

When Module Lattice Leaks: Horizontal Fusion Attacks on ML-DSA Implementation*

Yuhan Zhao
School of Cyber Science and
Engineering, Nanjing University of
Science and Technology
Nanjing, China
yhzha096@njust.edu.cn

Dalin He
School of Cyber Science and
Engineering, Nanjing University of
Science and Technology
Nanjing, China
hedalin@njust.edu.cn

Wei Cheng^{†‡}
School of Cyber Science and
Engineering, Nanjing University of
Science and Technology
Nanjing, China
wei.cheng@njust.edu.cn

Yuejun Liu
School of Cyber Science and
Engineering, Nanjing University of
Science and Technology
Nanjing, China
liuyuejun@njust.edu.cn

Jingdian Ming
School of Cyber Science and
Engineering, Nanjing University of
Science and Technology
Nanjing, China
mingjingdian@njust.edu.cn

Yongbin Zhou^{*§}
School of Cyber Science and
Engineering, Nanjing University of
Science and Technology
Nanjing, China
zhouyongbin@njust.edu.cn

Abstract

The standardization of ML-DSA has shifted the cryptographic community's focus toward its practical security. While profiled attacks against its implementations are well studied with a few traces, non-profiling attacks are widely assumed to require large trace complexity. We challenge this by introducing horizontal fusion attacks, demonstrating that non-profiling, few-trace key recovery is highly practical against ML-DSA, even against masked implementations.

We expose a structural vulnerability in the module lattice from a side-channel perspective. In particular, in ML-DSA's matrix-vector multiplication ($\hat{A} \circ \hat{y}$), the row-wise reuse of the ephemeral secret vector $\hat{y}$ indicates that one single signature generation exposes k (the row-wise size of $\hat{A}$) leakage instances of the same secret-dependent intermediate value, each with a distinct and known coefficient of $\hat{A}$. However, exploiting this in practice is highly non-trivial due to noise and/or compiler optimizations. To overcome this, we propose a variance-based weighted fusion strategy. This approach weights each operation by how far its leading candidate is separated from the runner-up candidates, and the signing relation ($y = z - c \cdot s_1$) lets us extract the signed coefficients of the secret key. Moreover, we introduce a fast, INTT-based algebraic sieve that further increases the success rate of key recovery.

Putting together, we achieve full key recovery using merely 4 traces against ML-DSA-87 (the highest security parameter set) with non-profiling correlation analysis. On the state-of-the-art first-order masked ML-DSA implementation, our attack extracts the secret key using no more than 90 traces. To the best of our knowledge, these non-profiling results establish a new record in trace complexity for both unprotected and first-order masked ML-DSA implementations, even comparable to these profiling-based attacks.

Keywords

Post-quantum cryptography, Side-channel analysis, ML-DSA, Horizontal attack, Leakage fusion, Number theoretic transform

1 Introduction

The standardization of the Module-Lattice-Based Digital Signature Algorithm (ML-DSA) as FIPS 204 [30], derived from CRYSTALS-Dilithium (abbreviated as Dilithium) [16], represents a defining step in the global transition to post-quantum cryptography (PQC). The security of this primitive is grounded in the worst-case hardness of the Module Learning With Errors (M-LWE), Module Short Integer Solution (M-SIS), and related Fiat-Shamir-with-aborts assumptions [22, 24]. Within the random-oracle and quantum-random-oracle models, these mathematical problems provide robust asymptotic guarantees against adversaries equipped with large-scale quantum computing capabilities. However, the deployment of the cryptographic algorithms onto practical embedded systems, ranging from high-performance CPUs to constrained microcontrollers, introduces a critical dissonance between theoretical security and practical resilience. The theoretical guarantees of cryptographic indistinguishability usually collapse once an algorithm is instantiated on a CMOS substrate. More precisely, the conditioned branching or the movement of data across logic gates and/or the switching of transistors inevitably modulate physical observables, such as power consumption and electromagnetic radiation [18, 23, 28], leading to side-channel analysis (SCA) that can reveal information about internal states or secrets.

Motivation. Within physical CMOS implementations, the execution of any intermediate algorithmic state inherently generates an observable side-channel vector. The landscape of side-channel analysis against post-quantum cryptography is typically categorized into two classes: *profiling* (including both secret-based profiling and public-based profiling) and *non-profiling* attacks. The profiling attacks, utilizing templates [10] or machine learning, currently define the upper bound of key-recovery efficiency. Recent profiling advancements with deep learning techniques have demonstrated high

*This is the full version of the paper accepted at ACM CCS 2026. DOI of the conference version: <https://doi.org/10.1145/3830454.3846766>.

[†]Corresponding author.

[‡]Also with LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France.

[§]Also with Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.

trace efficiency, even achieving single-trace attacks [34, 41, 43, 45]. On ML-DSA (including Dilithium), the profiling attacks can recover the secret keys in a few traces [27, 35, 45]. However, the efficacy of profiling attacks relies on a strong, often unrealistic assumption: that the adversary possesses a fully controllable clone of the target device to construct accurate leakage templates or models. In real-world adversarial settings, this assumption usually does not hold. Device-specific variations in process, voltage, and temperature often render templates non-portable, largely degrading the attack's success probability of secret key recovery.

By contrast, non-profiling attacks (e.g., correlation power analysis [7], CPA) operate without a priori templates, representing a strictly non-profiling, practical threat model. Yet, existing non-profiling literature has predominantly focused on the pointwise multiplication involving the challenge c . While this operation provides a clean leakage model, it provides only a limited number of exploitable leakage instances for each target coefficient in one signing execution. This necessitates the acquisition of thousands of traces to overcome measurement noise or countermeasure effects. Therefore, it remains to construct more efficient, few-trace attacks against ML-DSA in a more realistic non-profiling setting. In this work, we focus on purely non-profiling attacks against both unprotected and masked ML-DSA, and explore new vulnerabilities from a side-channel perspective.

New Techniques. We introduce a *structural horizontal attack* approach to dissect the algebraic features of the underlying module ring that can cause severe side-channel leakage. To the best of our knowledge, existing non-profiling attacks against ML-DSA mainly target the pointwise multiplication in the Number Theoretic Transform (NTT) domain. We therefore shift the focus to a broader structural analysis and to efficient non-profiling recovery of s_1 . Our approach consists of three components that can exploit the algebraic properties of ML-DSA as detailed below.

First, we propose to exploit repeated side-channel leakage from the matrix-vector product in the module lattice of ML-DSA. Specifically, during the execution of ML-DSA, the device computes the matrix-vector product to obtain w , namely $w = \hat{A} \circ \hat{y}$, where A is a $k \times \ell$ public matrix and y is the ephemeral secret vector. Unlike the NTT-domain product $\hat{c} \circ \hat{s}_1$, the vector y is reused across k distinct inner products (e.g., $k = 4, 6, 8$ for ML-DSA-44, -65 and -87, respectively). From a side-channel perspective, the module rank k defines how many times the same ephemeral coefficient is processed within one signing trial. By targeting this operation, the same ephemeral coefficient contributes to multiple row-wise leakage instances within a single execution trace, modulated by different, known rows of A . As a result, in this work, we jointly exploit those repeated leakages to construct more efficient attacks.

Second, we introduce an LD-guided leakage-fusion to jointly utilize side-channel leakage from multiple internal variables. Rather than combining all available operations evenly, each target operation is treated as producing a key candidate score (correlation coefficients) distribution for every secret coefficient. The *leading degree* (LD) metric, originally proposed for model performance evaluation in deep-learning-based SCA [46], is adapted as an unsupervised metric for selecting fusion weights. This allows the attack

to assign large weights to operations whose score distributions exhibit a clear leading candidate, while suppressing operations whose scores are dominated by variations or false peaks.

Third, we present a dual-domain algebraic sieve as post-processing to resolve the residual ambiguity left by the primary attacks. The attack ranks candidates in the NTT domain, whereas the range constraints of ML-DSA secret key are defined in the polynomial coefficient domain. Therefore, we introduce an INTT-based sieve to check the small-norm constraint $\|\text{INTT}(\hat{s}'_1)\|_\infty \leq \eta$. Because of the linear diffusion property of INTT, an error in one NTT-domain coefficient diffuses across multiple polynomial coefficients; thus, incorrect secret key candidates will satisfy this constraint with negligible probability. This INTT-based sieve serves as a lightweight, exact algebraic filter for the key candidates produced by our LD-guided fusion. Crucially, it provides a highly efficient alternative to exhaustive post-processing methods when the candidate lists already place the correct key within a small enumeration frontier.

Contributions. Our approach advances the efficiency of non-profiling side-channel analysis through the following contributions:

- We present a *structural horizontal attack* that exploits the structural feature of the module lattice against ML-DSA. Our approach targets the NTT-domain matrix-vector product $\hat{A} \circ \hat{y}$. By utilizing the signing relation $\hat{y} = \hat{z} - \hat{c} \circ \hat{s}_1$, our approach transforms the repeated leakage of $\hat{y}$ into hypotheses on the secret key s_1 that are parameterized by public signature data.
- We develop an LD-guided leakage-fusion approach. By adapting the leading degree metric [46] as an unsupervised heuristic, our approach dynamically fuses the leakage from multiple internal variables, including $\hat{A} \circ \hat{y}$, $\hat{c} \circ \hat{s}_1$, and $\text{INTT}(\hat{c} \circ \hat{s}_1)$.
- We construct an INTT-based sieve that verifies NTT-domain key candidates by utilizing the inherent small-norm constraint. This backend resolves residual coefficient ambiguity without invoking BKZ, SIS-style lattice reductions.
- We validate our findings on ARM Cortex-M4 STM32F405 targets under -O3&-Otime optimization. The attack recovers s_1 within 6/4/4 traces against unprotected ML-DSA-44/65/87, and with 136/114/90 traces against a state-of-the-art first-order masked implementation, respectively.

To the best of our knowledge, these results establish a new state-of-the-art in non-profiling attack efficiency, especially against ML-DSA-87. As a key finding, our evaluation highlights a critical dichotomy between theoretical and physical security scaling. We demonstrate that while theoretical strength increases from ML-DSA-44 to -87, physical resilience decreases. ML-DSA-87 is more vulnerable because its larger module dimensions force greater algorithmic repetition of the ephemeral variable, validating that our exposed structural vulnerabilities scale proportionally with the parameter sizes.

2 Related Work

Side-channel analysis against ML-DSA implementations [30] differ primarily by adversary types and leakage source. Table 1 categorizes those attacks into three classes. In particular, the *secret-profiling*

Table 1: Comparison with state-of-the-art side-channel attacks on ML-DSA/Dilithium. Note that D2/D3/D5 denote ML-DSA-44, -65 and -87, respectively

Work	Target	Attack Type	Platform	Compiler	Sampling Rate	Methodology	Traces (D2 / D3 / D5)
<i>Target: Unprotected Implementation</i>							
Fan et al. [17]	$\hat{A} \circ \hat{y}, \hat{c} \circ \hat{s}_1$	Secret-Prof.	STM32F407	N/A	125 MS/s	Multivariate Template	15 / 11 / 9
Sun et al. [37]	w_0 , hints h	Secret-Prof.	STM32F407	N/A	125 MS/s	MLP + DBDD Solver	305 / - / -
Zhou et al. [45]	Rejected $\hat{c}, \hat{z}$	Secret-Prof.	STM32F	N/A	62.5 MS/s	MLP + ILP	3×10^6 / 2.3×10^7 / 1.3×10^7
Liu et al. [27]	$\hat{c} \circ \hat{s}_1, \text{INTT}(\hat{c} \circ \hat{s}_1)$	Secret-Prof.	STM32F407	N/A	125 MS/s	Soft Analytic SCA	4 / 13 / 28
Hermelink et al. [19]	y	Secret-Prof.	STM32F415 (STM32F303)	-O3&-Ospace	100 MS/s	Soft Analytic SCA	247 / - / - (27.6 / - / -)
Qiao et al. [35]	$\text{INTT}(\hat{c} \circ \hat{s}_1)$	Public-Prof.	STM32F405	N/A	100 MS/s	Public-based Template	10 / 27 / 18
Zhao et al. [44]	$\hat{c} \circ \hat{s}_1$	Public-Prof.	STM32F405	N/A	100 MS/s	PTA + CPA	300 / - / -
Chen et al. [11]	$\hat{c} \circ \hat{s}_1$	Non-Prof.	STM32F405	-O3	1.25 GS/s	Standard CPA	157 / - / -
Qiao et al. [33]	$\hat{c} \circ \hat{s}_1$	Non-Prof.	STM32F405	-O3*	100 MS/s	CPA + SIS Solver	5,000 / 5,000 / 5,000
Tosun et al. [39]	$\hat{c} \circ \hat{s}_1$	Non-Prof.	MAX32520	N/A	10 GS/s	ZV-FA CPA	- / $220^{\S}$ / -
Tosun et al. [38]	$\hat{c} \circ \hat{s}_1$	Non-Prof.	STM32F303	-O0	29.2 MS/s [§]	Standard CPA	- / 35 / -
This work	$\hat{A} \circ \hat{y}, \hat{c} \circ \hat{s}_1,$ $\text{INTT}(\hat{c} \circ \hat{s}_1)$	Non-Prof.	STM32F405	-O3&-Otime	100 MS/s	LD-Guided Fusion CPA + INTT Sieve	6 / 4 / 4
<i>Target: First-Order Masked Implementation of Coron et al. [12]</i>							
Hermelink et al. [19]	y	Secret-Prof.	STM32F415	-O3&-Ospace	100 MS/s	Soft Analytic SCA	700 / - / -
Tosun et al. [38]	$\hat{c} \circ \hat{s}_1$	Non-Prof.	STM32F303	-O0	29.2 MS/s [§]	HOCAPA	- / $350^{\parallel}$ / -
This work	$\hat{A} \circ \hat{y}, \hat{c} \circ \hat{s}_1,$ $\text{INTT}(\hat{c} \circ \hat{s}_1)$	Non-Prof.	STM32F405 (STM32F303)	-O3&-Otime	100 MS/s	LD-Guided Fusion HOCAPA + INTT Sieve	136 / 114 / 90 (124 / - / 74)

- Secret-Prof., Public-Prof., and Non-Prof. denote secret-based profiling, public-information-based profiling/template (PTA), and non-profiling attacks, respectively.
- Distorted bounded distance decoding is abbreviated as DBDD in this table and throughout the remainder of the paper.
- [‡] Excludes approximately 1.5×10^6 traces processed for zero-value pre-filtering.
- [§] Acquisition by lowering the running frequency to 7.3 MHz on target, and four samples per clock cycle ($4 \times$ sampling frequency).
- ^{||} Cannot resolve the sign of the secret key due to symmetric prediction functions; relies on another auxiliary leakage source for $\pm$ disambiguation.

attacks require a prior characterization phase using secret-related intermediate values in a clone device. The *public-profiling* attacks utilize a template-construction phase where labels are derived from public variables, such as public inputs, requiring profiling without secret-related labels and clone device(s). By contrast, the *non-profiling* attacks eliminate the profiling phase, recovering the secret key directly from collected traces on the target device.

