

Too Small to Hide: Single-Trace Key Recovery from ML-KEM Key Generation

Vahid Jahandideh

v.jahandideh@cs.ru.nl

Radboud University

Nijmegen, The Netherlands

Abstract

Key generation in ML-KEM (CRYSTALS-Kyber) samples a short secret from a centered binomial distribution (CBD) and immediately transforms it with the number-theoretic transform (NTT). Each execution draws fresh randomness, so an attacker obtains a single trace and cannot average. We show that one power trace of the optimized pqm4 implementation on an Arm Cortex-M4 suffices, and that the two operations leak complementary information. The CBD sampler stores each coefficient as a signed 16-bit two's-complement word, which reveals its sign almost without error but too little about its magnitude to yield long error-free hint sets. The missing magnitude leaks in the subsequent NTT, where coefficients are multiplied by public twiddle factors. Used as hints in a primal lattice attack, the combined leakage reduces the estimated BKZ block size for ML-KEM-768 from 624 to 129, corresponding to $2^{182.2}$ and $2^{37.7}$ in the core-SVP model. A control experiment shows that the same NTT code leaks the Hamming weight of a near-uniform operand as strongly as that of the secret, yet reveals almost nothing about its value. What makes a coefficient vulnerable is how few values it can take, not how small it is. That points to a countermeasure inside the CBD sampler. Each sample v is stored as a random representative $\tilde{v} = v + rq$ of its residue class, with r nonzero, so the NTT and all later code run unchanged. This q -randomization drives the sign leakage to the noise floor, reduces the NTT magnitude leakage by 4.5×, and raises the block size from 129 back to 611—about four bits of estimated security lost instead of more than 140. The sampler grows by 937 bytes of firmware.

CCS Concepts

• Security and privacy → Hardware attacks and countermeasures.

Keywords

ML-KEM/Kyber, single-trace side-channel attack, centered binomial sampling, number-theoretic transform, lattice attacks with hints.

ACM Reference Format:

Vahid Jahandideh. 2026. Too Small to Hide: Single-Trace Key Recovery from ML-KEM Key Generation. In *1st Workshop on Assisted Cryptographic Technologies for Implementation Verification (ACTIVE '26)*, November 15–19, 2026, The Hague, Netherlands. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3847162.3849237>



This work is licensed under a Creative Commons Attribution 4.0 International License. *ACTIVE '26, The Hague, Netherlands*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-3023-8/2026/11

<https://doi.org/10.1145/3847162.3849237>

1 Introduction

CRYSTALS-Kyber [8], standardized by NIST as the Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) [20], bases its security on the hardness of Module Learning With Errors (MLWE). During key generation, ML-KEM samples a secret vector $\mathbf{s}$ and an error vector $\mathbf{e}$ with small coefficients and publishes

$$\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q},$$

where $\mathbf{A}$ is public. Without additional information, recovering $\mathbf{s}$ from $(\mathbf{A}, \mathbf{b})$ requires solving the underlying MLWE problem.

Physical implementations, however, may reveal partial information about $\mathbf{s}$ and $\mathbf{e}$. When combined with the public MLWE instance, this information, commonly called *hints* [11, 14, 15], can substantially reduce the cost of key recovery.

Obtaining hints from key generation poses a particular challenge: each execution produces a new secret, so measurements from independent executions cannot normally be combined against the same secret. An exception arises when the randomness comes from a stored seed, since regenerating the key reproduces the same secret and allows averaging over repeated traces. We address the harder setting, in which key generation runs with fresh randomness, and study what a single execution reveals. Our target is the optimized m4fspeed ML-KEM implementation of pqm4 [18] on an Arm Cortex-M4, and we assume a profiled attacker who first characterizes the device with known keys.

Previous work by Peng and Kuhn [21] and Fouque et al. [12] identified the centered binomial distribution (CBD) sampler as a single-trace target. The CBD sampler produces coefficients from a very small set and stores them as signed 16-bit integers, a representation that strongly exposes the sign: for example, $-1 = 0\text{x}\text{FFFF}$ while $+1 = 0\text{x}\text{0001}$.

Our measurements show that this leakage is highly asymmetric. The CBD window reveals the signs of the secret coefficients almost perfectly, but provides much less information about their magnitudes. Values such as $+1$ and $+2$ have different binary representations but the same Hamming weight, making them difficult to distinguish through the dominant store leakage. In our setup, the CBD window supplies reliable sign information about $\mathbf{s}$ but too few hints to make key recovery inexpensive on its own (Section 4).

The missing information appears immediately afterwards in the forward number-theoretic transform (NTT). During the NTT, secret-dependent operands are multiplied by public twiddle factors. These multiplications turn small differences between input coefficients into substantially different intermediate values, exposing the magnitude information that the CBD store did not reveal. The leakage is strongest when a coefficient is multiplied before being mixed with others, and weakens in later butterfly layers.

We construct a profiled attack that follows each coefficient to its most informative point in the NTT trace. The NTT alone recovers most of the information needed for the lattice attack; the signs obtained from the CBD window provide a further refinement. We estimate the attack cost through the required block size β of BKZ lattice reduction, where smaller block sizes indicate lower estimated cost (Sections 3 and 5).

A control experiment explains why the NTT leakage is exploitable at all. Pessl and Primas [22] observed that short inputs enable a single-trace NTT attack. That observation admits two readings: either short operands leak differently, or ordinary leakage becomes informative because the input carries little uncertainty. Running the same NTT code on the five-valued secret and on the near-uniform coefficients of $As + e$ separates the two: both leak their Hamming weight comparably, yet the leakage identifies the value only for the secret. The decisive property is thus the *entropy* of the operand rather than its size.

This reading also suggests the form a countermeasure should take. Masking protects binomial sampling [28] and the complete ML-KEM computation [13] with formal guarantees, but at a cost in cycles and code size that constrained devices cannot always afford. If low entropy is the root cause, that entropy can instead be raised directly. We store each secret coefficient v as a randomized representative

$$\tilde{v} = v + rq,$$

where r is drawn from a small symmetric set excluding zero, independently for each coefficient. Because $\tilde{v} \equiv v \pmod{q}$, the NTT and the modular operations that follow return the same result modulo q with no unmasking step, while the value entering the NTT has higher entropy and no longer exposes the sign of v . The change is confined to the sampler. It is an empirical leakage-reduction technique rather than a masking scheme with a security proof, but it largely removes the advantage obtained by the attack studied here (Section 6).

1.1 Our contributions

A single-trace attack combining two leakage sources. The CBD sampler and the NTT leak complementary properties of the same secret, and one power trace of the optimized pqm4 implementation reaches both: the sampler fixes the sign of each coefficient, the NTT resolves its magnitude. Used together as hints for a primal lattice attack on ML-KEM-768, they supply 521 of the 768 secret coordinates exactly and lower the estimated BKZ block size from 624 to 129, or $2^{182.2}$ to $2^{37.7}$ in the core-SVP model (Sections 4–5).

An explanation of why the NTT leaks. Short NTT inputs are known to make single-trace attacks more effective [22], but the responsible property has not been isolated, since small operands are also low-entropy ones. We separate the two with a matched control that runs identical NTT code on the five-valued secret and on near-uniform inputs. Hamming-weight leakage is comparable in both cases; only for the low-entropy operand does that leakage identify the value. Entropy, not size, is what the attack exploits, and that is what tells us where a countermeasure belongs (Section 5.3).

