium-2

HW	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Frequency	1	7	19	26	19	6	1	1	6	18	26	19	7	1

Table 7: HW corresponding to 2β possible rejected $\mathbf{z}_i$ coefficient in the interval of $[-\gamma_1 + 1 - \beta, -\gamma_1 + 1 + \beta]$ for Dilithium-3

HW	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
Frequency	1	8	28	52	56	36	13	2	2	13	36	55	52	28	8	1

Table 8: HW corresponding to $2\beta + 1$ possible rejected $\mathbf{z}_i$ coefficient in the interval of $[\gamma_1 - \beta, \gamma_1 + \beta]$ for Dilithium-3

HW	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
Frequency	1	8	28	53	56	36	13	2	2	13	36	56	52	28	8	1

Table 9: HW corresponding to 2β possible rejected $\mathbf{z}_i$ coefficient in the interval of $[-\gamma_1 + 1 - \beta, -\gamma_1 + 1 + \beta]$ for Dilithium-5

HW	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
Frequency	1	7	21	35	34	18	4	3	18	34	35	21	7	1

Table 10: HW corresponding to $2\beta + 1$ possible rejected $\mathbf{z}_i$ coefficient in the interval of $[\gamma_1 - \beta, \gamma_1 + \beta]$ for Dilithium-5

HW	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
Frequency	1	7	21	35	35	18	4	4	18	34	35	21	7	1