Table 1 positions our work compared with prior attacks against ML-DSA (including Dilithium¹). To the best of our knowledge, all existing non-profiling attacks target a single operation such as $\hat{c} \circ \hat{s}_1$, $\text{INTT}(\hat{c} \circ \hat{s}_1)$, or rejection-related leakage. In contrast, our attack exploits the repeated matrix-vector product $\hat{A} \circ \hat{y}$ and links its ephemeral leakage to the long-term secret through $\hat{y} = \hat{z} - \hat{c} \circ \hat{s}_1$, enabling non-profiling fusion and algebraic candidate verification.

2.1 Profiling Attacks against ML-DSA

The profiling attacks can be further categorized into two sub-classes, namely the secret-based profiling and the public-based profiling analysis. On the one hand, the secret-profiling adversaries characterize target leakage via a clone device, statistical templates, or learning models prior to extraction [4, 10]. Fan et al. [17] combine $\hat{A} \circ \hat{y}$ and $\hat{c} \circ \hat{s}_1$ using a multivariate template attack, reporting recovery of the secret key of Dilithium-2/3/5 with 15/11/9 traces. Ulitzsch et al. [40] performed a profiled attack targeting the bit-unpacking of the ephemeral vector y , requiring approximately 756,000 signatures for Dilithium-2. Sun et al. [37] target w_0 and hints h utilizing a multi-layer perceptron paired with a DBDD solver, requiring 305

traces for Dilithium-2. Zhou et al. [45] target rejection logic, recovering keys with single traces after an offline profiling phase requiring 3×10^6 traces.

The soft analytical SCA (SASCA) and generic solvers have also been applied to profiling scenarios. Bronchain et al. [8] applied SASCA and belief propagation (BP) to small-norm polynomial multiplication, requiring less than 10 traces at a Signal-to-Noise Ratio (SNR) of 100, and under 120 traces at an SNR of 0.1 for unprotected implementations. Liu et al. [27] target $\hat{c} \circ \hat{s}_1$ and $\text{INTT}(\hat{c} \circ \hat{s}_1)$ with SASCA, reporting 4/13/28 traces for Dilithium-2/3/5. Hermelink et al. [20] proposed a generic solver framework for side-channel hints in LWE-based cryptosystems.

On the other hand, adversaries operating without a clone device have to rely on public data templates or direct statistical extraction. Liu et al. [26] and Qiao et al. [32] first proposed the public template attacks (PTA), demonstrating that an adversary can characterize leakage directly on the target device using public inputs with randomness leakages. This approach has been extended to exploit the rejection sampling mechanism of ML-DSA. Zhao et al. [44] utilized public templates derived from accepted and rejected signing trials to recover ML-DSA-44 keys using 300 traces. Similarly, Qiao et al. [35] targeted $\text{INTT}(\hat{c} \circ \hat{s}_1)$ via CNN-based public templates, requiring 10/27/18 traces against Dilithium-2/3/5.

2.2 Non-profiling Attacks against ML-DSA

Strictly non-profiling attacks bypass the profiling phase entirely, relying on direct statistical correlation between hypothetical leakage and the physical observations.

In particular, Chen et al. [11] provided the first practical non-profiling CPA attack against CRYSTALS-Dilithium. They applied

¹Throughout this work, we refer to the standardized parameter sets as ML-DSA-44, -65, and -87. When evaluating legacy or pre-standardization implementations, we use their corresponding Round 3 designations: Dilithium-2, -3, and -5, respectively.

correlation power analysis (CPA) to target the pointwise multiplication $\hat{c} \circ \hat{s}_1$ during signing and used a two-stage strategy to reduce the hypothesis space, achieving key recovery against Dilithium-2 with 157 traces. Liu et al. [25] further optimized this two-stage CPA, mainly improving the computational efficiency of the attack procedure while retaining the same pointwise multiplication target.

Qiao et al. [33] also targeted NTT-domain leakage but combined CPA with algebraic recovery. Their attack observes leakage from pointwise multiplication and formulates the remaining uncertainty as an SIS instance. This reduces the amount of side-channel information that must be recovered directly for each coefficient, and achieves full key recovery for Dilithium-2/3/5 with 5,000 traces.

To accelerate statistical convergence without the aid of templates, subsequent research has focused on filtering techniques and implementation-specific vulnerabilities. Tosun et al. [39] introduced a zero-value filtering strategy for $\hat{c} \circ \hat{s}_1$. By isolating specific leakage characteristics, this approach enhanced candidate distinguishability, reducing the required number of traces to 220 against Dilithium-3. In a follow-up evaluation targeting an unoptimized -00 compilation, Tosun et al. [38] reported secret key recovery with 35 traces against an unprotected Dilithium-3 (on an ARM Cortex M4 STM32F303 device).

Despite those optimizations, existing non-profiling attacks predominantly target isolated arithmetic operations. Because computations such as $\hat{c} \circ \hat{s}_1$ process each target secret coefficient exactly once per signature generation, the exploitable leakage volume per trace is intrinsically constrained. Consequently, to compensate for this limited distinguishability, adversaries operating in a strictly non-profiling model are traditionally forced to accept higher trace complexities. Bridging this data-efficiency gap by exploiting denser structural leakage of the module lattice, such as the matrix-vector product, without invoking a computationally expensive profiling phase remains a central open challenge in evaluating the practical security of ML-DSA. More importantly, it remains an open question which ML-DSA operation constitutes the optimal physical leakage source, and how to systematically exploit it to construct more efficient non-profiling attacks.

2.3 Masked Implementations and Attacks

Masking is one of the primary implementation-level countermeasures against side-channel analysis for lattice-based cryptography [2, 31]. For instance, Migliore et al. [29] introduced masked implementations of Dilithium and evaluated their leakage behavior, while del Pino et al. [15] developed high-order masking of lattice signatures in quasilinear time. For ML-DSA, Coron et al. [12] provide a widely evaluated first-order protected implementation, improving the masked generation of the masking vector and rejection sampling. More recently, SUCRE [3] hardens the rejection-sampling interface by combining shuffling and unmasking. These works address the leakage induced by the rejection logic and masked intermediate generation, while the upstream polynomial arithmetic remains an important target for evaluation.

In the profiling setting, Hermelink et al. [19] target the ephemeral vector $\mathbf{y}$ in masked ML-DSA. Their attack reports 700 traces on an STM32F415 target against Dilithium-2. Damm et al. [13] formulate the leakage from the Boolean-to-arithmetic conversion in masked

Dilithium as a concealed ILWE problem. Their attack uses a profiled classifier to obtain leakage information and then solves the resulting structured instance. The reported key recovery requires 590,000 signatures for Dilithium-2 and 2.55×10^6 signatures for Dilithium-3 against the first-order masked implementation.

In the non-profiling setting, Tosun et al. [38] evaluate first-order masked lattice implementations using higher-order CPA. For Dilithium, they target the pointwise multiplication $\hat{c} \circ \hat{s}_1$ and report 350 traces against Dilithium-3. Their attack uses an optimal prediction function (OPF), which selects a leakage prediction adapted to the central-reduction behavior of the implementation rather than relying on a standard Hamming-weight model. Combined with centered-product preprocessing, this OPF improves distinguishability in the masked setting. The resulting model is effective for recovering absolute-value information, but the induced sign ambiguity remains a relevant issue when targeting masked implementations.

3 Structural Vulnerability in ML-DSA Signing

This section identifies the structural vulnerability exploited in this work. By *structural vulnerability*, we mean repeated observations of the same ephemeral coefficients that arise from the module lattice structure of ML-DSA itself, rather than from device-specific implementation flaws or errors. In particular, the NTT-domain matrix-vector product $\hat{\mathbf{w}} = \hat{\mathbf{A}} \circ \hat{\mathbf{y}}$ reuses the same ephemeral vector $\hat{\mathbf{y}}$ across multiple matrix rows. This reuse is algebraically legitimate by design and does not contradict the theoretical security of ML-DSA. However, once the computation is executed on a physical device, it creates repeated observations on the same secret-dependent coefficients. Such repetition of the same secret-dependent intermediate value constitutes the horizontal leakage that our fusion attack exploits.

We first introduce the notation used throughout this work, then show how row-wise reuse introduces multiple times of observation of each ephemeral coefficient within one signing execution, how the ephemeral leakage is tied to the secret key, and why the same structural vulnerability remains relevant under first-order masking.

Notation and domains. We follow the notation of FIPS 204. The polynomial ring and the NTT representation are

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1), \quad T_q = \prod_{v=0}^{n-1} \mathbb{Z}_q, \quad (1)$$

where $n = 256$, $q = 8, 380, 417$. Elements of T_q are length- n arrays $\hat{\mathbf{a}} = (\hat{a}[0], \dots, \hat{a}[n-1])$ with $\hat{a}[v] \in \mathbb{Z}_q$. Bold lowercase letters denote vectors, bold uppercase letters denote matrices, and hats denote elements in T_q . Table 2 lists the parameters of ML-DSA.

For $\hat{\mathbf{a}}, \hat{\mathbf{b}} \in T_q$, the operator $\circ$ denotes multiplication in T_q :

$$(\hat{\mathbf{a}} \circ \hat{\mathbf{b}})[v] = \hat{a}[v] \cdot \hat{b}[v] \pmod{q}. \quad (2)$$

Thus, $\circ$ is used before coefficient indexing, while $\cdot$ denotes scalar multiplication in $\mathbb{Z}_q$ after indexing. Products without hats, such as $\mathbf{cs}_1$, are products in $\mathcal{R}_q$.

For $\hat{\mathbf{A}} \in T_q^{k \times \ell}$ and $\hat{\mathbf{y}} \in T_q^\ell$, the FIPS notation $\hat{\mathbf{w}} = \hat{\mathbf{A}} \circ \hat{\mathbf{y}}$ means

$$\hat{\mathbf{w}}_i = \sum_{j=0}^{\ell-1} \hat{A}_{i,j} \circ \hat{y}_j, \quad \hat{w}_i[v] = \sum_{j=0}^{\ell-1} \hat{A}_{i,j}[v] \cdot \hat{y}_j[v] \pmod{q}. \quad (3)$$

Table 2: FIPS 204 parameters relevant to row-wise repetition and small-norm candidate verification.

Parameter	ML-DSA-44	ML-DSA-65	ML-DSA-87
Module Rank (k, ℓ)	(4, 4)	(6, 5)	(8, 7)
Secret Key Bound η	2	4	2
$\beta (= \tau\eta)$	78	196	120
Challenge Weight τ	39	49	60

3.1 Attack-Relevant Signing Flow

ML-DSA is defined over $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ with $n = 256$ and $q = 8,380,417$ for all FIPS 204 parameter sets. Its black-box security relies on module lattice assumptions such as M-LWE and M-SIS. We emphasize that our analysis does not contest these underlying assumptions. Instead, it exposes vulnerabilities inherent in the physical instantiation from a side-channel perspective, specifically targeting the hardware execution phase, where ring operations are reduced to coefficient-wise NTT arithmetics.

The signing key contains short polynomial vectors $\mathbf{s}_1 \in S'_\eta$ and $\mathbf{s}_2 \in S_\eta^k$, whose coefficients are bounded by η . The public key is derived from $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$, where $\mathbf{A} \in \mathcal{R}_q^{k \times \ell}$ is public. During signing, the device samples an ephemeral vector $\mathbf{y}$, computes the commitment $\mathbf{w} = \mathbf{A}\mathbf{y}$, derives the challenge c , and forms

$$\mathbf{z} = \mathbf{y} + c\mathbf{s}_1. \quad (4)$$

This signing relation is the bridge used later to connect leakage of the ephemeral vector to hypotheses on $\mathbf{s}_1$.