A lightweight countermeasure. Storing each secret coefficient as a random representative $v + rq$ raises the entropy of the value the NTT multiplies and leaves every later step unchanged modulo q ,

with no unmasking. The change is confined to the sampler, though it depends on the order in which the compiler emits the operations. The countermeasure drives the sign leakage to the noise floor and reduces the NTT magnitude leakage by 4.5×. For the primal attack, it raises the block size from 129 to 611—about four bits of estimated security lost instead of more than 140—for 937 bytes of firmware (Section 6).

1.2 Related work

Key-generation and CBD leakage. Peng and Kuhn [21] study adaptive template attacks against the centered binomial sampler, while Fouque et al. [12] combine supervised classification of sampler leakage with lattice reduction to recover an ML-KEM key from a single trace. Their acquisition settings provide substantially more samples per target clock cycle than our four-samples-per-clock setup, allowing finer temporal resolution. In our measurements the CBD leakage alone does not suffice for an efficient attack; we therefore combine the complementary leakage of two consecutive operations.

Single-trace attacks on the NTT. Primas et al. [24] introduced a single-trace attack that combines local leakage from an inverse NTT through belief propagation over its butterfly graph. Pessl and Primas [22] subsequently targeted the forward NTT of the short ephemeral polynomial r during Kyber encryption on an optimized Cortex-M4 implementation. They state that the attack applies to the short inputs. Our control experiment isolates the mechanism behind that observation: the operand entropy rather than the operand size.

The key-generation forward NTT. Chinbat et al. [9, 10] directly attack the forward NTT of s during Kyber key generation, using deep-learning classifiers on the early butterflies of an FPGA NTT accelerator. Ravi et al. [27] examine the same transform from a fault-analysis perspective. Our experiments instead target the m4fspeed ML-KEM implementation in pqm4, whose NTT is written in Arm assembly and uses packed 16-bit coefficients, improved Plantard multiplication, fused layers, and lazy reduction [1, 18, 23]. We combine the CBD and NTT observations as exact and approximate hints [11, 14, 15].

Countermeasures and scope. Masking can protect both binomial sampling [28] and the complete ML-KEM computation [13], while configurable masking and shuffling have been studied specifically for the NTT [26]. These constructions come with security arguments, but also with an overhead that is not always affordable on constrained targets. The randomized representation evaluated here is a lighter, implementation-specific leakage-reduction technique and does not provide the formal guarantees of masking; Zijlstra et al. [30] previously used a similar technique to protect RLWE decryption on FPGAs. Finally, our target is the NTT transform itself; attacks on point-wise or pair-pointwise multiplication after the secret has entered the NTT domain address a different operation [7, 19, 25, 29].

2 Preliminaries

Ring and parameters. ML-KEM fixes the prime modulus $q = 3329$ and ring dimension $n = 256$, and performs its polynomial arithmetic

in

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1).$$

The relation $X^n = -1$ makes polynomial multiplication negacyclic. Each element of $\mathbb{Z}_q$ has a unique centered representative in $\{-(q-1)/2, \dots, (q-1)/2\}$. More generally, any integer $\tilde{v} \equiv v \pmod{q}$ is a representative of v . In the implementation studied here, polynomial coefficients are stored as signed 16-bit integers.

ML-KEM evaluates polynomial products through the NTT:

$$fg = \text{NTT}^{-1}(\text{BaseMul}(\text{NTT}(f), \text{NTT}(g))).$$

Since q admits 256th but not 512th roots of unity, $X^n + 1$ splits into 128 quadratic factors over $\mathbb{Z}_q$. Accordingly, BaseMul multiplies corresponding pairs of coefficients, viewed as polynomials of degree at most one, modulo these quadratic factors [20, Algorithms 11 and 12].

Notation. A polynomial is written in sans-serif lowercase (f), a vector of polynomials in bold lowercase ($\mathbf{f}$), and a matrix in bold uppercase ($\mathbf{F}$). For $\mathbf{f} \in \mathcal{R}_q^k$, where k denotes the module rank, the i th component polynomial is written f_i . We identify $\mathbf{f}$ with the length- kn coefficient vector obtained by concatenating its component polynomials, so that

$$f_i(X) = \sum_{j=0}^{n-1} f_{in+j} X^j, \quad 0 \leq i < k.$$

Hence f_j , for $0 \leq j < kn$, denotes the j th scalar coordinate of the concatenated vector, whereas f_i continues to denote a component polynomial. For $\mathbf{u}, \mathbf{v} \in \mathcal{R}_q^k$, we write

$$\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=0}^{k-1} u_i v_i \in \mathcal{R}_q.$$

Implementation identifiers are written in typewriter font: for example, s_0 denotes a polynomial mathematically, whereas $s[0]$ denotes its source-code representation.

The centered binomial distribution. We write $x \stackrel{\$}{\leftarrow} \chi$ for a draw from a distribution χ , and let $\mathcal{U}_m$ denote the uniform distribution on $\{0, \dots, m-1\}$; the same notation is used for uniform sampling over rings and matrices, as in $\mathcal{U}_q(\mathcal{R}_q^{k \times k})$. The centered binomial distribution χ_η is sampled as

$$x = \sum_{i=1}^{\eta} (a_i - b_i), \quad a_i, b_i \stackrel{\$}{\leftarrow} \mathcal{U}_2,$$

where the a_i and b_i are independent uniform bits. The distribution has support $\{-\eta, \dots, \eta\}$, mean zero, and variance $\sigma_\chi^2 = \eta/2$. Thus, χ_2 has five possible outputs and χ_3 has seven. When discussing recovery of a sampled coefficient, its sign refers to the sampled integer before reduction modulo q , with zero counted as nonnegative.

The notation $\mathbf{f} \stackrel{\$}{\leftarrow} \chi_\eta(\mathcal{R}_q)$ means that the coefficients of $\mathbf{f}$ are sampled independently from χ_η and interpreted modulo q . The notation extends componentwise to polynomial vectors.

Algorithm 1 High-level in-place forward NTT.

```

1: function NTT(a)
2:    $\kappa \leftarrow 1$ 
3:   for len  $\leftarrow 128, 64, \dots, 2$  do
4:     for start  $\leftarrow 0, 2\text{len}, \dots, 256 - 2\text{len}$  do
5:        $\zeta \leftarrow \zeta_\kappa; \kappa \leftarrow \kappa + 1$ 
6:       for j  $\leftarrow \text{start}, \dots, \text{start} + \text{len} - 1$  do
7:          $t \leftarrow \text{PLANTARDMUL}(\zeta, a[j + \text{len}])$ 
8:          $a[j + \text{len}] \leftarrow a[j] - t$ 
9:          $a[j] \leftarrow a[j] + t$ 
10:  return a

```

2.1 Optimized NTT with Plantard reduction

Transform structure. For $q = 3329$, let $\omega = 17$, which has order 256 in $\mathbb{Z}_q$. Since $512 \nmid (q-1)$, $X^{256} + 1$ does not split into linear factors over $\mathbb{Z}_q$. Instead,

$$X^{256} + 1 = \prod_{i=0}^{127} (X^2 - \omega^{2i+1}).$$

The forward NTT therefore stops after seven Cooley–Tukey layers and represents a polynomial by 128 coefficient pairs. It is evaluated in place, from standard coefficient order to bit-reversed order.

A Cooley–Tukey butterfly has the form

$$t = \zeta v \bmod q, \quad (u, v) \mapsto (u + t, u - t),$$

where ζ is a public power of ω . The high-level transform is given in Algorithm 1 [18]. Here, $\zeta_1, \dots, \zeta_{127}$ are the standard bit-reversed twiddle factors, and $\text{PLANTARDMUL}(\zeta, x)$ returns a centered representative of $\zeta x \bmod q$.