Algorithm 1 highlights the operations that are relevant to our attack. The first class consists of operations on the ephemeral path, namely $\hat{\mathbf{y}} = \text{NTT}(\mathbf{y})$ and the matrix-vector product $\hat{\mathbf{A}} \circ \hat{\mathbf{y}}$. The second class consists of operations that directly involve the long-term secret, namely $\hat{c} \circ \hat{\mathbf{s}}_1$ and its inverse transform. Existing non-profiling attacks mainly target the latter class. Our attack starts from the former class and uses the public signing relation $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$ to bind ephemeral leakage to $\mathbf{s}_1$. This relation holds for every valid signature irrespective of how the private mask seed ρ'' (used to expand $\mathbf{y}$) is derived, so the binding is unaffected by whether the ML-DSA implementation generates signatures deterministically in normal mode or probabilistically in hedged mode [30].

Small-norm constraints. ML-DSA uses rejection sampling to ensure that the signature distribution remains independent of the secret. A candidate signature is accepted only if the potential signature $\mathbf{z}$ and the low-order bits of the commitment satisfy strict infinity-norm constraints:

$$(\|\mathbf{z}\|_\infty < \gamma_1 - \beta) \wedge (\|\text{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)\|_\infty < \gamma_2 - \beta), \quad (5)$$

where $\beta = \tau\eta$ as in Table 2. In our attack, the same small-norm structure also provides a verification condition for candidate keys: an NTT-domain candidate for $\hat{\mathbf{s}}_1$ must transform back to coefficient values bounded by η .

3.2 Row-Wise Reuse of the Ephemeral Vector

The main structural property exploited in this work is the row-wise reuse of the ephemeral vector inside the NTT-domain matrix-vector

Algorithm 1 ML-DSA.SIGN_INTERNAL with attack-relevant operations highlighted

Require: Secret key sk ; formatted message $M' \in \{0, 1\}^*$; per-message randomness $rnd \in \{0, \dots, 255\}^{32}$

Ensure: Encoded signature σ

```

1:  $(\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0) \leftarrow \text{SKDECODE}(sk)$ 
2:  $\hat{\mathbf{s}}_1 \leftarrow \text{NTT}(\mathbf{s}_1)$ 
3:  $\hat{\mathbf{s}}_2 \leftarrow \text{NTT}(\mathbf{s}_2)$ 
4:  $\hat{\mathbf{t}}_0 \leftarrow \text{NTT}(\mathbf{t}_0)$ 
5:  $\hat{\mathbf{A}} \leftarrow \text{EXPANDA}(\rho)$ 
6:  $\mu \leftarrow H(\text{BYTESToBITS}(tr) \parallel M', 64)$ 
7:  $\rho'' \leftarrow H(K \parallel rnd \parallel \mu, 64)$ 
8:  $\kappa \leftarrow 0$ 
9:  $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$ 
10: while  $(\mathbf{z}, \mathbf{h}) = \perp$  do
11:    $\mathbf{y} \leftarrow \text{EXPANDMASK}(\rho'', \kappa)$ 
12:    $\hat{\mathbf{y}} \leftarrow \text{NTT}(\mathbf{y})$ 
13:    $\hat{\mathbf{w}} \leftarrow \hat{\mathbf{A}} \circ \hat{\mathbf{y}}$  ▷ Target operation
14:    $\mathbf{w} \leftarrow \text{NTT}^{-1}(\hat{\mathbf{w}})$ 
15:    $\mathbf{w}_1 \leftarrow \text{HIGHTBITS}(\mathbf{w})$ 
16:    $\tilde{c} \leftarrow H(\mu \parallel \text{w1ENCODE}(\mathbf{w}_1), \lambda/4)$ 
17:    $c \leftarrow \text{SAMPLEINBALL}(\tilde{c})$ 
18:    $\hat{c} \leftarrow \text{NTT}(c)$ 
19:    $\hat{\mathbf{x}}_1 \leftarrow \hat{c} \circ \hat{\mathbf{s}}_1$  ▷ Target operation
20:    $\mathbf{x}_1 \leftarrow \text{NTT}^{-1}(\hat{\mathbf{x}}_1)$  ▷ Target operation
21:    $\mathbf{x}_2 \leftarrow \text{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{s}}_2)$ 
22:    $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{x}_1$ 
23:    $\mathbf{r}_0 \leftarrow \text{LOWBITS}(\mathbf{w} - \mathbf{x}_2)$ 
24:   if  $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta \vee \|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$  then
25:      $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$ 
26:   else
27:      $\mathbf{x}_0 \leftarrow \text{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{t}}_0)$ 
28:      $\mathbf{h} \leftarrow \text{MAKEHINT}(-\mathbf{x}_0, \mathbf{w} - \mathbf{x}_2 + \mathbf{x}_0)$ 
29:     if  $\|\mathbf{x}_0\|_\infty \geq \gamma_2 \vee \text{wt}(\mathbf{h}) > \omega$  then
30:        $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$ 
31:     end if
32:   end if
33:    $\kappa \leftarrow \kappa + \ell$ 
34: end while
35:  $\sigma \leftarrow \text{SIGENCODE}(\tilde{c}, \mathbf{z} \bmod^\pm q, \mathbf{h})$ 
36: return  $\sigma$ 

```

product $\hat{\mathbf{w}} = \hat{\mathbf{A}} \circ \hat{\mathbf{y}}$. More explicitly, for every coefficient index ν , each row of the public matrix $\hat{\mathbf{A}}$ contributes one component of the commitment:

$$\hat{\mathbf{w}}_i[\nu] = \sum_{j=0}^{\ell-1} \hat{A}_{i,j}[\nu] \cdot \hat{\mathbf{y}}_j[\nu] \pmod{q}. \quad (6)$$

This expression exposes the repeated operations involving $\hat{\mathbf{y}}_j[\nu]$. For a fixed pair (j, ν) , the scalar $\hat{\mathbf{y}}_j[\nu]$ does not change with the row index i . Therefore, within a single signing execution, the device processes the same secret-dependent scalar once for each public row of $\hat{\mathbf{A}}$, while the public coefficients $\hat{A}_{i,j}[\nu]$ act as known modulation factors. Algorithm 2 explicitly shows this row-wise reuse.

From a side-channel perspective, the loop over the module rank k yields k observations of each ephemeral coefficient within one signing trial. The repeated observations are not identical, because each row multiplies the same $\hat{y}_j$ by a different public coefficient $\hat{A}_{i,j}$. Nevertheless, they are all determined by the same underlying ephemeral coefficient. The joint use of these k row-wise instances increases the amount of exploitable data available from one signing execution. In practice, compiler scheduling, memory effects, and cross-row interference determine how much of this repetition is reflected in the final distinguisher, which motivates the LD-guided weighted fusion strategy in Section 4.

3.3 Binding Ephemeral Leakage to s_1

On its own, the repetition above concerns only the ephemeral vector $\mathbf{y}$ and is insufficient for key recovery. The crucial observation is that an accepted signature publicly links $\mathbf{y}$ to s_1 . Since $\mathbf{z} = \mathbf{y} + cs_1$, we have in the NTT domain that

$$\hat{\mathbf{y}} = \hat{\mathbf{z}} - \hat{c} \circ \hat{s}_1 \pmod{q}. \quad (7)$$

Thus, for each hypothesis on $\hat{s}_{1,j}[v]$, the attacker obtains a corresponding hypothesis on the repeated ephemeral coefficient $\hat{y}_j[v]$.

Integrating Eq. (7) into the coefficient expansion $\hat{\mathbf{w}} = \hat{\mathbf{A}} \circ \hat{\mathbf{y}}$ gives

$$\hat{w}_i[v] = \sum_{j=0}^{\ell-1} \hat{A}_{i,j}[v] \cdot (\hat{z}_j[v] - \hat{c}[v] \cdot \hat{s}_{1,j}[v]) \pmod{q}. \quad (8)$$

All quantities in Eq. (8) are public except $\hat{s}_{1,j}[v]$. Therefore, for a fixed (j, v) , the k row-wise products $\hat{A}_{i,j}[v] \cdot \hat{y}_j[v]$ are k distinct intermediate values that depend on the same unknown $\hat{s}_{1,j}[v]$; this structural property of the module lattice is what our attack exploits.

3.4 Physical Manifestation on Embedded Device

The dependency in Eq. (8) is algorithmic rather than platform-specific. In the NTT domain, multiplication is pointwise by definition; consequently, the decomposition of $\hat{\mathbf{A}} \circ \hat{\mathbf{y}}$ into $\mathbb{Z}_q$ coefficient multiplications holds for any correct ML-DSA implementation, independent of the target platform. What varies across platforms is not whether this decomposition exists, but how much of it is exposed in the side-channel traces. On the single-issue Cortex-M cores considered in this work (e.g., M0, M3 and M4), the $k\ell n$ coefficient multiplications of $\hat{\mathbf{A}} \circ \hat{\mathbf{y}}$ are executed sequentially. Each multiplication $\hat{A}_{i,j}[v] \cdot \hat{y}_j[v]$ can therefore be associated with a localized time interval in the trace, making the contributions of different rows temporally separable. Each coefficient is loaded into

Algorithm 2 Row-wise reuse in $\hat{\mathbf{w}} \leftarrow \hat{\mathbf{A}} \circ \hat{\mathbf{y}}$

Require: Public matrix $\hat{\mathbf{A}} \in T_q^{k \times \ell}$; ephemeral vector $\hat{\mathbf{y}} \in T_q^\ell$

Ensure: Commitment vector $\hat{\mathbf{w}} \in T_q^k$

```

1: for  $i \leftarrow 0$  to  $k-1$  do                                 $\triangleright k$ -times repetition
2:    $\hat{w}_i \leftarrow 0$ 
3:   for  $j \leftarrow 0$  to  $\ell-1$  do
4:      $\text{acc} \leftarrow \hat{A}_{i,j} \circ \hat{y}_j$            $\triangleright$  same  $\hat{y}_j$ , different public row
5:      $\hat{w}_i \leftarrow \hat{w}_i + \text{acc}$ 
6:   end for
7: end for
8: return  $\hat{\mathbf{w}}$ 
```

registers, multiplied by public operands, reduced modulo q , and stored or accumulated in memory. This sequential execution does not affect the black-box security of ML-DSA, but it changes the physical leakage surface: the same algebraic object is executed as a sequence of scalar register and memory transitions.

Throughout this paper, we use a standard non-profiling leakage model in which the instantaneous leakage of an intermediate value V_t is approximated as

$$L(t) = a \cdot w_H(V_t) + b + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2), \quad (9)$$

where $w_H(\cdot)$ denotes the Hamming weight of the intermediate values (which is the special case $v_0 = 0$ of the Hamming-distance model $\text{HD}(v_0, \cdot)$) and a, b are device-dependent constants, and ϵ_t denotes the measurement noise. For memory transitions, the analogous Hamming distance model applies. We do not assume access to a clone device or profiling traces; the attack method in Section 4 uses target traces to rank secret hypotheses.

Micro-architectural optimizations such as lazy reduction enlarge the dynamic range of intermediate accumulators, while $-O3\&-O4$ time scheduling can compress and overlap neighboring coefficient operations. These effects make direct exploitation non-trivial, but they do not remove the row-wise repetition identified above.

3.5 Implication for Masked Implementations

Masking changes the statistical order at which the repeated leakage of the intermediate becomes exploitable, but does not remove the signing dependency that creates row-wise reuse. In a masked implementation, a sensitive intermediate is represented by randomized shares, so direct first-order leakage of the unshared value is suppressed. However, the matrix-vector product $\hat{\mathbf{A}} \circ \hat{\mathbf{y}}$ still processes the shares of the same ephemeral coefficient across the public rows of $\hat{\mathbf{A}}$. Therefore, the structural repetition in Eq. (8) remains present at the level of share-dependent computations.

The public shift $\hat{\mathbf{z}}$ in Eq. (7) also plays a useful role in the masked setting. Absolute-value leakage models can create a sign ambiguity between a value and its additive inverse. As we jointly exploit the leakages, the shifted forms induced by $\hat{\mathbf{z}} - \hat{c} \circ \hat{s}_1$ are generally asymmetric when $\hat{z}_j[v] \neq 0$, which helps break this ambiguity for most coefficients. The remaining corner cases are handled by the algebraic verification step as introduced in Section 4.

Such distinction guided the attack design. Masking prevents direct first-order exploitation of the repeated intermediate, but it does not remove the algebraic repetition itself. Section 4 therefore exploits higher-order leakages from the repeated matrix product with leakage from $\hat{c} \circ \hat{s}_1$, $\text{INTT}(\hat{c} \circ \hat{s}_1)$ and other related operations, to mount CPA, followed by the verification of key coefficients using the native small-norm constraint of ML-DSA.

4 Non-profiling Key Recovery Scheme

This section formalizes the non-profiling recovery pipeline. The attack does not construct templates, train leakage models, or use secret-dependent profiling traces. Instead, it correlates physical measurements from the target device with candidate-dependent intermediate values derived from public signing data. Trace windows corresponding to polynomial arithmetic are localized using macroscopic physical markers, such as SPA-visible operation boundaries.

4.1 Exploitation of Algorithmic Redundancy

The main leakage source exploited in this work is the NTT-domain matrix-vector product $\hat{\mathbf{w}} = \hat{\mathbf{A}} \circ \hat{\mathbf{y}}$. For a module rank k , each coefficient of the ephemeral vector $\hat{\mathbf{y}}$ is reused across the k rows of the public matrix $\hat{\mathbf{A}}$. Thus, for each coefficient index v , the same ephemeral value participates in multiple row-wise products with known public multipliers. This row-wise reuse provides multiple exploitable leakage instances within one signing trace, without requiring additional signatures.

In an unprotected implementation, the sequential evaluation of these products leads to data-dependent register and memory transitions. For a row i , column j , and coefficient index v , we model the relevant transition as:

$$L(v) = a \cdot w_H(\text{MontReduce}(\hat{A}_{i,j}[v] \cdot \hat{y}_j[v])) + b + \epsilon, \quad (10)$$

where again a and b are device-dependent constants, and ϵ denotes measurement noise. All experiments in this paper and the released artifact use this Hamming-weight form. The important property is not a change in the per-sample leakage quality, but the repeated occurrence of related leakage instances for the same coefficient.

4.2 Per-Operation Leakage Models for s_1

It remains to show how a hypothesis on s_1 is turned into a predicted intermediate for each target operation. Recall from Section 3.3 that every accepted signature publicly binds the ephemeral vector to the secret through $\hat{\mathbf{y}} = \hat{\mathbf{z}} - \hat{\mathbf{c}} \circ \hat{s}_1$ (see Eq. (7)), so a candidate $g \in \mathbb{Z}_q$ for the coefficient $\hat{s}_{1,j}[v]$ induces a candidate for the ephemeral coefficient $\hat{y}_j[v]$. We now instantiate this binding per operation.

Matrix-vector product $\hat{\mathbf{A}} \circ \hat{\mathbf{y}}$. Substituting Eq. (7) into the row-wise product gives, for each row i , column j , and coefficient index v , the predicted intermediate

$$V_{Ay}^{(i,j,v)}(g) = \hat{A}_{i,j}[v] \cdot (\hat{z}_j[v] - \hat{c}[v] \cdot g) \bmod q, \quad (11)$$

i.e., k predicted intermediates per coefficient, one per matrix row. The distinguisher scores the Hamming weight of this intermediate against the unprotected traces, and its centered absolute representative $|\cdot|_q$ against the share-combined traces in the masked case. The public offset $\hat{z}_j[v]$ makes the model candidate-dependent for s_1 while still using leakage from the matrix-vector computation. For coefficients with $\hat{z}_j[v] \neq 0$, this offset also helps distinguish the two sign choices that arise in absolute-value leakage models.

Pointwise product $\hat{\mathbf{c}} \circ \hat{s}_1$. This operation computes $\hat{\mathbf{x}}_1 = \hat{\mathbf{c}} \circ \hat{s}_1$ directly on the secret, so the predicted intermediate is simply

$$V_{cs1}^{(j,v)}(g) = \hat{c}[v] \cdot g \bmod q, \quad (12)$$

scored with the same prediction functions as above.

Inverse transform $\text{INTT}(\hat{\mathbf{c}} \circ \hat{s}_1)$. The device then computes $\mathbf{x}_1 = \text{INTT}(\hat{\mathbf{x}}_1)$. Since the INTT is a fixed public linear map, the hypothesis on $\hat{s}_{1,j}[v]$ contributes the known, per-signature-varying term $\hat{c}[v]g$ to every output coefficient through that map, and is scored against each output separately. Because each output coefficient mixes all 256 product coefficients, a single-coordinate hypothesis explains only 1/256 of the output variance, which is consistent with the weak standalone recovery of this operation in Section 5.

4.3 Leading Degree-Guided Score Fusion

The matrix-vector product provides repeated row-wise leakage instances, while the pointwise product $\hat{\mathbf{c}} \circ \hat{s}_1$ and its inverse transform $\text{INTT}(\hat{\mathbf{c}} \circ \hat{s}_1)$ provide more localized candidate scores. We therefore combine these sources at the score level.

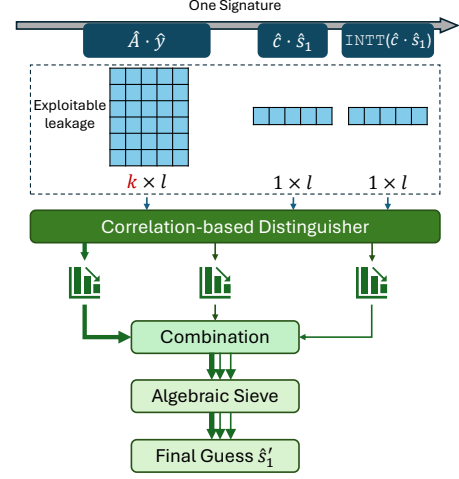


Figure 1: Joint multi-operation fusion analysis.

As illustrated in Figure 1, each evaluated operation produces one candidate-score distribution per coefficient. For an operation o , coefficient index v , and candidate g , let $V_o(g)$ be the predicted intermediate in Section 4.2 and let L_o be the measured leakage on the target operation. The distinguisher is CPA, namely $r_o(g)$ is the Pearson correlation between the intermediate and the measured leakage as follows.

$$r_o(g) = |\rho(w_H(V_o(g)), L_o)|, \quad (13)$$

where the intermediate variable V_o is V_{Ay} or V_{cs1} or other target operations. We map $r_o(g)$ with the Fisher transformation

$$z_o(g) = \frac{1}{2} \ln \frac{1 + r_o(g)}{1 - r_o(g)} \quad (14)$$

used only as a variance-stabilizing reparameterization of this scalar score; the transformed values are not treated as calibrated log-likelihoods². For each coefficient, the fused candidate score is

$$\mathcal{S}_{fused}(g, \alpha) = \sum_o \alpha_o \mathcal{S}_o(g), \quad \alpha_o \geq 0, \quad \sum_o \alpha_o = 1, \quad (15)$$

where, in the primary two-operation case, $\mathcal{S}_{Ay}(g) = z_{Ay}(g)$, $\mathcal{S}_{cs1}(g) = z_{cs1}(g)$, and the simple form is equivalent to weight α via

$$\mathcal{S}_{fused}(g, \alpha) = \alpha \cdot z_{Ay}(g) + (1 - \alpha) \cdot z_{cs1}(g).$$

We search the weight α on a discrete grid $\frac{\alpha}{1-\alpha} = 2^e$ with 25 values of $e \in \{-4, -3.5, \dots, 8\}$, rather than a uniform step.

For each weighting vector α and coefficient index v , let $g_{v,0}$ denote the highest-scoring candidate and let $g_{v,m}$ denote the m -th runner-up. A weighting is preferred when the leading candidate is separated not only from the best competitor but from the whole group of near-maximal competitors, since a large gap to g_1 can

²In the implementation, $r_o(g)$ is clipped strictly below 1 before evaluating Eq. (14) for numerical stability.

coexist with several candidates immediately below it. We therefore score a weighting by a decreasing-weight sum of the first M gaps, which we call the discrete Leading Degree (LD) objective:

$$\mathcal{M}_{LD}^{(M)}(\alpha) = \sum_{m=1}^M \left(\frac{M+1-m}{M+1} \right) \frac{\mathcal{S}_{fused}(g_0, \alpha) - \mathcal{S}_{fused}(g_m, \alpha)}{\sqrt{2}\sigma_S}, \quad (16)$$

where σ_S denotes the standard deviation of $\mathcal{S}_{fused}(\cdot)$ over the full candidate set $\mathbb{Z}_q$. Maximizing $\mathcal{M}_{LD}^{(M)}$ selects weights whose fused score distribution has a separated leading candidate and penalizes cases where several false candidates remain close to the maximum. The linearly decreasing coefficients concentrate the objective on the first gaps, where separation matters most. In our implementation, we set $M = 4$.

4.4 Key Enumeration and Algebraic Sieve

The fusion step produces an ordered candidate list for each NTT coefficient of $\hat{s}_1$. Because some coefficients may still be misranked, we enumerate NTT-domain candidate vectors by the number of coefficients that leave the rank-1 guess. Coefficients are ordered by increasing LD score gap between the top two candidates; at width w , the w coefficients with lowest LD scores are deepened to ranks $\delta_1 \geq \dots \geq \delta_w$ chosen so that $\prod_i \delta_i \leq B_w$, where $\sum_w B_w$ is the total enumeration number. Algorithm 3 shows the steps of our enumeration with the sieve.

Algorithm 3 Algebraic sieve

Require: Score-ordered $\{G_v\}_{v=0}^{n-1}$ after attacks; η ; $\{B_w\}_{w=1}^W$
Ensure: $\hat{s}'_1$ with $\|\text{INTT}(\hat{s}'_1)\|_\infty \leq \eta$, or $\perp$

- 1: $(v_1, \dots, v_n) \leftarrow$ indices sorted by increasing top-1/top-2 gap
- 2: $x_0 \leftarrow$ all-rank-1 vector
- 3: **if** $\|\text{INTT}(x_0)\|_\infty \leq \eta$ **then**
- 4: **return** x_0
- 5: **end if**
- 6: **for** $w \leftarrow 1$ **to** W **do**
- 7: Choose $\delta_1 \geq \dots \geq \delta_w$ with $\prod_i \delta_i \leq B_w$
- 8: **for** $\mathbf{r} \in \prod_{i=1}^w \{0, \dots, \delta_i - 1\} \setminus \{\mathbf{0}\}$ **do**
- 9: $\mathbf{x} \leftarrow x_0$; $\mathbf{x}[v_i] \leftarrow G_{v_i}[r_i]$ for $i = 1, \dots, w$
- 10: **if** $\|\text{INTT}(\mathbf{x})\|_\infty \leq \eta$ **then**
- 11: **return** $\mathbf{x}$
- 12: **end if**
- 13: **end for**
- 14: **end for**
- 15: **return** $\perp$

As depicted in Figure 2, synthesized NTT-domain candidate vectors $\hat{s}'_1$ produced by the enumeration are checked in the coefficient domain. The sieve evaluates whether the inverse transform satisfies the ML-DSA small-norm constraint:

$$\|\text{INTT}(\hat{s}'_1)\|_\infty \leq \eta. \quad (17)$$

A wrong NTT-domain vector is unlikely to satisfy the coefficient-domain bound after the inverse transform. Assuming a malformed candidate behaves like a random vector over $\mathbb{Z}_q$, the probability

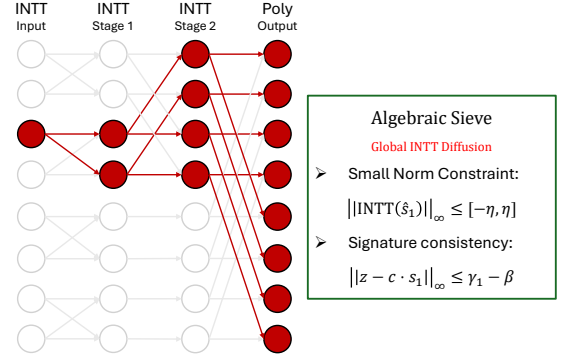


Figure 2: Dual-domain algebraic sieve.

that all coefficients satisfy the bound is estimated by

$$P_{false} \approx \left(\frac{2\eta + 1}{q} \right)^{256}. \quad (18)$$

The sieve therefore serves as a lightweight filter that discards the vast majority of incorrect candidate vectors. It does not replace the statistical distinguisher; it only removes candidates that already violate the coefficient-domain norm bound until the correct vector appears within the enumerated frontier. Final acceptance of a surviving candidate is decided by the signing verification equations of ML-DSA.

Applicability to masked implementations. The same pipeline applies after the univariate leakage is replaced by the combined leakage according to the masking order. For the first-order masking ($d = 2$), if $L_{(x)_0}$ and $L_{(x)_1}$ denote the two share-dependent observations, the second-order centered product is

$$L_{CP} = (L_{(x)_0} - \mathbb{E}(L_{(x)_0})) \cdot (L_{(x)_1} - \mathbb{E}(L_{(x)_1})). \quad (19)$$

For d shares, the attack uses a corresponding degree- d combination in place of Eq. (19). The predicted intermediates, LD-guided score fusion, and INTT sieve remain unchanged; only the combined leakage fed to the correlation distinguisher is replaced.

Higher-order share combination raises the noise levels of the exploitable leakage, but it does not remove the k row-wise observations of each ephemeral coefficient. Across the simulated noise levels, such multiplicity continues to reduce the traces needed for full recovery relative to the attack on $\hat{c} \circ \hat{s}_1$, while MTD increases smoothly with the noise variance.

Public acceptance of a candidate. The algebraic sieve is only a filter: every sufficiently short vector can pass. An adversary who already holds the public key and accepted signatures can decide correctness without the secret, e.g. by checking $\|z - cs'_1\|_\infty \leq \gamma_1$ on acquired signatures and/or the public-key bound $\|\text{As}'_1 - t_1 \cdot 2^{13}\|_\infty \leq 2^{12} + \eta$ once a complete s'_1 is assembled (cf. the forgery proxy in Appendix B.4). Our experimental MTD counts a recovery when the enumerated polynomial matches the target; those public checks are the operational acceptance criterion and are not required as an online oracle inside the enumerator.