Improved Plantard multiplication. Let $\text{hi}_{16}(x)$ denote the signed upper half-word of the low 32 bits of x . The improved Plantard construction uses

$$\ell = 16, \quad \alpha = 3, \quad q' = q^{-1} \bmod 2^{32} = 1806234369.$$

These parameters satisfy the required condition $q < 2^{\ell-\alpha-1} = 2^{12}$. For signed operands in the supported range, Plantard multiplication [16] is

$$\text{Plant}(a, b) = \text{hi}_{16}(q (\text{hi}_{16}(abq') + 2^\alpha)).$$

It satisfies

$$\text{Plant}(a, b) \equiv -ab(2^{32})^{-1} \pmod{q}, \quad -\frac{q}{2} < \text{Plant}(a, b) < \frac{q}{2}.$$

For NTT multiplication, the fixed scaling factor is absorbed into the precomputed twiddle constants. Thus, $\text{PLANTARDMUL}(\zeta, v)$ returns a value t satisfying

$$t \equiv \zeta v \pmod{q}, \quad -\frac{q}{2} < t < \frac{q}{2}.$$

Lazy reduction. The twiddle product t is reduced to a centered representative, but the butterfly outputs $u + t$ and $u - t$ are not immediately reduced. Hence, if B_r bounds the coefficient magnitudes before layer r , then

$$B_{r+1} \leq B_r + \frac{q}{2}.$$

Correctness therefore requires the intermediate representatives to remain within the signed 16-bit range. Starting from CBD coefficients with $B_0 \leq \eta$, seven layers give $B_7 \leq \eta + 7q/2 < 2^{14}$, so all intermediates fit comfortably in the signed 16-bit range. Section 6 uses the remaining headroom in this budget.

Table 1: ML-KEM parameter sets. All variants use $n = 256$ and $q = 3329$.

Parameter set	k	η_1	η_2	d_u	d_v
ML-KEM-512	2	3	2	10	4
ML-KEM-768	3	2	2	10	4
ML-KEM-1024	4	2	2	11	5

Algorithm 2 Public-key encryption underlying ML-KEM.

```

1: function Key-Gen( $k$ )
2:    $\mathbf{A} \xleftarrow{\$} \mathcal{U}_q(\mathcal{R}_q^{k \times k})$ 
3:    $\mathbf{s} \xleftarrow{\$} \chi_{\eta_1}(\mathcal{R}_q^k), \mathbf{e} \xleftarrow{\$} \chi_{\eta_1}(\mathcal{R}_q^k)$ 
4:    $\mathbf{b} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}$ 
5:   return  $pk := (\mathbf{A}, \mathbf{b}), sk := \mathbf{s}$ 

6: function Encrypt( $pk, m, seed$ )
7:    $(\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2) \xleftarrow{seed} (\chi_{\eta_1}(\mathcal{R}_q^k), \chi_{\eta_2}(\mathcal{R}_q^k), \chi_{\eta_2}(\mathcal{R}_q^k))$ 
   ▷ seed drives a deterministic PRNG
8:    $\mathbf{u} \leftarrow \mathbf{A}^T \mathbf{r} + \mathbf{e}_1$ 
9:    $\mathbf{v} \leftarrow \langle \mathbf{b}, \mathbf{r} \rangle + \mathbf{e}_2 + \lceil m \rceil_{2 \rightarrow q}$  ▷  $0 \mapsto 0, 1 \mapsto \lfloor q/2 \rfloor$ 
10:   $\mathbf{c}_1 \leftarrow \lceil \mathbf{u} \rceil_{q \rightarrow D_u}, \mathbf{c}_2 \leftarrow \lceil \mathbf{v} \rceil_{q \rightarrow D_v}$ 
11:  return  $c := (\mathbf{c}_1, \mathbf{c}_2)$ 

12: function Decrypt( $\mathbf{c}_1, \mathbf{c}_2, sk$ )
13:   $\mathbf{u}' \leftarrow \lceil \mathbf{c}_1 \rceil_{D_u \rightarrow q}, \mathbf{v}' \leftarrow \lceil \mathbf{c}_2 \rceil_{D_v \rightarrow q}$ 
14:   $\mathbf{m}' \leftarrow \mathbf{v}' - \langle \mathbf{u}', \mathbf{s} \rangle$ 
15:   $\mathbf{m} \leftarrow \lceil \mathbf{m}' \rceil_{q \rightarrow 2}$ 
16:  return  $\mathbf{m}$ 

```

2.2 ML-KEM

ML-KEM is obtained by applying a Fujisaki–Okamoto transform to an IND-CPA-secure public-key encryption scheme. The latter consists of Key-Gen, Encrypt, and Decrypt, summarized in Algorithm 2 [8, 20]. The message is a 256-bit string represented as a polynomial $\mathbf{m} \in \mathcal{R}_2$.

The module rank is a security parameter and is denoted by k . We write $D_u = 2^{d_u}$ and $D_v = 2^{d_v}$ for the ciphertext moduli. For $x \in \{0, \dots, a-1\}$, define

$$\lceil x \rceil_{a \rightarrow b} = \left\lceil \frac{b}{a} x \right\rceil \bmod b,$$

where $\lceil \cdot \rceil$ denotes rounding to the nearest integer. The map is applied coefficientwise to polynomials and vectors. It represents compression when $a > b$ and decompression when $a < b$. The parameters of the three standardized variants are given in Table 1.

Our attacks target Key-Gen. In the next section, we combine the public pair $(\mathbf{A}, \mathbf{b})$ with side-channel hints to estimate the cost of a primal lattice attack.

3 The Primal Attack with Hints

We translate the measured side-channel information into coordinate predictions and residual uncertainty, then estimate its effect on a primal lattice attack. This provides a common measure for comparing the leakage sources studied in Sections 4 and 5.

3.1 ML-KEM as a structured LWE instance

Expanding the polynomial products into coefficient coordinates gives a structured search-LWE instance with $N = kn$ secret coordinates and N public equations. Retaining $M \leq N$ equations yields

$$\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}, \quad \mathbf{A} \in \mathbb{Z}_q^{M \times N}.$$

Initially, the secret and error coordinates are centered and have standard deviation $\sigma_s = \sigma_e = \sigma_\chi = \sqrt{\eta_1/2}$. Exact hints remove unknown secret coordinates, while approximate hints reduce uncertainty about the coordinates that remain. All hints in this work concern $\mathbf{s}$: the implementation accumulates $\mathbf{A}\mathbf{s} + \mathbf{e}$ rather than storing $\mathbf{e}$ separately, so the error coordinates leakage is too weak to use. Consequently, σ_e is retained.

3.2 Primal lattice estimate

Let $\boldsymbol{\mu}$ be a known estimate of the secret’s center, and let $\sigma_s > 0$ describe its residual uncertainty. Without hints, $\boldsymbol{\mu} = \mathbf{0}$ and $\sigma_s = \sigma_\chi$. We balance the secret and error blocks using $\xi = \sigma_e/\sigma_s$ [5]. A centered form of Kannan’s embedding [11, 17] has row basis

$$B_{\text{emb}} = \begin{pmatrix} qI_M & 0 & 0 \\ \mathbf{A}^T & \xi I_N & 0 \\ -\mathbf{b}^T & -\xi \boldsymbol{\mu}^T & 1 \end{pmatrix}.$$

It contains the planted vector

$$\mathbf{v}_{\text{emb}} = (-\mathbf{e}^T, \xi(\mathbf{s} - \boldsymbol{\mu})^T, 1).$$

Centering changes the target vector but leaves the lattice dimension and volume unchanged:

$$d = M + N + 1, \quad \text{Vol}(\Lambda_{\text{emb}}) = q^M \xi^N.$$

Both the scaling and the centering make the secret block real-valued; the uSVP model below applies unchanged, as it depends only on the dimension, the volume, and the length of the planted vector.