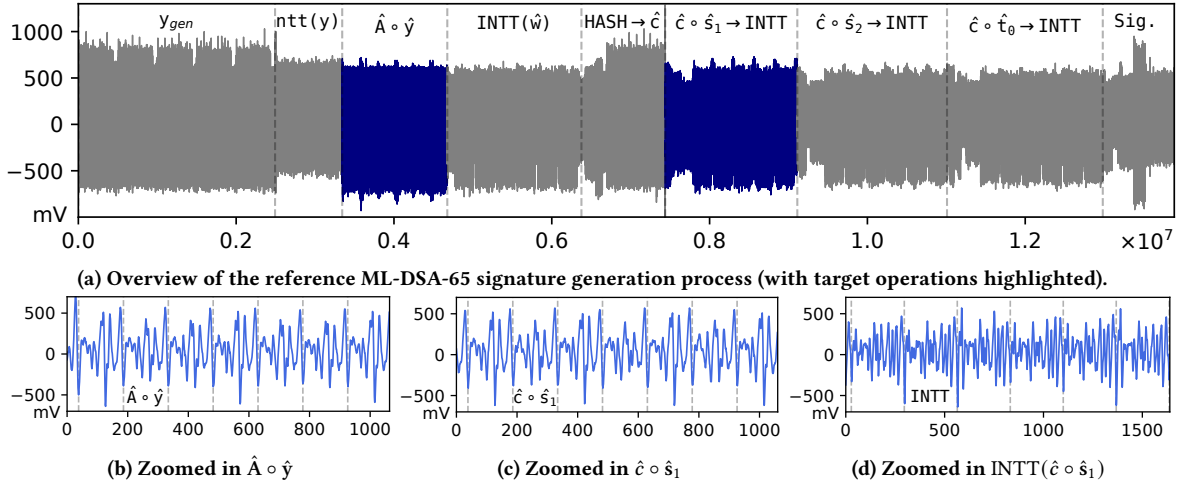


Figure 3: Power traces of unprotected ML-DSA-65 on STM32F405 with precomputation removed for clarity.

5 Experimental Evaluation

Before presenting the experimental setup, we highlight one trend that guides the interpretation of the results. The standardized cryptographic security level of an ML-DSA parameter set does not directly determine the trace complexity of this attack. In our setting, the relevant factor is the module rank k , which determines how many row-wise matrix-vector leakage instances are available within one signing execution.

This distinction is visible across the three FIPS 204 parameter sets. Although ML-DSA-87 targets the highest cryptographic security level, it also has the largest module rank, with $k = 8$. The matrix-vector product therefore provides more row-wise leakage instances than in ML-DSA-65 and ML-DSA-44. As the results below confirm, the trace complexity of our attack therefore decreases as the module rank increases: this makes ML-DSA-87 require the fewest traces under our row-wise fusion attack, while ML-DSA-44 requires the most among the evaluated parameter sets.

5.1 Experimental Setup

Evaluations target three parameter sets (ML-DSA-44, 65, and 87) executing on a 32-bit STM32F405RGT6 ARM Cortex-M4 microcontroller clocked at default 168 MHz. We target two implementations: the unprotected reference implementation of the pq-crystals project³, and the masked implementation of Coron et al. [12]⁴. The latter is a widely used open-source reference for masked ML-DSA and therefore serves as the primary protected target. More recent schemes such as SUCRE [3] add shuffling to the rejection-sampling phase but leave the upstream polynomial arithmetic targeted in this work unchanged⁵.

Due to the 192 KiB SRAM constraint of the STM32F405 platform, we isolated and executed the target operations (INTT transformations, matrix-vector product, and pointwise polynomial arithmetic) to capture power traces correspondingly. Such isolation accommodates the memory overhead of the masking gadgets, and follows

the measurement practice of prior evaluations on the same platform [12, 13, 19] for cycle counting and physical acquisition. The memory footprint of the matrix-vector product ($\hat{A} \circ \hat{y}$) scales multiplicatively with the number of shares and, for ML-DSA-87, exceeds the available SRAM at masking order two. The protected evaluation is therefore restricted to the first-order masked implementation in this work.

The firmware is compiled with the ARM Keil toolchain under two configurations: `-O0` to isolate instruction-level leakage, and `-O3` with `-Otime` to replicate a production environment characterized by pipeline interleaving. Unless stated otherwise, the reported results correspond to the `-O3&-Otime` configuration. Power consumption is measured as the voltage drop across a shunt resistor in the supply line of the target. To optimize the SNR, the measured signal passes a Mini-Circuits BLP-5+ low-pass filter, a PA303 pre-amplifier, and a secondary BLP-5+ filter prior to digitization by a Teledyne LeCroy WaveRunner 9104 oscilloscope. Datasets were captured at 25 MS/s and 100 MS/s. We acquired 1,000 traces per security level for the unprotected implementation. For the masked implementation, the centered-product combination of the shares lowers the effective SNR, and we therefore acquired 5,000 traces per parameter set.

Figure 3 shows a complete power trace of the unprotected ML-DSA-65 signing execution. The targeted operations—the matrix product $\hat{A} \circ \hat{y}$, the pointwise product $\hat{c} \circ \hat{s}_1$, and the inverse transform $INTT(\hat{c} \circ \hat{s}_1)$, remain clearly identifiable despite the compiler optimizations, permitting precise windowing for the attack.

5.2 Attack Objective and Evaluation Metrics

Prior to experimental results, it is necessary to define the criteria for a successful side-channel attack against ML-DSA. The ultimate goal of the adversary is universal signature forgery. As established in prior cryptanalytic literature [9, 36], forging a valid ML-DSA signature does not require recovery of the entire secret key structure; recovering the secret vector s_1 is sufficient (cf. the forgery discussion in Appendix B.4).

The secret vector s_1 consists of ℓ discrete polynomials, each containing 256 coefficients. During signing, these ℓ polynomials are processed sequentially through identical algebraic operations.

³<https://github.com/pq-crystals/dilithium>

⁴https://github.com/fragerar/tches24_masked_Dilithium

⁵To the best of our knowledge, the implementation of the SUCRE masking scheme is not publicly available at the time of writing.

Consequently, the physical leakage produced by each polynomial is structurally identical, meaning equal difficulty for recovering each polynomial. The trace complexity of recovering one polynomial is therefore representative of the recovery of the whole vector.

Accordingly, a successful recovery is defined as the exact recovery of all 256 coefficients of target polynomial s_1 . The recovery is counted only when it is stable, i.e., it does not fall below 256 coefficients at any larger trace count tested. Achieving this Minimum Traces to Disclosure (MTD) demonstrates the practical capability to sequentially recover the full s_1 vector, thereby enabling signature forgery. Unless stated otherwise, each configuration is evaluated over ten independent instances, and the reported MTD is the average across these instances.

5.3 On Simulated Leakage

To separate algorithmic scaling from board-specific acquisition effects, we first evaluate the attack under a known leakage model before turning to the physical measurements on STM32 devices. Following the Hamming-weight simulation setting of Tosun et al. [38], each sensitive 32-bit intermediate (each share, in the masked case) x leaks as $L = \text{HW}(x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Unprotected intermediates are attacked by CPA under this Hamming-weight model. For first-order masking ($d = 2$), the two share traces are combined by the centered product of Eq. (19), and the hypotheses are scored against the optimal prediction function (OPF) of Tosun et al. [38], concretely the centered absolute value $|x|$ of the shared intermediate. We fix the parameter set to ML-DSA-87, the largest module rank ($k = 8$), i.e., the most favorable case for our row-wise attack, and test different noise levels with $\sigma \in \{0, 2, 4, 8\}$ for both the unprotected and the first-order masked setting. For uniformly distributed 32-bit intermediates, $\text{Var}(\text{HW}(x)) = 8$, so these noise levels correspond to $\text{SNR} = 8/\sigma^2$. ML-DSA-44 and ML-DSA-65 are covered by the physical experiments below, where results of all three parameter sets are evaluated on the STM32F405 board.

We report the results as shown in Table 3: two single operations; unweighted Fisher-z fusion ($\alpha = 1/2$); LD-guided fusion; and LD fusion followed by the algebraic sieve (enumeration budget 10^6 ; at most 10 coefficients may leave the rank-1 guess). Table 3 gives *average* MTD: the average over 50 independent seeds of the smallest trace count that recovers all 256 coefficients exactly and remains exact thereafter.

Table 3: Simulated average MTD of ML-DSA-87 versus different noise levels σ .

d	Case	$\sigma = 0$	$\sigma = 2$	$\sigma = 4$	$\sigma = 8$
1	$\hat{c} \circ \hat{s}_1$	10.40	28.75	51.20	138.50
	$\hat{A} \circ \hat{y}$	2.00	4.00	7.43	21.33
	Unweighted fusion	2.00	11.90	20.53	47.48
	LD-based fusion	2.00	4.00	7.43	21.27
	Fusion + sieve	2.00	3.50	6.80	19.13
2	$\hat{c} \circ \hat{s}_1$	272.25	358.10	610.40	2750.20
	$\hat{A} \circ \hat{y}$	36.20	47.40	86.20	395.50
	Unweighted fusion	75.20	94.11	167.20	785.50
	LD-based fusion	35.56	46.96	84.70	375.80
	Fusion + sieve	31.88	41.36	73.90	325.00

As the noise level σ increases, average MTD grows monotonically for all cases, while the qualitative ordering is preserved: $\hat{c} \circ \hat{s}_1$ remains far from full recovery; $\hat{A} \circ \hat{y}$ dominates; LD fusion closely matches $\hat{A} \circ \hat{y}$ and possess slight advantages; with the sieve further reducing the average MTD. The effect of our fusion approach is visible in both settings: in the unprotected case, LD-guided fusion consistently outperforms unweighted averaging for $\sigma \geq 2$ (e.g., 7.43 versus 20.53 traces at $\sigma = 4$ with $d = 1$); in the first-order masked case, unweighted averaging requires nearly two times the MTD for full key recovery, compared with LD-guided fusion. First-order masking raises the trace cost at each σ . The subsequent subsections confirm the same structural picture on a real-world device (STM32F405) for all three FIPS 204 levels.

5.4 On Unprotected Implementation

The unprotected implementation is evaluated to establish the baseline exploitation of algorithmic redundancy. Unless otherwise specified, the convergence curves report the average MTD across 10 independent trials. Consequently, the number of correctly recovered coefficients may exhibit fractional values prior to achieving complete convergence. Figure 4 illustrates the empirical convergence trajectories of the primary leakage sources across all FIPS 204 parameter sets.

For ML-DSA-87, the matrix-vector product $\hat{A} \circ \hat{y}$ dominates the low-trace recovery process. This aligns with the structural argument established in Section 3: for each coefficient index v , the identical ephemeral value $\hat{y}_j[v]$ is processed across the $k = 8$ rows of the public matrix. By yielding multiple leakage observations parameterized by known public rows, the matrix product becomes substantially more informative than the isolated pointwise product $\hat{c} \circ \hat{s}_1$. Because the matrix product establishes high baseline distinguishability independently, the relative margin of improvement provided by the fusion framework is constrained in such a high-redundancy setting.

Table 4: Observed MTD against the unprotected ML-DSA implementations on STM32F405 under -03&-0time at 100 MS/s.

Metric	ML-DSA-44	ML-DSA-65	ML-DSA-87
Rank (k, ℓ)	(4, 4)	(6, 5)	(8, 7)
MTD (LD Fusion Only)	7	5	4
MTD (LD Fusion + Sieve)	6	4	4
Traces Saved	1	1	0

However, Table 4 and Figure 4 demonstrate that this dynamic shifts at lower security levels. As the module rank (k, ℓ) decreases to (4, 4) in ML-DSA-44, the structural redundancy of the matrix-vector product diminishes, yielding fewer repeated leakage instances per signature. This reduction lowers the aggregate leakage volume and decreases the relative dominance of $\hat{A} \circ \hat{y}$. Consequently, the standalone matrix-vector curve reaches the full-recovery boundary later. The fused curve benefits from the additional candidate scores provided by $\hat{c} \circ \hat{s}_1$ and $\text{INTT}(\hat{c} \circ \hat{s}_1)$, which is most visible for ML-DSA-44. As observed in Figure 4(c), the fused trajectory visibly diverges from the isolated $\hat{A} \circ \hat{y}$ curve, confirming that the fusion mechanism effectively compensates for the loss of matrix redundancy by aggregating supplementary leakage.

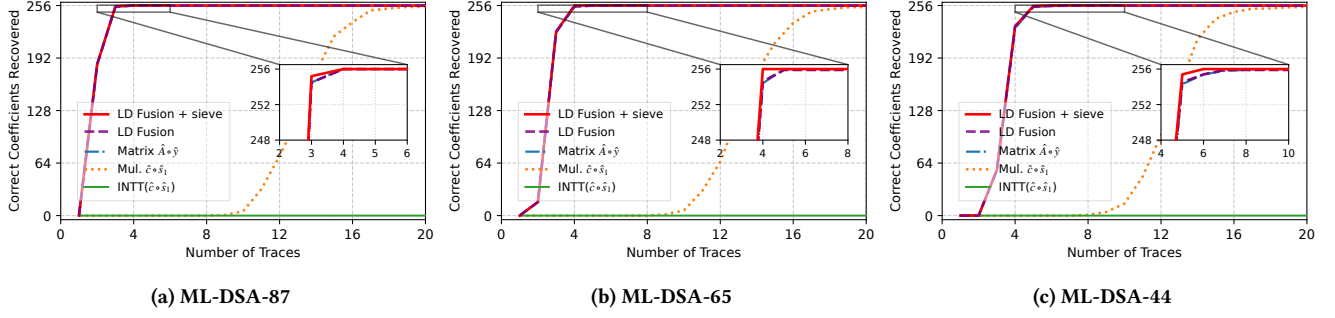


Figure 4: Key recovery against unprotected implementation on the STM32F405 target board under the -03&-0time setting.