We estimate the required BKZ block size using the geometric-series assumption and the root-Hermite approximation

$$\delta_\beta \approx \left(\frac{\beta}{2\pi e} (\pi\beta)^{1/\beta} \right)^{1/(2\beta-2)}.$$

Following the primal-uSVP model [2–4], we use the heuristic success condition

$$\sqrt{\beta} \sigma_e \leq \delta_\beta^{2\beta-d-1} \left(q^M \xi^N \right)^{1/d}. \quad (1)$$

For each retained equation count M , we find the smallest satisfying block size and then minimize over M ; this optimization is repeated for every hint configuration, since the optimal number of retained equations shifts as the secret block shrinks. Without hints, the model returns $\beta = 624$ for ML-KEM-768, in agreement with the standard estimates for this parameter set, and it therefore provides a consistent comparison between leakage scenarios. We express the lattice-reduction cost using the classical core-SVP convention [4, 6],

$$C_{\text{core}}(\beta) \approx 2^{0.292\beta}.$$

3.3 Incorporating side-channel hints

We model the classifier outputs as single-coordinate hints [11], separating selected value predictions from approximate information about the remaining coordinates.

Exact hints. For an observed leakage instance L , let $\mathcal{H}(L)$ contain the h highest-confidence predictions, $h < N$, and write $\mathcal{H} = \mathcal{H}(L)$ for brevity. Let $\widehat{s}_{\mathcal{H}}$ denote their predicted values. Conditional on all these predictions being correct,

$$\begin{aligned} \mathbf{b}_h &= \mathbf{b} - \mathbf{A}_{\mathcal{H}} \widehat{s}_{\mathcal{H}} \\ &= \mathbf{A}_{\overline{\mathcal{H}}} \mathbf{s}_{\overline{\mathcal{H}}} + \mathbf{e} \pmod{q}, \end{aligned}$$

where $\mathbf{A}_{\mathcal{H}}$ denotes the columns indexed by $\mathcal{H}$ and a bar denotes the complementary index set. The residual instance has $N_h = N - h$ unknown secret coordinates. An incorrect prediction changes the residual instance, so confidence in individual predictions must be distinguished from the probability that the entire selected set is correct.

Approximate hints. For each remaining coordinate, let $p_i(x | L)$ be the classifier's estimated posterior probability that $s_i = x$. Its mean and variance are

$$\begin{aligned} \mu_i(L) &= \sum_x x p_i(x | L), \\ \tau_i^2(L) &= \sum_x (x - \mu_i(L))^2 p_i(x | L). \end{aligned}$$

We use the posterior means to center the remaining secret coordinates in the embedding. Their average residual variance is

$$\sigma_{\text{rem}}(h)^2 = \mathbb{E}_L \left[\frac{1}{N_h} \sum_{i \notin \mathcal{H}(L)} \tau_i^2(L) \right].$$

The expectation is estimated from the attack-set posteriors after hint selection. Thus, coordinates promoted to exact hints no longer contribute to the residual variance.

For estimation, we approximate the centered residual secret by an isotropic distribution with variance $\sigma_{\text{rem}}(h)^2$. This approximation neglects inter-coordinate correlations and differences between coordinate variances; its accuracy also depends on the calibration of the estimated posteriors. Its effect on the estimate is conservative: a coordinatewise balancing would scale coordinate i by σ_{χ}/τ_i , so that the volume depends on the geometric mean of the residual widths, which never exceeds the average used here. A larger residual width yields a smaller volume and hence a larger block size, so the reported figures do not understate the security of the residual instance. We retain the prior error width $\sigma_e = \sigma_{\chi}$, giving

$$\xi_h = \frac{\sigma_{\chi}}{\sigma_{\text{rem}}(h)}, \quad d_h = M + N_h + 1, \quad \text{Vol}(\Lambda_h) = q^{M_{\chi} N_h}.$$

Substituting these parameters into Equation (1) gives the estimated block size for each operating point.

The reported core-SVP costs describe the lattice-reduction component conditional on all selected exact hints being correct. They exclude profiling costs and any additional enumeration or correction of incorrect hints. Hint-set success probabilities must therefore be considered separately when interpreting the overall attack.

4 Sign Leakage from the CBD Sampler

At Line 3 of Key-Gen, the CBD sampler generates each coefficient of $\mathbf{s}$ and $\mathbf{e}$ as

$$a = \sum_{t=1}^{\eta_1} a_t, \quad b = \sum_{t=1}^{\eta_1} b_t, \quad v = a - b,$$

where a_t, b_t are independent uniform bits. Thus, $v \in \{-\eta_1, \dots, \eta_1\}$; we write $\text{neg}(v)$ for the binary class $v < 0$ versus $v \geq 0$.

The m4fspeed sampler has two modes. With $\text{add}=0$, it stores v to form $\mathbf{s}$; with $\text{add}=1$, it adds v to the existing coefficient of $\mathbf{As}$, storing $\mathbf{As} + \mathbf{e}$. The compiler emits separate loop bodies, each with the inner loop over coefficient positions fully unrolled. Both modes extract a and b identically. Figure 1 shows the source-level samplers.

Secret coefficients are stored as signed 16-bit two's-complement integers. Negative values set most bits, whereas small nonnegative values set only low bits:

$$-1 = 0\text{x}\text{FFFF}, \quad +1 = 0\text{x}0001, \quad +2 = 0\text{x}0002.$$

In particular, $+1$ and $+2$ have the same Hamming weight, suggesting that stores reveal the binary sign more strongly than the magnitude.

4.1 Measurement setup

We target the pqm4 m4fspeed implementations of ML-KEM-512 and ML-KEM-768 [18] on an STM32F303RCT7 Arm Cortex-M4. A ChipWhisperer-Lite CW1173 records four samples per 7.385 MHz clock cycle (29.54 MS/s). The ADC is clocked from the target, so no trace realignment is needed. Profiling and attack traces are acquired from the same device.

Each outer-loop iteration of `cbd2` derives $G = 8$ coefficients from a 32-bit word, and each iteration of `cbd3` derives $G = 4$ from a 24-bit word. We analyze complete groups, so every within-group position $j \in \{0, \dots, G-1\}$ is equally represented. For a leakage sample L_t at time index t and class label Y , we use

$$\text{SNR}_t(Y) = \frac{\text{Var}_Y[\mathbb{E}(L_t | Y)]}{\mathbb{E}_Y[\text{Var}(L_t | Y)]}.$$

For each coefficient we take the maximum of $\text{SNR}_t(Y)$ over t , and report the median of these maxima across the selected coefficients. Independently shuffled labels estimate the noise floor for each model and path. We measure only the first component polynomials s_0 and e_0 ; the remaining components follow the same code paths and are expected to behave similarly.

4.2 Leakage of the secret and error paths

Using 8000 traces, Figure 2a compares ML-KEM-768 leakage models for the secret and error paths over 32 coefficients selected as complete groups. Figure 2b shows coefficient-value SNRs for all 256 coefficients of s_0 . For s_0 , the coefficient-value, Hamming-weight, and binary-sign models have nearly identical median peak SNRs. Within the nonnegative class $v \geq 0$, the magnitude SNR is roughly two orders of magnitude below the unconditional sign SNR. The store therefore reveals mainly the sign.

For e_0 , leakage from the intermediates a and b remains detectable, especially when the class label is the pair (a, b) . Leakage of the sampled value v is much weaker at write-back, where the implementation stores coefficients of $\mathbf{As} + \mathbf{e}$. The accumulated store therefore reveals much less about the sampled error, although earlier intermediates still leak.