Furthermore, the application of the dual-domain algebraic sieve resolves residual statistical ambiguity. As highlighted in the magnified insets of Figure 4, the sieve truncates the convergence tail, attaining the 256-coefficient recovery threshold earlier than the only statistical top-1 candidates. Finally, as detailed in Table 4, while the theoretical cryptographic strength scales upwards from ML-DSA-44 to -87, our physical evaluation reveals an inverse trend in trace complexity: ML-DSA-87 requires the fewest traces under our row-wise fusion attack. This inversion is a direct consequence of the structural vulnerability exposed in this work. Because higher security levels dictate larger module dimensions, they necessitate increased algorithmic repetition of the ephemeral secret during the $\hat{A} \circ \hat{y}$ computation, thereby amplifying the exploitable leakage.

5.5 On Masked Implementation

We first evaluated our approach against the first-order masked implementation of Coron et al. [12]. Unless otherwise stated, the experiments target STM32F405 under -03&-0time compilation, acquired at 100 MS/s. We report average MTD over 10 independent trials. For reference, on the STM32F405 board with the measurement setups, the maximum PCCs with different operations are 0.428 for $\hat{c} \circ \hat{s}_1$, 0.280 for $\hat{A} \circ \hat{y}$, and 0.113 for $\text{INTT}(\hat{c} \circ \hat{s}_1)$, respectively.

Figure 5 separates the recovery behavior of each operation. The matrix-vector product $\hat{A} \circ \hat{y}$ reaches the full-key boundary earlier than the pointwise multiplication $\hat{c} \circ \hat{s}_1$ and $\text{INTT}(\hat{c} \circ \hat{s}_1)$ across all parameter sets. The inverse-transform curve recovers only a small number of coefficients within the plotted trace window.

The LD fusion curve uses the candidate-score distributions from the three operations. Its effect is most visible near the final recovery boundary, where the fused ordering reaches 256 coefficients before the isolated matrix-vector curve in the lower-rank parameter sets.

The ordering across parameter sets follows the module rank. For ML-DSA-87, where $k = 8$, the matrix-vector curve is close to the fused curves. For ML-DSA-65 and ML-DSA-44, where $k = 6$ and $k = 4$, the matrix-vector curve reaches full recovery at larger trace counts. This is consistent with the number of row-wise instances available in $\hat{A} \circ \hat{y}$.

The INTT sieve verifies complete NTT-domain candidates produced by the LD-ranked enumerator. The enumerator alters coefficients in increasing order of the top-two score gap and stops when a complete candidate satisfies $\|\text{INTT}(\hat{s}'_1)\|_\infty \leq \eta$.

Table 5 reports the trace count and enumeration depth at the first verified recovery. For all three parameter sets, the verified candidate

Table 5: MTD and enumeration depth against first-order masked ML-DSA on STM32F405 (MTD reported in traces).

Metric	ML-DSA-87	ML-DSA-65	ML-DSA-44
$\hat{A} \circ \hat{y}$ Only	114	144	216
Fusion MTD	106	132	154
Fusion + Sieve MTD	90	114	136
Worst True Rank	6	7	5
Checked Vectors	402,368	497,804	305,740
Enumeration Time	927 ms	1210 ms	752 ms

appears within enumeration budget sum: 305,740 checked vectors for ML-DSA-44, 497,804 for ML-DSA-65, and 402,368 for ML-DSA-87. The verification time take about 1 second in all cases.

Increasing the enumeration sum beyond the reported frontier did not change the recovered key in our experiments.

5.6 Robustness Analysis

We use two robustness tests to separate compiler and acquisition effects from the main masked results. First, the -00 unprotected setting serves as a control for easier instruction-level alignment. As shown in Table 6, the sieve changes the MTD by one trace for ML-DSA-44 and does not change the MTD for ML-DSA-65. This confirms that the sieve has a limited visible effect when the statistical ranking already reaches full recovery within a small trace count. We therefore focus on -03&-0time, where polynomial operations are compressed by compiler optimization (see Appendix A).

Second, we evaluate the acquisition rate on protected ML-DSA-65. Under the same -03&-0time binary, reducing the sampling rate from 100 MS/s to 25 MS/s increases the MTD from 114 to 182 traces. The attack still reaches full recovery, but the lower sampling rate provides fewer samples over the same arithmetic window and increases the number of signatures required by the fused distinguisher.

Table 6: Robustness analysis of compiler optimization level and sampling rate.

Setting	Parameter Set	Rate	Compiler	MTD
Unprotected sieve ablation	ML-DSA-65	100 MS/s	-00	4 $\rightarrow$ 4
Unprotected sieve ablation	ML-DSA-44	100 MS/s	-00	6 $\rightarrow$ 5
Protected sampling-rate (+ sieve)	ML-DSA-65	100 MS/s	-03&-0time	114
Protected sampling-rate (+ sieve)	ML-DSA-65	25 MS/s	-03&-0time	182

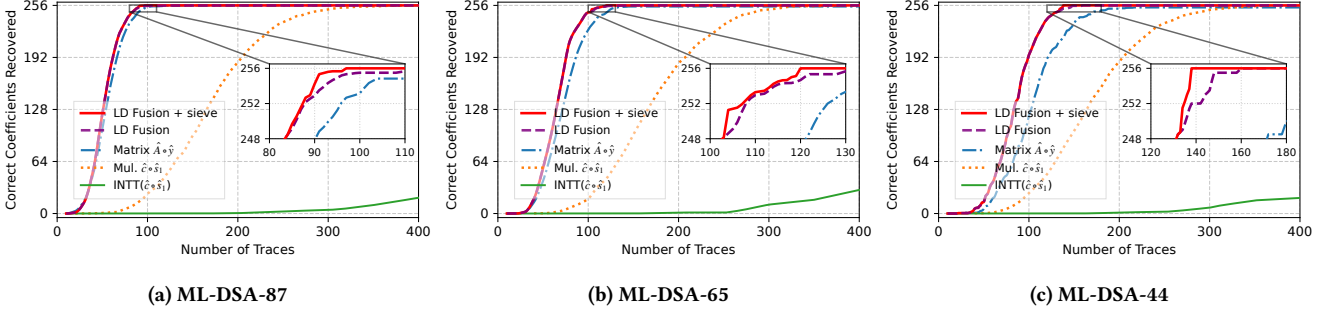


Figure 5: Key recovery against first-order masked implementation on STM32F405 under the -O3&-Otime setting.

5.7 Cross-Platform Validation

We also evaluate the first-order masked ML-DSA-87 and ML-DSA-44 on an STM32F303 board in the ChipWhisperer setup. This experiment uses the same scheme and serves as a platform check against the STM32F405 results. The fused attack reaches complete recovery at 82 and 144 traces for ML-DSA-87 and ML-DSA-44, and the INTT sieve reduces the MTD to 74 and 124 traces. The corresponding maximum PCCs on STM32F303 are 0.431 for $\hat{c} \circ \hat{s}_1$, 0.293 for $\hat{A} \circ \hat{y}$, and 0.150 for $\text{INTT}(\hat{c} \circ \hat{s}_1)$, respectively.

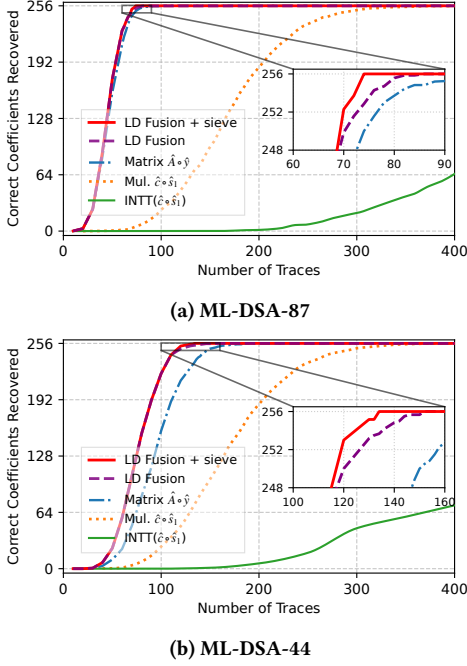


Figure 6: Key recovery against first-order masked implementation on STM32F303.

Figure 6 shows the same pattern as on the STM32F405 device: with k halved from 8 to 4, the fused curve lies closer to the $\hat{A} \circ \hat{y}$ curve. The platform therefore provides a useful cross-check: the matrix-vector product remains the strongest individual source, and LD-guided fusion preserves the final ordering needed by the sieve. The result also confirms that our approach is not specific to one single Cortex-M4 board.

6 Discussion

6.1 Computational Efficiency

To quantify the operational overhead of the attack approach, execution times were measured using a workstation equipped with an Intel Xeon 8336C processor (32 cores, 64 threads) and dual NVIDIA RTX 3090 GPUs. We extract per coefficient multiplication with 400 or 500 samples (-O3 or -O0) at a 100 MS/s sampling rate. The correlation power analysis (CPA) typically requires Point-of-Interest (PoI) selection or dimensionality reduction to remain computationally tractable. In contrast, our parallelized exploitation approach processes the raw temporal samples directly, eliminating the need for data preprocessing. Under these baseline conditions, the complete key recovery executes in under 10 seconds for the 10-trace unprotected dataset. Against the first-order masked ML-DSA-87 implementation, our attack recovers the secret key from 90 traces in approximately 42 seconds.

Because our approach operates in a strictly non-profiling setting, these execution times represent the complete operational cost of the attack. This distinction is necessary when comparing performance with profiling schemes. Profiling approaches frequently report rapid key-extraction times during the template matching phase; however, this omits the prior overhead required to construct the initial leakage models. Building these templates typically necessitates the acquisition and processing of tens of thousands of characterization traces (e.g., 20,000 to 50,000) on a clone device. By eliminating the need for time-consuming template building, our proposed approach achieves competitive execution time while reducing the data and time complexities to practical levels.

6.2 Higher-Order Masking

Our experiments evaluate the first-order masked implementation of Coron et al. [12] on STM32F405 for all three FIPS 204 parameter sets (Section 5.5). Going beyond first order is limited less by the absence of row-wise reuse than by platform cost and noise: each additional share multiplies the memory footprint of $\hat{A} \circ \hat{y}$, and for ML-DSA-87 second-order masking already exceeds the 192 KiB SRAM under our isolation setup in Section 5.

Higher-order masking changes the statistical extraction step: an intermediate protected with d shares requires a degree- d combination of the observations, which increases the number of samples needed for reliable candidate ranking. The row-wise reuse in $\hat{A} \circ \hat{y}$ remains present at the algorithmic level, and LD-guided fusion with the algebraic sieve remains applicable once candidate lists

are formed. What changes in practice is the cost of forming those lists, which depends on the leakage model, share placement, and implementation schedule. We leave a complete evaluation of implementations masked at second order and beyond to future work.

6.3 Hardware Implementations

Our evaluation targets software on sequential microcontrollers, where the k row-wise products of one ephemeral coefficient occupy distinct time intervals. Hardware implementations of ML-DSA challenge this premise in two ways. First, parallel multipliers and pipelined NTT/pointwise units collapse several coefficient operations into the same clock cycles, so the per-operation leakage superposes rather than separates in time; the measured power consumption is then the joint switching activity of the parallel units, which lowers the per-operation SNR and complicates leakage interval locating. Second, a k -wide multiplier array may evaluate all rows concurrently, reshaping the temporal repetition that our fusion exploits into a single aggregated leakage; whether the aggregate switching still carries enough structure for score fusion must be assessed on a concrete implementation. Hardware-software co-designs for Kyber/Dilithium on RISC-V SoC FPGAs [42] illustrate how strongly the microarchitecture reshapes the schedule of the polynomial arithmetic. Assessing the transfer of our attack to such targets therefore requires extra effort and is left to future work.

Recent CPA on an industry-grade hardware ML-DSA root-of-trust implementation [21] already shows that non-profiling key recovery is practical on FPGA platforms. To be specific, that attack targets the pointwise product $\hat{c} \circ \hat{s}_1$ (via modular reduction after NTT-domain multiplication). Our contribution is the structural use of row-wise multiplicity in $\hat{A} \circ \hat{y}$, which should be reassessed when parallel NTT and pointwise units might collapse that multiplicity into concurrent switching.

6.4 On Other Module-Lattice Schemes

The structural leakage we exploit comes from the module-lattice matrix-vector product: one secret-dependent vector is multiplied by every row of a public matrix, so each coefficient is processed k times in a single execution. In ML-DSA that vector is the ephemeral $\hat{y}$ in $\hat{w} = \hat{A} \circ \hat{y}$, with module rank $k \in \{4, 6, 8\}$.