The per-coefficient results also show strong dependence on the position within an iteration. This motivates a classifier that processes a complete sampler iteration and accounts for the different coefficient positions.

```

void cbd2(poly *r, const unsigned char *buf, int add){
    unsigned int i, j;
    uint32_t t, d;
    int16_t a, b;

    for (i = 0; i < 256 / 8; i++) {
        t = load32_littleendian(buf + 4 * i);
        d = t & 0x55555555;
        d += (t >> 1) & 0x55555555;

        for (j = 0; j < 8; j++) {
            a = (d >> (4 * j)) & 0x3;
            b = (d >> (4 * j + 2)) & 0x3;

            if(!add)
                r->coeffs[8 * i + j] = 0;

            r->coeffs[8 * i + j] += (a - b); } }
}

```

(a) $\eta_1 = 2$: ML-KEM-768.

```

void cbd3(poly *r, const unsigned char *buf, int add) {
    unsigned int i, j;
    uint32_t t, d;
    int16_t a, b;

    for (i = 0; i < 256 / 4; i++) {
        t = load24_littleendian(buf + 3 * i);
        d = t & 0x00249249;
        d += (t >> 1) & 0x00249249;
        d += (t >> 2) & 0x00249249;

        for (j = 0; j < 4; j++) {
            a = (d >> (6 * j + 0)) & 0x7;
            b = (d >> (6 * j + 3)) & 0x7;

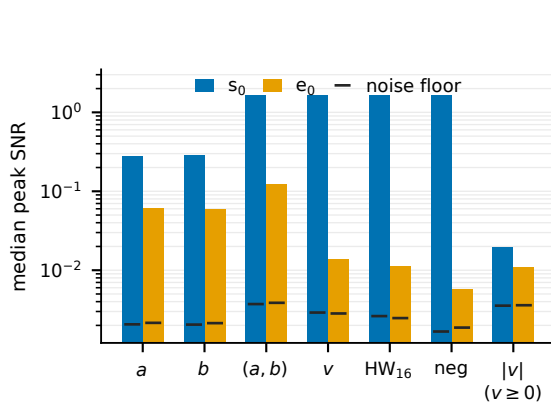
            if(!add)
                r->coeffs[4 * i + j] = 0;

            r->coeffs[4 * i + j] += (a - b); } }
}

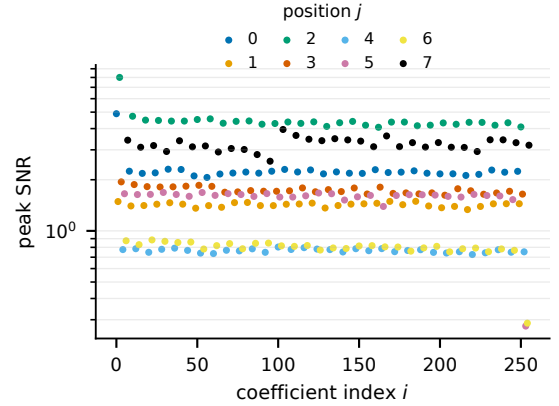
```

(b) $\eta_1 = 3$: ML-KEM-512.

Figure 1: Simplified C representations of the two CBD samplers.



(a) Leakage models and their path-specific noise floors.



(b) Coefficient-value SNR, colored by within-group position.

Figure 2: ML-KEM-768 CBD leakage from 8000 traces. Black marks in (a) show model- and path-specific noise-floor estimates. The $|v|$ ($v \geq 0$) model in (a) is evaluated within the $v \geq 0$ class; all other models are unconditional.

4.3 Profiled recovery and hint quality

We combine the predictions of two independently trained one-dimensional CNNs. Each processes a window covering a complete sampler iteration, using shared convolutional layers and one classification head per coefficient position. Each head estimates a posterior distribution over the possible coefficient values. We use 10 200 profiling and 1800 attack traces for ML-KEM-512, and 51 000 profiling and 9000 attack traces for ML-KEM-768. Each held-out trace is attacked independently.

Coefficient-value prediction accuracy reaches about 73% for ML-KEM-512 and 76% for ML-KEM-768. Binary-sign recovery is nearly perfect: we observe no sign errors for ML-KEM-512 and accuracy above 99.99% for ML-KEM-768. Before selecting exact hints, the

residual widths are approximately $\sigma_{\text{rem}}(0) = 0.64$ and 0.53 , respectively, with uncertainty remaining almost entirely in the magnitude.

For exact hints, individual prediction accuracy must be distinguished from the probability that every selected prediction is correct. For each attack trace, we rank the 256 coefficient predictions for s_0 by their maximum posterior probability. Figure 3 reports the fraction of traces whose top m predictions are all correct. At $m = 128$, half a polynomial, no entirely correct set is observed for either parameter set.

For ML-KEM-768's error polynomial e_0 , we use 17 000 profiling and 3000 attack traces to target the intermediates a and b . The network predicts one of the nine possible pairs (a, b) , then sums the

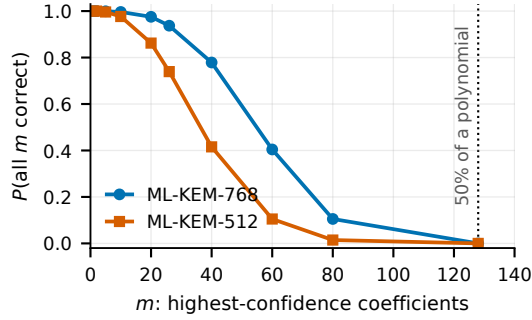


Figure 3: Empirical success probability for the m highest-confidence CBD predictions. No entirely correct set is observed at $m = 128$.

probabilities of pairs with the same difference $v = a - b$. Coefficient-value prediction accuracy reaches about 41%, compared with the prior baseline of 37.5% obtained by always predicting zero. Sign accuracy and residual width remain close to their prior baselines. No attack trace yields an entirely correct set of its 20 highest-confidence coefficient predictions. The evaluated attack therefore provides only limited information about the error.

The CBD window provides reliable binary-sign hints for s but insufficient magnitude information for large sets of correct coefficient-value predictions. We next examine the NTT for the missing magnitude information.

5 Magnitude Leakage from the NTT

The CBD window reliably identifies $\text{neg}(v)$ but provides much less information about $|v|$. The NTT follows immediately: the implementation calls `polyvec_ntt(&skpv)`, reloads the secret coefficients, and multiplies them by public twiddle factors. Since $\zeta \cdot 1$ and $\zeta \cdot 2$ generally have different word-level representations, this multiplication creates a complementary magnitude channel.

We evaluate this window on ML-KEM-768 using 4000 traces of `poly_ntt(s[0])`. We reserve 3400 traces for profiling and 600 for attack. For a coefficient v , let SNR_{mag} denote the SNR of $|v|$ evaluated on the subset $v \geq 0$. Combined results use the sign recovered by the measured CBD attack.

5.1 Magnitude leakage and butterfly structure

The NTT has no constant per-coefficient stride, so we locate one point of interest (POI) per coefficient using only profiling traces and extract a 160-sample window around it. The POI depends on when the coefficient first reaches the upper input of a Cooley–Tukey butterfly.

A coefficient with index $i \geq 128$ meets a twiddle in the first layer while still equal to the raw secret value. Coefficients with $64 \leq i < 128$ are mixed once before their first multiplication, and those with $i < 64$ are mixed further before being multiplied (see Algorithm 1). In our target implementation [18], packed even and odd half-words also follow different instructions, producing different leakage effects.

The median SNR_{mag} decreases from 0.473 in the first layer to 0.228 in the second layer and 0.035 in later layers. The reason

for decrease in the SNR is mixing that is happening in each layer. Across all coefficients, the median is 0.175, compared with 0.018 in the matched CBD experiment: a $9.5\times$ increase in magnitude information.

5.2 Profiled recovery and exact hints

We train one pooled one-dimensional CNN over all coefficient-specific windows. It predicts $v \in \{-2, -1, 0, 1, 2\}$; when the CBD sign is available, the posterior is restricted to $v < 0$ or $v \geq 0$ and renormalized.

Exact recovery reaches 0.8677 from the NTT alone and 0.9076 after adding the measured CBD sign. The latter is within 0.0005 of the 0.9081 true-sign result, while the matched-trace CBD baseline is 0.7528. Thus, the gain comes primarily from NTT magnitude leakage, not from assuming a sign oracle.

Figure 4a shows that per-coefficient accuracy follows SNR_{mag} closely. Figure 4b gives the exact-hint budget: the 128 most-confident predictions, corresponding to half of s_0 , are all correct with probability 0.93, whereas the CBD probability is zero.

5.3 Why narrow operands are vulnerable

To distinguish leakage caused by small values from leakage made informative by low entropy, we run the same `ntt_fast` implementation on the five-valued secret and on near-uniform coefficients of $\mathbf{A}s + \mathbf{e}$, using 2000 traces for each. Let $\text{HW}_{16}(x)$ denote the Hamming weight of the signed 16-bit two’s-complement representation of x .

Figure 4c shows comparable HW_{16} leakage for both distributions. In contrast, the value modulo 8 leaks strongly for the secret but falls below the noise floor for the near-uniform input. Thus, the physical leakage is not specific to small operands; it becomes value-revealing because a CBD coefficient has only five possibilities.

This observation motivates increasing the entropy of the representative entering the multiplication, as developed in the next section.

5.4 Estimated lattice complexity

At a selected confidence threshold, let h be the number of coefficients supplied as exact hints over the complete secret vector. Each such hint removes one unknown coordinate. The remaining information is summarized by the posterior standard deviation σ_{rem} ; without leakage, $\sigma_{\text{rem}} = \sigma_{\chi} = 1$ for ML-KEM-768. For each operating point, we apply the dimension and width updates of Section 3.3 to (1). In particular,

$$N_h = N - h, \quad \xi_h = \frac{\sigma_{\chi}}{\sigma_{\text{rem}}}, \quad \text{Vol}(\Lambda_h) = q^M \xi_h^{N_h}.$$

We then report the smallest satisfying BKZ block size β , optimized over the number M of retained equations.

We select the operating points from the measured confidence curves while accounting for the reliability of the proposed exact hints. Table 2 then reports the lattice-reduction component using the complexity convention fixed in Section 3.

Most of the reduction comes from the NTT window, with the CBD sign providing a further refinement. The cost estimate assumes that the unmeasured component polynomials behave like s_0

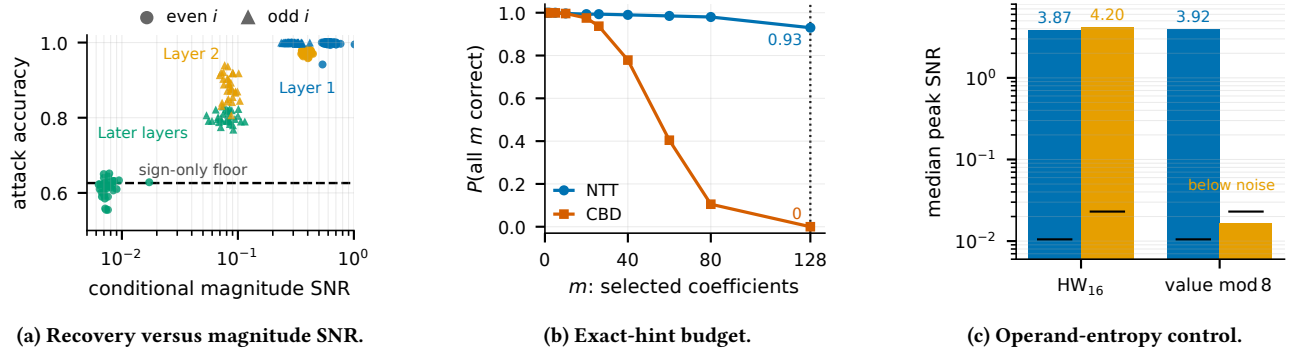


Figure 4: Three views of NTT leakage in ML-KEM-768. In (a), accuracy follows the conditional magnitude leakage across butterfly layers; circles and triangles denote even and odd coefficient indices. In (b), the NTT provides 128 error-free predictions with probability 0.93, while the CBD probability is zero. In (c), blue bars represent the five-valued secret and orange bars the near-uniform $As + e$ input; short black ticks mark their respective noise floors.

because they execute the same code; it also assumes same-device profiling.

The two windows are therefore complementary: the CBD sampler reveals the binary sign, while the NTT supplies the missing magnitude information.

6 A Simple Countermeasure

Section 5 showed that the NTT leakage becomes useful because its input is drawn from a narrow distribution. This suggests a local countermeasure: broaden the representation produced by the sampler before the secret reaches the NTT. The construction below changes neither the represented residue nor any downstream code.

6.1 Construction

Instead of storing a secret coefficient as v , we choose a random representative of the same residue class,

$$\tilde{v} = v + rq, \quad r \xleftarrow{\$} \{-6, \dots, -1\} \cup \{1, \dots, 6\},$$

independently for each coefficient. Since $\tilde{v} \equiv v \pmod{q}$, all subsequent computations produce the same result modulo q and no explicit unmasking is needed. We randomize only the $\text{add}=0$ path that produces s . The $\text{add}=1$ path accumulates the error into the already near-uniform value As and did not provide useful stored-value leakage in Section 4.

The randomizer is symmetric, so the sign of $\tilde{v}$ is determined by r rather than by v . It also excludes zero, which would leave the original short coefficient exposed. The range is limited by the lazy

NTT. A conservative bound on a signed 16-bit intermediate is

$$|v| + |r|q + \frac{7q}{2} < 2^{15}.$$

For ML-KEM-768, where $|v| \leq 2$, this permits $|r| \leq 6$; the next value can overflow. The chosen set therefore provides $\log_2 12 = 3.585$ bits of entropy per coefficient. The implementation enforces the range bound at compile time.

```
void patched_cbd2(poly *r, const unsigned char *buf, int add){
    unsigned int i, j;
    uint32_t t, d;
    int16_t a, b, R;
    int32_t masked_a;

    for (i = 0; i < 256 / 8; i++) {
        t = load32_littleendian(buf + 4 * i);
        d = t & 0x55555555;
        d += (t >> 1) & 0x55555555;

        for (j = 0; j < 8; j++) {
            a = (d >> (4 * j)) & 0x3;
            b = (d >> (4 * j + 2)) & 0x3;

            if (!add){
                R = non_zero_rand_between(-6, 6);
                masked_a = (int32_t)a + R * KYBER_Q;
                OPT_BARRIER(masked_a);
                r->coeffs[8 * i + j] = (int16_t)(masked_a - b);
            }
            else
                r->coeffs[8 * i + j] += (a - b); } }
}
```

Figure 5: Simplified code for the patched CBD sampler with $\eta_1 = 2$. OPT_BARRIER denotes the compiler barrier that preserves the shown evaluation order; random-number-generation details are omitted.