The same structure appears elsewhere in the module-lattice family. In ML-KEM (CRYSTALS-Kyber) [5], key generation forms $\hat{t} = \hat{A} \circ \hat{s} + \hat{e}$, reusing each coefficient of the static secret $\hat{s}$ once per matrix row; encapsulation forms $\mathbf{u} = \text{INTT}(\hat{A}^\top \circ \hat{y}) + \mathbf{e}_1$, reusing each coefficient of the ephemeral $\hat{y}$ across the k rows of $\hat{A}^\top$ ($k \in \{2, 3, 4\}$). Decapsulation multiplies the long-term secret into a public ciphertext polynomial, which is a pointwise NTT product of the same type as $\hat{c} \circ \hat{s}_1$ in ML-DSA.

Saber [14] keeps a module-LWR matrix-vector product over a fully splitting NTT ring, again with row-wise reuse of a secret or ephemeral vector. FrodoKEM [6] omits NTT, but still multiplies large matrices in which a secret matrix or vector is shared across many rows and columns. In each case, the same secret-dependent coefficients are repeatedly processed by the matrix-vector operations within one execution.

Table 7: Average MTDs of the algebraic sieve and the lattice reduction method using flatter under the simulation scenario on ML-DSA-87.

d	Method	$\sigma = 0$	$\sigma = 2$	$\sigma = 4$	$\sigma = 8$
1	LD-based fusion	2.00	4.00	7.43	21.27
	Fusion + sieve	2.00	3.50	6.80	19.13
	Fusion + lattice	2.00	3.00	5.07	14.07
2	LD-based fusion	35.56	46.96	84.70	375.80
	Fusion + sieve	31.88	41.36	73.90	325.00
	Fusion + lattice	21.63	28.56	49.30	216.00

6.5 Lattice Reduction vs. Algebraic Sieving

Several prior attacks recover the remaining secret coefficients by lattice reduction or related solvers after a partial side-channel recovery [33, 38]. The short-norm sieve in this work is a different post-processing step. From the ranked lists by LD fusion, it enumerates possible NTT-domain vectors over a small set of low-confidence coefficients (width at most $W = 10$, total budget $\sum_w B_w \approx 10^6$), keeps those with $\|\text{INTT}(\hat{s}'_1)\|_\infty \leq \eta$, and accepts a survivor only under the verification equations of ML-DSA (Section 4.4). It does not run BKZ and does not solve separate SIS or LWE instances.

The sieve still needs good side-channel output in the sense that as many coefficients as possible are correctly recovered. Indeed, if many correct coefficients sit far down their top rankings, a width- W enumeration cannot reach the correct coefficient vector—just as lattice reduction fails when too few coordinates are recovered. Both steps assume that CPA has already placed most correct candidates near the top of the lists.

For comparison, on the same ranked lists from the LD fusion, we also fix coefficients with a clear score margin and recover the free coefficients using Babai’s nearest-plane algorithm [1] after basis reduction, trying free-set sizes in $\{48, 64, 96, 128, 160\}$. Table 7 gives the average MTD for LD fusion, fusion with the short-norm sieve, and fusion with this lattice reduction under the simulated scenario of Section 5.3. We use the sieve on the device measurements (roughly one second at the budgets of Table 5). The lattice reduction can further lower the average MTD when LD scores are stable, at about one minute per solver on the same computing platform. In both cases, the limit is how well the non-profiling attack ranks the correct coefficients. Still, it is worth mentioning that our INTT-based sieve is designed to solve the last-mile problem after an efficient CPA and provide an alternative to the exhaustive post-processing methods. It offers distinct value in the side-channel setting, despite being less effective than the lattice reduction-based approach.

6.6 Implications for Countermeasures

Our results suggest that countermeasures for ML-DSA should account for arithmetic before the rejection checks. Schemes that protect only rejection sampling, hint generation, or final validity checks do not necessarily remove leakage from the NTT-domain matrix-vector product.

A direct mitigation is to randomize the execution order of the row-wise computation, in the spirit of the shuffle-based countermeasures that protect the rejection-sampling phase, e.g., SUCRE [3].

Concretely, if the order in which the k row products of the same ephemeral coefficient are executed is permuted independently in every signing execution, the repetitions of the same intermediate no longer align across traces, and the gain of horizontal averaging degrades accordingly. Note that shuffling raises the trace complexity of the attack but does not remove the per-trace leakage itself; it is therefore a complement to, not a substitute for, masking. Operationally, such shuffling can be realized by permuting the matrix rows, interleaving independent work, or injecting dummy row operations, at the cost of additional tables to keep the result correctly ordered. Such techniques must be evaluated together with masking, since randomization can also introduce overhead and may interact with memory layout. The main design requirement is that repeated processing of the same ephemeral coefficient should not remain consistently aligned across traces.

7 Conclusion

This work exposes the structural vulnerability introduced by the core algebraic structures of the post-quantum signature scheme ML-DSA from a side-channel perspective. We demonstrated that the necessity of performing matrix-vector operations in the NTT domain creates a systematic side-channel signal that persists despite compiler optimizations and typical masking countermeasures. Our evaluation reveals a critical paradox: the physical side-channel resilience of ML-DSA inversely correlates with its theoretical security. ML-DSA-87 is shown to be the most vulnerable because its larger module dimensions inherently force greater algorithmic repetition of the ephemeral variable, inadvertently amplifying horizontal leakage.

To exploit such a vulnerability, we introduced a leading-degree (LD) guided dynamic fusion engine that integrates temporal leakage across distinct mathematical operations, combining candidate-score distributions from distinct arithmetic operations. By pairing this statistical fusion with a short-norm algebraic sieve, we established a candidate-verification method that avoids lattice-reduction post-processing in our recovery pipeline.

Our empirical evaluation on the STM32F405 microcontroller confirmed the efficacy of this approach, extracting the full secret key from a masked implementation of ML-DSA-87 with 90 traces. Ultimately, these findings demonstrate that the currently proposed countermeasures should also account for repeated leakage in upstream arithmetic operations. Securing post-quantum implementation for physical deployment requires a countermeasure design that also covers upstream arithmetic that actively disrupts structural repetition throughout the entire algorithmic computation path.

8 Open Science

The code and datasets of this paper will be available on Zenodo under the reserved DOI <https://doi.org/10.5281/zenodo.22538276>.

Acknowledgments

The authors sincerely thank the reviewers for their constructive and insightful comments, which have substantially strengthened the contribution of this work.

This work is supported in part by the National Natural Science Foundation of China (No.U2336205, No.62502207, No.62302226,

No.62302224), Yunnan Provincial Major Science and Technology Special Plan Projects (202502AD080008) and Yunnan Provincial New R&D Institution Cultivation Project (202404BQ040148).

References

- [1] László Babai. 1986. On Lovász' Lattice Reduction and the Nearest Lattice Point Problem. *Combinatorica* 6, 1 (1986), 1–13. doi:10.1007/BF02579403
- [2] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. 2018. Masking the GLP Lattice-Based Signature Scheme at Any Order. In *EUROCRYPT 2018, Part II (LNCS, Vol. 10821)*. 354–384. doi:10.1007/978-3-319-78375-8_12
- [3] Sonia Belaïd, Ryad Benadjila, Julien Devevey, Morgane Guerreau, Thomas Legavre, Ange Martinelli, Thomas Ricosset, Matthieu Rivain, and Mélissa Rossi. 2026. ML-DSA masking sweetened with SUCRE: Shuffle-and-Unmask Countermeasure for REjection sampling. *TCHES* 2026 (2026), 618–659. doi:10.46586/TCHES.V2026.I1.618-659
- [4] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. 2020. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptogr. Eng.* 10, 2 (2020), 163–188. doi:10.1007/S13389-019-00220-8
- [5] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2018. CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. In *EuroS&P, IEEE*, 353–367. doi:10.1109/EUROSP.2018.00032
- [6] Joppe W. Bos, Maximilian Ofner, Joost Renes, Tobias Schneider, and Christine van Vredendaal. 2021. The Matrix Reloaded: Multiplication Strategies in FrodoKEM. In *CANS*. Springer, 72–91.
- [7] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation Power Analysis with a Leakage Model. In *CHES 2004, Cambridge, MA, USA, August 11-13, 2004. Proceedings (LNCS, Vol. 3156)*, Marc Joye and Jean-Jacques Quisquater (Eds.). Springer, 16–29. doi:10.1007/978-3-540-28632-5_2
- [8] Olivier Bronchain, Melissa Azouaoui, Mohamed ElGhamrawy, Joost Renes, and Tobias Schneider. 2024. Exploiting Small-Norm Polynomial Multiplication with Physical Attacks Application to CRYSTALS-Dilithium. *TCHES* 2024, 2 (2024), 359–383. doi:10.46586/TCHES.V2024.I2.359-383
- [9] Leon Groot Bruinderink and Peter Pessl. 2018. Differential Fault Attacks on Deterministic Lattice Signatures. *TCHES* 2018, 3 (2018), 21–43. doi:10.13154/tches.v2018.i3.21-43
- [10] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. 2002. Template Attacks. In *CHES 2002, Redwood Shores, CA, USA, August 13-15, 2002 (LNCS, Vol. 2523)*, Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar (Eds.). Springer, 13–28. doi:10.1007/3-540-36400-5_3
- [11] Zhaohui Chen, Emre Karabulut, Aydin Aysu, Yuan Ma, and Jiwu Jing. 2021. An Efficient Non-Profiled Side-Channel Attack on the CRYSTALS-Dilithium Post-Quantum Signature. In *39th IEEE ICCD 2021, Storrs, CT, USA, October 24-27, 2021*. 583–590. doi:10.1109/ICCD53106.2021.00094
- [12] Jean-Sébastien Coron, François Gérard, Tancrede Lepoint, Matthias Trannoy, and Rina Zeitoun. 2024. Improved High-Order Masked Generation of Masking Vector and Rejection Sampling in Dilithium. *TCHES* 2024, 4 (2024), 335–354. doi:10.46586/TCHES.V2024.I4.335-354
- [13] Simon Damm, Asja Fischer, Alexander May, Soundes Marzougui, Leander Schwarz, Henning Seidler, Jean-Pierre Seifert, Jonas Thietke, and Vincent Quentin Ulitzsch. 2025. Solving Concealed ILWE and Its Application for Breaking Masked Dilithium. In *ASIACRYPT 2025, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part II (LNCS)*, Goichiro Hanaoka and Bo-Yin Yang (Eds.). Springer, 608–640. doi:10.1007/978-981-95-5096-8_19
- [14] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. 2018. Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM. In *AFRICACRYPT (LNCS, Vol. 10831)*. Springer, 282–305. doi:10.1007/978-3-319-89339-6_16
- [15] Rafaël del Pino, Thomas Prest, Mélissa Rossi, and Markku-Juhani O. Saarinen. 2023. High-Order Masking of Lattice Signatures in Quasilinear Time. In *S&P 2023*. IEEE, 1168–1185. doi:10.1109/SP46215.2023.10179342
- [16] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2018. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *TCHES* 2018, 1 (2018), 238–268. doi:10.13154/TCHES.V2018.I1.238-268
- [17] Haopeng Fan, Hailong Zhang, Yongjuan Wang, Wenhao Wang, Haojin Zhang, and Qingjun Yuan. 2025. Multivariate Template Attack Against NTT-Based Polynomial Multiplication of Dilithium. *IEEE Trans. Inf. Forensics Secur.* 20 (2025), 8570–8582. doi:10.1109/TIFS.2025.3598577
- [18] Karine Gandolfi, Christophe Mourtél, and Francis Olivier. 2001. Electromagnetic Analysis: Concrete Results. In *CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings (LNCS)*, Çetin Kaya Koç, David Naccache, and Christof Paar (Eds.). Springer, 251–261. doi:10.1007/3-540-44709-1_21
- [19] Julius Hermelink, Kai-Chun Ning, and Richard Petri. 2025. Finding and Protecting the Weakest Link - On Side-Channel Attacks on y in Masked ML-DSA. In *CRYPTO*