Table 2: Selected ML-KEM-768 operating points and their estimated lattice complexity.

source	h	σ_{rem}	β	complexity estimate
none	0	1.000	624	$2^{182.2}$
CBD	162	0.588	432	$2^{126.1}$
NTT	508	0.805	150	$2^{43.8}$
NTT + CBD sign	521	0.545	129	$2^{37.7}$

6.2 Implementation and correctness

The order of operations is essential. If the sampler first computes $v = a - b$ and then adds rq , the unprotected difference still appears

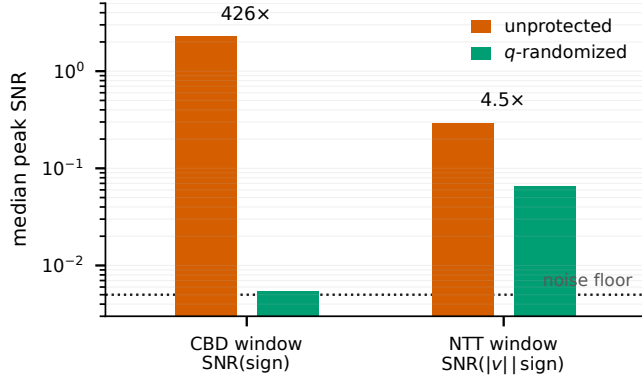


Figure 6: Leakage before and after q -randomization in ML-KEM-768: the CBD-window sign SNR and the NTT-window conditional magnitude SNR, relative to the estimated noise floor (dotted). The sampler channel reaches the noise floor.

Table 3: Attack outcome with and without the countermeasure (ML-KEM-768). Accuracy baselines are 0.375 without a sign and 0.624 when only the sign is given.

metric	unprotected	q -randomized
exact accuracy, sign unknown	0.868	0.309
exact accuracy, sign given	0.908	0.571
σ_{rem} , sign given	0.317	0.535
$P(\text{all } 10)$	0.997	0.028
$P(\text{all } 128)$	0.930	0.000

in a register. We instead form $a + rq$ before subtracting b , as shown in Figure 5. At -03 , however, GCC may reassociate $(a + rq) - b$ as $(a - b) + rq$. The abstract OPT_BARRIER in the figure prevents this rewrite; inspection of the generated assembly confirms that the randomized intermediate is formed first.

This distinction was visible in measurement: applying the randomizer after the difference left substantial sign leakage, even though value-level leakage models favored the chosen randomizer. Such models describe a value, not the instruction sequence that places it on a register or bus, and therefore cannot replace inspection of the compiled code.

6.3 Evaluation

We assess the sampler window using 3000 traces and seven stratified coefficients. The NTT experiment uses 4000 traces, with 85% reserved for profiling. Figure 6 shows that the protected sampler reaches its estimated noise floor, while the conditional magnitude leakage in the NTT is reduced by a factor of 4.5.

We evaluate the unchanged classifier without supplying the sign, because the protected sampler no longer provides a reliable sign oracle. As shown in Table 3, its accuracy does not exceed the corresponding prior baseline, and long sets of error-free predictions disappear. Applying the estimate of Section 3 to the measured posteriors raises the required block size from 150 to 611, close to the unaided value 624 (Figure 7). Under the complexity estimate of

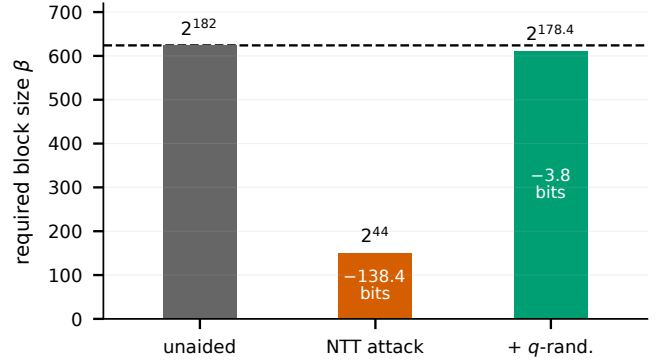


Figure 7: Required BKZ block size β for ML-KEM-768: unaided, under the single-trace NTT attack, and with q -randomization. The dashed line marks the unaided estimate.

Section 3, this leaves approximately four bits of estimated security loss.

The prototype changes only the sampler. Its static instruction count increases from 114 to 212, and the firmware image grows by 937 bytes; the NTT, and later operations are unchanged.

6.4 Limitations

The proposed countermeasure is an empirical leakage-reduction technique, not a masking scheme with a security proof. We evaluate first-order leakage and our profiled attack on one implementation and device; more advanced attacks remain outside our scope.

Consider a Cooley–Tukey butterfly with randomized inputs

$$\tilde{u} = u + r_u q, \quad \tilde{v} = v + r_v q.$$

It computes

$$t = \text{PlantardMul}(\zeta, \tilde{v}), \quad (\tilde{u}, \tilde{v}) \mapsto (\tilde{u} + t, \tilde{u} - t).$$

Since $\zeta \tilde{v} \equiv \zeta v \pmod{q}$, the reduced product t is not randomized. The multiple $r_v q$ is therefore removed by the Plantard reduction, whereas $r_u q$ remains in both lazy outputs:

$$(\tilde{u} + t, \tilde{u} - t) = (u + t + r_u q, u - t + r_u q).$$

Thus, random offsets propagate through the lazy additions but are progressively removed by twiddle multiplications.

This residual product leakage is consistent with the measured results: the sampler sign SNR decreases by more than 400 $\times$ to the noise floor, whereas the NTT magnitude SNR decreases by 4.5 $\times$. Nevertheless, the probability that the ten highest-confidence predictions are all correct falls to 0.028.

Future work should investigate whether residual leakage from multiple butterflies can be combined using graph-based methods such as belief propagation [24]. In particular, it remains open whether leakage from the early NTT layers is sufficient before the secret coefficients become increasingly mixed.

Acknowledgments

This work was supported by the European Union under Grant Agreement No. 101225691 QARC.

References

- [1] Amin Abdulrahman, Vincent Hwang, Matthias J. Kannwischer, and Amber Sprenkels. 2022. Faster Kyber and Dilithium on the Cortex-M4. In *Applied Cryptography and Network Security – ACNS 2022 (Lecture Notes in Computer Science, Vol. 13269)*. Springer, 853–871. doi:10.1007/978-3-031-09234-3_42
- [2] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. 2017. Revisiting the Expected Cost of Solving uSVP and Applications to LWE. In *Advances in Cryptology—ASIACRYPT 2017 (Lecture Notes in Computer Science, Vol. 10624)*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer, 297–322. doi:10.1007/978-3-319-70694-8_11
- [3] Martin R. Albrecht, Rachel Player, and Sam Scott. 2015. On the Concrete Hardness of Learning with Errors. *Journal of Mathematical Cryptology* 9, 3 (2015), 169–203. doi:10.1515/jmc-2015-0016
- [4] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2016. Post-quantum Key Exchange—A New Hope. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 327–343. <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim>
- [5] Shi Bai and Steven D. Galbraith. 2014. Lattice Decoding Attacks on Binary LWE. In *Information Security and Privacy—ACISP 2014 (Lecture Notes in Computer Science, Vol. 8544)*, Willy Susilo and Yi Mu (Eds.). Springer, 322–337. doi:10.1007/978-3-319-08344-5_21
- [6] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. 2016. New Directions in Nearest Neighbor Searching with Applications to Lattice Sieving. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*. Society for Industrial and Applied Mathematics, 10–24. doi:10.1137/1.9781611974331.ch2
- [7] Estuardo Alpirez Bock, Gustavo Banegas, Chris Brzuska, Lukasz Chmielewski, Kirithivasan Puniyamurthy, and Milan Sorf. 2024. Breaking DPA-Protected Kyber via the Pair-Pointwise Multiplication. In *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 14584)*, Christina Pöpper and Lejla Batina (Eds.). Springer, 101–130. doi:10.1007/978-3-031-54773-7_5
- [8] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2021. CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation, Version 3.01. <https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf> Round-3 submission to the NIST Post-Quantum Cryptography Standardization Project.
- [9] Munkhbaatar Chinbat, Liji Wu, and Xiangmin Zhang. 2026. Transformer Network for Side-Channel Analysis of the NTT Accelerator in PQC CRYSTALS-Kyber. In *2026 5th International Conference on Electronics, Integrated Circuits and Communication Technology (EICCT)*. 840–843. doi:10.1109/EICCT69950.2026.11564609
- [10] Munkhbaatar Chinbat, Liji Wu, Xiangmin Zhang, Yifan Yang, and Man Wei. 2025. Comparative Deep Learning-Based Side-Channel Analysis of an FPGA-Based CRYSTALS-Kyber NTT Accelerator. *Cryptography* 9, 4 (2025), 64. doi:10.3390/cryptography9040064
- [11] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. 2020. LWE with Side Information: Attacks and Concrete Security Estimation. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12171)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, 329–358. doi:10.1007/978-3-030-56880-1_12
- [12] Pierre-Alain Fouque, Damien Marion, Quyen Nguyen, and Alexandre Wallet. 2025. Avengers assemble! Supervised learning meets lattice reduction: A single power trace attack against CRYSTALS-Kyber Key Generation. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025, 4 (Sep. 2025), 409–436. doi:10.46586/tches.v2025.i4.409-436
- [13] Daniel Heinz, Matthias J. Kannwischer, Georg Land, Thomas Pöppelmann, Peter Schwabe, and Amber Sprenkels. 2022. First-Order Masked Kyber on ARM Cortex-M4. Cryptology ePrint Archive, Paper 2022/058. <https://eprint.iacr.org/2022/058>
- [14] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreier Rodosek. 2023. Belief Propagation Meets Lattice Reduction: Security Estimates for Error-Tolerant Key Recovery from Decryption Errors. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023, 4 (2023), 287–317. doi:10.46586/TCHES.V2023.I4.287-317
- [15] Julius Hermelink, Silvan Streitz, Erik Mårtensson, and Richard Petri. 2025. A Generic Framework for Side-Channel Attacks Against LWE-Based Cryptosystems. In *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques Madrid, Spain, May 4-8, 2025, Proceedings, Part VIII (Lecture Notes in Computer Science, Vol. 15608)*, Serge Fehr and Pierre-Alain Fouque (Eds.). Springer, 3–32. doi:10.1007/978-3-031-91101-9_1
- [16] Junhao Huang, Jipeng Zhang, Haosong Zhao, Zhe Liu, Ray C. C. Cheung, Çetin Kaya Koç, and Donglong Chen. 2022. Improved Plantard Arithmetic for Lattice-based Cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022, 4 (2022), 614–636. doi:10.46586/TCHES.V2022.I4.614-636
- [17] Ravi Kannan. 1987. Minkowski’s Convex Body Theorem and Integer Programming. *Mathematics of Operations Research* 12, 3 (1987), 415–440. doi:10.1287/moor.12.3.415
- [18] Matthias J. Kannwischer, Richard Petri, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. [n. d.]. PQM4: Post-Quantum Crypto Library for the ARM Cortex-M4. <https://github.com/mupq/pqm4>.
- [19] Yen-Ting Kuo and Atsushi Takayasu. 2024. A Lattice Attack on CRYSTALS-Kyber with Correlation Power Analysis. In *Information Security and Cryptology – ICISC 2023*, Hwajeong Seo and Suhri Kim (Eds.). Springer Nature Singapore, Singapore, 202–220.
- [20] National Institute of Standards and Technology. 2024. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Federal Information Processing Standards Publication NIST FIPS 203. U.S. Department of Commerce. doi:10.6028/NIST.FIPS.203
- [21] Eric Chun-Yu Peng and Markus G. Kuhn. 2025. Adaptive Template Attacks on the Kyber Binomial Sampler. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025, 3 (Jun. 2025), 470–492. doi:10.46586/tches.v2025.i3.470-492
- [22] Peter Pessl and Robert Primas. 2019. More Practical Single-Trace Attacks on the Number Theoretic Transform. In *Progress in Cryptology – LATINCRYPT 2019 (Lecture Notes in Computer Science, Vol. 11774)*. Springer, 130–149. doi:10.1007/978-3-030-30530-7_7
- [23] Thomas Plantard. 2021. Efficient Word Size Modular Arithmetic. *IEEE Trans. Emerg. Top. Comput.* 9, 3 (2021), 1506–1518. doi:10.1109/TETC.2021.3073475
- [24] Robert Primas, Peter Pessl, and Stefan Mangard. 2017. Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption. In *Cryptographic Hardware and Embedded Systems – CHES 2017 (Lecture Notes in Computer Science, Vol. 10529)*. Springer, 513–533. doi:10.1007/978-3-319-66787-4_25
- [25] Zehua Qiao, Yuejun Liu, Yongbin Zhou, Mingyao Shao, and Shuo Sun. 2023. When NTT Meets SIS: Efficient Side-channel Attacks on Dilithium and Kyber. Cryptology ePrint Archive, Paper 2023/1866. <https://eprint.iacr.org/2023/1866>
- [26] Prasanna Ravi, Romain Poussier, Shivam Bhasin, and Anupam Chattopadhyay. 2020. On Configurable SCA Countermeasures Against Single Trace Attacks for the NTT: A Performance Evaluation Study over Kyber and Dilithium on the ARM Cortex-M4. In *Security, Privacy, and Applied Cryptography Engineering (Lecture Notes in Computer Science)*. Springer, 123–146. doi:10.1007/978-3-030-66626-2_7
- [27] Prasanna Ravi, Bolin Yang, Shivam Bhasin, Fan Zhang, and Anupam Chattopadhyay. 2023. Fiddling the Twiddle Constants – Fault Injection Analysis of the Number Theoretic Transform. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2023, 2 (2023), 447–481. doi:10.46586/tches.v2023.i2.447-481
- [28] Tobias Schneider, Clara Paglialonga, Tobias Oder, and Tim Güneysu. 2019. Efficiently Masking Binomial Sampling at Arbitrary Orders for Lattice-Based Crypto. In *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 11443)*, Dongdai Lin and Kazuo Sako (Eds.). Springer, 534–564. doi:10.1007/978-3-030-17259-6_18
- [29] Kai Wang, Dejun Xu, and Jing Tian. 2025. An Improved Two-Step Attack on Lattice-Based Cryptography: A Case Study of Kyber. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 44, 9 (2025), 3643–3647. doi:10.1109/TCAD.2025.3550443
- [30] Timo Zijlstra, Karim Bigou, and Arnaud Tisserand. 2019. FPGA Implementation and Comparison of Protections Against SCAs for RLWE. In *Progress in Cryptology – INDOCRYPT 2019 (Lecture Notes in Computer Science, Vol. 11898)*, Feng Hao, Sushmita Ruj, and Sourav Sen Gupta (Eds.). Springer, Cham, 535–555. doi:10.1007/978-3-030-35423-7_27

A Open Science

We provide the source code, analysis scripts, and device-captured traces with the submission. The accompanying README explains how to reproduce the analyses and figures. Measurements can also be repeated with matching hardware and acquisition settings, although noise and device variation may cause small differences in the results.