- 2025, Santa Barbara, CA, USA, August 17-21, 2025, *Proceedings, Part V (LNCS, Vol. 16004)*, Yael Tauman Kalai and Seny F. Kamara (Eds.). Springer, 3–37. doi:10.1007/978-3-032-01901-1_1
- [20] Julius Hermelink, Silvan Streit, Erik Mårtensson, and Richard Petri. 2025. A Generic Framework for Side-Channel Attacks Against LWE-Based Cryptosystems. In *EUROCRYPT 2025, Madrid, Spain, May 4-8, 2025, Proceedings, Part VIII (LNCS, Vol. 15608)*, Serge Fehr and Pierre-Alain Fouque (Eds.). Springer, 3–32. doi:10.1007/978-3-031-91101-9_1
- [21] Merve Karabulut and Reza Azarderakhsh. 2025. Efficient Cpa Attack on Hardware Implementation of ML-DSA in Post-Quantum Root of Trust. In *IEEE HOST 2025, San Jose, CA, USA, May 5-8, 2025*. IEEE, 392–402. doi:10.1109/HOST64725.2025.11050056
- [22] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. 2018. A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model. In *EUROCRYPT 2018, Part III (LNCS, Vol. 10822)*, 552–586. doi:10.1007/978-3-319-78372-7_18
- [23] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *CRYPTO '99, Santa Barbara, California, USA, August 15-19, 1999, Proceedings (LNCS, Vol. 1666)*, Michael J. Wiener (Ed.). Springer, 388–397. doi:10.1007/3-540-48405-1_25
- [24] Adeline Langlois and Damien Stehlé. 2015. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.* 75, 3 (2015), 565–599. doi:10.1007/S10623-014-9938-4
- [25] Yong Liu, Yuejun Liu, Yongbin Zhou, Yiwen Gao, Zehua Qiao, and Huaxin Wang. 2024. A Novel Power Analysis Attack against CRYSTALS-Dilithium Implementation. In *IEEE European Test Symposium, ETS 2024, The Hague, Netherlands, May 20-24, 2024*. IEEE, 1–6. doi:10.1109/ETS61313.2024.10567325
- [26] Yuejun Liu, Yongbin Zhou, Shuo Sun, Tianyu Wang, Rui Zhang, and Jingdian Ming. 2021. On the Security of Lattice-Based Fiat-Shamir Signatures in the Presence of Randomness Leakage. *IEEE Trans. Inf. Forensics Secur.* 16 (2021), 1868–1879. doi:10.1109/TIFS.2020.3045904
- [27] Zheng Liu, An Wang, Congming Wei, Yaoling Ding, Jingqi Zhang, Annyu Liu, and Liehuang Zhu. 2025. Release the Power of Rejected Signatures: An Efficient Side-Channel Attack on the ML-DSA Cryptosystem. *IEEE Trans. Inf. Forensics Secur.* 20 (2025), 13356–13369. doi:10.1109/TIFS.2025.3643784
- [28] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2007. *Power analysis attacks - revealing the secrets of smart cards*. Springer.
- [29] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. 2019. Masking Dilithium - Efficient Implementation and Side-Channel Evaluation. In *Applied Cryptography and Network Security (LNCS)*. Springer, 344–362. doi:10.1007/978-3-030-21568-2_17
- [30] National Institute of Standards and Technology (NIST). 2024. *FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA)*. Standard. U.S. Department of Commerce. doi:10.6028/NIST.FIPS.204 Federal Information Processing Standards Publication 204.
- [31] Robert Primas, Peter Pessl, and Stefan Mangard. 2017. Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption. In *CHES 2017 (LNCS, Vol. 10529)*, 513–533. doi:10.1007/978-3-319-66787-4_25
- [32] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Jingdian Ming, Chengbin Jin, and Huizhong Li. 2023. Practical Public Template Attacks on CRYSTALS-Dilithium With Randomness Leakages. *IEEE Trans. Inf. Forensics Secur.* 18 (2023), 1–14. doi:10.1109/TIFS.2022.3215913
- [33] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Mingyao Shao, and Shuo Sun. 2023. When NTT Meets SIS: Efficient Side-channel Attacks on Dilithium and Kyber. *IACR Cryptol. ePrint Arch.* (2023), 1866. <https://eprint.iacr.org/2023/1866>
- [34] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Yuhua Zhao, and Shuyi Chen. 2024. Single Trace is All It Takes: Efficient Side-channel Attack on Dilithium. *IACR Cryptol. ePrint Arch.* 2024 (2024), 512. <https://eprint.iacr.org/2024/512>
- [35] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Yuhua Zhao, Hao Yuan, and Dixiao Du. 2025. Efficient CNN-Based Side-Channel Attacks on Dilithium without Device Access. In *ISCAS 2025*. IEEE, 1–5. doi:10.1109/ISCAS56072.2025.11043271
- [36] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. 2018. Side-channel Assisted Existential Forgery Attack on Dilithium - A NIST PQC candidate. *IACR Cryptol. ePrint Arch.* 2018 (2018), 821. <https://eprint.iacr.org/2018/821>
- [37] Yiteng Sun, Tianfu Zhang, Zhuo Huang, Yu Yu, Yan Zhuang, Shuo Sun, and Weijia Wang. 2025. Finding More Hints - Improved Power Analysis Attacks on Dilithium. *IEEE Trans. Inf. Forensics Secur.* 20 (2025), 11963–11974. doi:10.1109/TIFS.2025.3618387
- [38] Tolun Tosun, Elisabeth Oswald, and Erkan Savas. 2025. Non-Profiled Higher-Order Side-Channel Attacks against Lattice-Based Post-Quantum Cryptography. *IACR Commun. Cryptol.* 2, 3 (2025), 31. doi:10.62056/AOTXL8N4E
- [39] Tolun Tosun and Erkan Savas. 2024. 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 (2024), 3353–3365. doi:10.1109/TIFS.2024.3359890
- [40] Vincent Quentin Ulitzsch, Soundes Marzougui, Mehdi Tibouchi, and Jean-Pierre Seifert. 2022. Profiling Side-Channel Attacks on Dilithium - A Small Bit-Fiddling

Leak Breaks It All. In *SAC 2022, Windsor, ON, Canada, August 24-26, 2022, Revised Selected Papers*, 3–32. doi:10.1007/978-3-031-58411-4_1

- [41] Ruize Wang, Kalle Ngo, Joel Gärtner, and Elena Dubrova. 2024. Unpacking Needs Protection: A Single-Trace Secret Key Recovery Attack on Dilithium. *IACR Commun. Cryptol.* 1, 3 (2024), 12. doi:10.62056/AOFH89N4E
- [42] Tengfei Wang, Chi Zhang, Xiaolin Zhang, Dawu Gu, and Pei Cao. 2024. Optimized Hardware-Software Co-Design for Kyber and Dilithium on RISC-V SoC FPGA. *TCHES* 2024, 3 (2024), 99–135. doi:10.46586/TCHES.V2024.I3.99-135
- [43] Lichao Wu, Guilherme Perin, and Stjepan Picek. 2022. The Best of Two Worlds: Deep Learning-assisted Template Attack. *TCHES* 2022, 3 (2022), 413–437. doi:10.46586/TCHES.V2022.I3.413-437
- [44] Yuhua Zhao, Wei Cheng, Zehua Qiao, Yuejun Liu, and Yongbin Zhou. 2026. Rejection Matters: Efficient Non-Profiling Side-Channel Attack on ML-DSA via Exploiting Public Templates. In *DATE 2026, Verona, Italy, April 20-22, 2026*. IEEE, 1–7. doi:10.23919/DATE69613.2026.11539074
- [45] Yuanyuan Zhou, Weijia Wang, Yiteng Sun, and Yu Yu. 2025. Rejected Signatures' Challenges Pose New Challenges: Key Recovery of CRYSTALS-Dilithium via Side-Channel Attacks. *TCHES* 2025, 4 (2025), 817–847. doi:10.46586/TCHES.V2025.I4.817-847
- [46] Junfan Zhu and Jiqiang Lu. 2025. Leading Degree: A Metric for Model Performance Evaluation and Hyperparameter Tuning in Deep Learning-Based Side-Channel Analysis. *TCHES* 2025, 2 (2025), 333–361. doi:10.46586/TCHES.V2025.I2.333-361

A Impact of Compiler Optimization

Figure 7 compares the leakage profile of the unprotected reference implementation compiled with `-O0` and `-O3` time. The `-O0` trace exhibits regularly spaced correlation peaks, reflecting the sequential processing of polynomial coefficients. In contrast, the optimized trace is temporally compressed by instruction scheduling and pipeline interleaving, causing adjacent coefficient leakages to overlap and reducing the effectiveness of alignment.

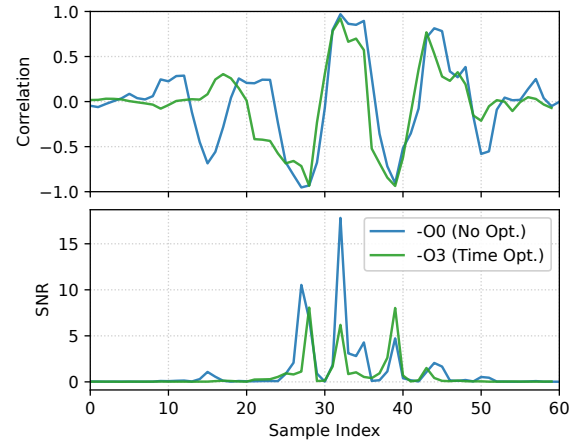


Figure 7: Impact of compiler optimization on leakage from $\hat{c}_0 \circ \hat{s}_1$ in the unprotected reference implementation.

After normalizing the execution time of the optimized trace to the `-O0` baseline, the macroscopic operation-level leakage profile remains recognizable. This supports our focus on full-speed `-O3`–`-Otime` evaluation: the optimized setting is harder to align at instruction granularity, but the operation-level leakage needed by the fusion attack remains exploitable.

B Empirical Validation of the Algebraic Sieve

To rigorously evaluate the reliability and operational feasibility of the proposed Dual-Domain Algebraic Sieve, we conducted a

large-scale stress test focusing on two critical performance metrics: the empirical False Positive Rate (FPR) and the computational throughput of the algebraic sieve.

B.1 Experimental Approach

We implemented a highly optimized evaluation framework utilizing OpenMP for parallelization and AVX2 intrinsics for vectorized polynomial arithmetic. The experiment was executed on a 64-core workstation to simulate a resource-rich adversary. The testing approach adhered to a strict *negative control protocol* designed to measure the probability of non-secret vectors inadvertently satisfying the cryptographic norm constraints.

- **Null Hypothesis (H_0)** A random polynomial vector $\hat{\mathbf{x}} \in \mathbb{Z}_q^n$, uniformly distributed in the NTT domain, satisfies the infinity norm bound $\|\text{INTT}(\hat{\mathbf{x}})\|_\infty \leq \eta$ with non-negligible probability.
- **Perturbation Injection** To simulate realistic candidate enumeration, we generated N trial vectors by injecting random noise terms $\delta \in \mathbb{Z}_q$ into random coefficients of a valid key vector. This replicates the output of a fusion attack where most coefficients are correct, but a few contain errors.
- **Scale** The stress test was scaled logarithmically up to $N = 10^{13}$ (10 trillion) independent trials to ensure statistical significance beyond standard cryptographic margins.

B.2 Statistical Robustness Analysis

We monitored the sieve for Type I errors (false positives). The results, summarized in Table 8, demonstrate an empirical FPR of exactly zero across all 10^{13} trials.

Table 8: Empirical Validation of Sieve Robustness.

Trials (N)	Throughput (Mops/s)	False Positives	Observed FPR
10^8	38.77	0	0
10^9	39.83	0	0
10^{10}	39.91	0	0
10^{11}	40.84	0	0
10^{12}	40.76	0	0
10^{13}	41.15	0	0

This empirical result is consistent with the random-candidate estimate in Eq. (18). We conclude that the Algebraic Sieve acts as a short-norm filter with negligible false-positive rate. Any candidate passing the sieve is, with high probability, the correct secret key $\mathbf{s}_1$.

B.3 Computational Throughput Evaluation

The AVX2-optimized implementation stabilizes at a throughput of approximately 41.15 million operations per second (Mops/s). The execution time scales approximately linearly ($O(N)$), evaluating 10^{13} trials in approximately 67.5 hours. The throughput remains stable over the tested range. This throughput translates to a per-candidate verification latency of ≈ 24.3 ns. For a tail-end enumeration scenario checking the top 10^6 combinations, the wall-clock time is approximately 25 milliseconds, introducing negligible latency to the extraction pipeline. Figures 8a and 8b show execution time and throughput, respectively.

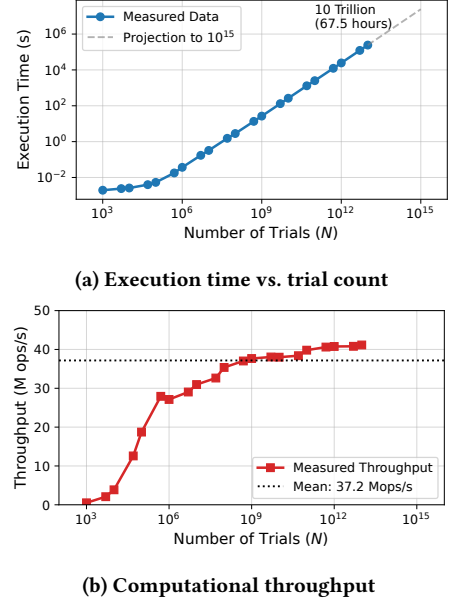


Figure 8: Empirical complexity analysis of the dual-domain algebraic sieve

B.4 Cryptographic Impact and Universal Forgery

The recovery of $\mathbf{s}_1$ is sufficient for universal forgery. Following the forgery construction established by prior work [9], the adversary derives a proxy vector $\mathbf{u} = \mathbf{A}\mathbf{s}_1 - \mathbf{t}_1 \cdot 2^{13}$ to bypass the rejection checks. This yields two post-processing steps:

- (1) **Filtering:** The Algebraic Sieve ($\|\text{INTT}(\hat{\mathbf{s}}'_1)\|_\infty \leq \eta$) filters candidate vectors that violate the short-norm bound.
- (2) **Acceptance:** For the validated candidate surviving the sieve, the adversary executes the forgery algorithm locally using the target's public key.

Consequently, the sieve provides a bounded post-processing step without requiring lattice reduction. Even under trace-constrained or noisy conditions where statistical ambiguity pushes a few difficult-to-recover coefficients down to a local candidate rank of $R \approx 1000$, the bounded search depth of the high-throughput sieve efficiently resolves such localized uncertainty. This combination guarantees the successful identification and verification of the secret key without requiring further interaction with the target device.