On the Expected Number of Iterations in ML-DSA Signing

A Complete Accounting of the Rejection Conditions

Kris Kwiatkowski

PQShield
kris@amongbytes.com

Abstract. FIPS 204 Table 1 gives the expected number of iterations of the ML-DSA signing loop (ML-DSA.Sign_internal, Algorithm 7), citing CRYSTALS-Dilithium Equation (5). That equation covers only the validity checks on line 23 ($\|z\|_\infty$ and $\|r_0\|_\infty$) and omits the two encoding-correctness checks on line 28 ($\|c \cdot t_0\|_\infty$ and the hint weight), a scoping distinction the Dilithium specification states explicitly and FIPS 204 does not reproduce. We give a complete accounting of all four conditions. The z check on line 23 is computed exactly by counting; the $c \cdot t_0$ term follows a closed-form Irwin-Hall tail, restating a peer-reviewed result by Yan, Luo and Wang; the hint-weight term's per-coefficient probability, conditioned on line 23 having passed, does have a closed form, but the tail of the sum over $256k$ such coefficients does not, since their probabilities are unequal, random, and correlated with each other, so we give the conditioned per-coefficient probability in closed form and estimate its tail by Monte Carlo. The corrected expected iteration counts are 4.36, 5.14 and 3.91 for ML-DSA-44, -65 and -87, matching a real end-to-end measurement to within 0.14% and already reflected in NIST's own errata to FIPS 204.

1 Introduction

FIPS 204 [1] Table 1 gives the expected number of iterations of the ML-DSA signing loop, and Appendix C uses those figures to derive a loop bound for implementations that choose to impose one. The discrepancy is small but systematic. NIST's errata [2] corrects the numbers without a derivation. The figures in Table 1 of [1] account for **one** of the two rejection points in the signing loop. The missing one is a consequence of public-key compression. Correcting it changes the expected iteration counts by roughly 1-3%.

The change does not affect the security of ML-DSA; only the runtime cost changes.

Related work The error was first reported on the pqc-forum mailing list by Becker [3], with analysis by Saarinen showing the figures account only for the line 23 checks. Saarinen reported corrected figures $\{4.358, 5.137, 3.905\}$ from a full accounting but did not publish the derivation [3]; this note supplies one. The

$c \cdot t_0$ rejection was first analysed by Hamburg and Xiao [4], who estimated its probability by convolving the distribution of t_0 with itself and confirmed the estimate experimentally. Yan, Luo and Wang [5] subsequently gave the same result in closed form, via the Irwin-Hall distribution. The hint-rejection term was analysed by Yan, Luo and Wang [5], but under an unconditioned model: each coefficient's hint probability is computed as if r_0 could take any value in $(-\gamma_2, \gamma_2]$, which overstates that probability (defined here as P_{hint}) by 13-32% because it ignores that line 23 has already confined r_0 to a narrower range by the time hints are computed.

This note contributes a hint-rejection probability conditioned on the line 23 checks having already passed, correcting the unconditioned model used in [5]. The corrected model is stated in terms of the per-coefficient probability, and the tail is estimated by Monte Carlo. It also gives the exact product form for probability of surviving both security checks (defined here as P_{norm}), showing that CRYSTALS-Dilithium Equation (5) [6] overstates the per-attempt acceptance probability by about 1%, and completes the accounting into the corrected Table 1 figures and the Appendix C loop bound.

2 Where the Signing Loop Rejects

ML-DSA is built on Fiat-Shamir with Aborts. A signing attempt produces a candidate signature, checks it against several bounds, and discards the attempt if any bound is violated. Algorithm 7 of [1] has this shape, with irrelevant lines elided.

Note the structure: lines 25-28 sit inside the `else` branch, reached only by attempts that already survived line 23. Section 6 uses this to condition the hint-rejection probability on line 23 having passed.

This listing is numbered independently of FIPS 204 and appears below as "Algorithm 1", while reproducing FIPS 204's Algorithm 7.

Algorithm 1 ML-DSA.Sign_internal(sk, M', rnd)

Require: private key sk, formatted message M', randomness rnd
Ensure: signature σ

1: $(\rho, K, tr, s_1, s_2, t_0) \leftarrow \mathrm{skDecode}(sk)$
2: $\hat{s}_1 \leftarrow \mathrm{NTT}(s_1)$
3: $\hat{s}_2 \leftarrow \mathrm{NTT}(s_2)$
4: $\hat{t}_0 \leftarrow \mathrm{NTT}(t_0)$
5: $\hat{A} \leftarrow \mathrm{ExpandA}(\rho)$ // stored in NTT form
6: $\mu \leftarrow \mathrm{H}(\mathrm{BytesToBits}(tr) \| M', 64)$ // message representative
7: $\rho'' \leftarrow \mathrm{H}(K \| rnd \| \mu, 64)$ // private random seed
8: $\kappa \leftarrow 0$
9: $(z, h) \leftarrow \perp$
10: **while** $(z, h) = \perp$ **do**
11: $\quad y \leftarrow \mathrm{ExpandMask}(\rho'', \kappa)$
12: $\quad w \leftarrow \mathrm{NTT}^{-1}(\hat{A} \circ \mathrm{NTT}(y))$
13: $\quad w_1 \leftarrow \mathrm{HighBits}(w)$ // signer's commitment
14: $\quad$ // HighBits applied componentwise
15: $\quad \tilde{c} \leftarrow \mathrm{H}(\mu \| \mathrm{w1Encode}(w_1), \lambda/4)$ // commitment hash
16: $\quad c \leftarrow \mathrm{SampleInBall}(\tilde{c})$ // verifier's challenge
17: $\quad \hat{c} \leftarrow \mathrm{NTT}(c)$
18: $\quad \langle\langle cs_1 \rangle\rangle \leftarrow \mathrm{NTT}^{-1}(\hat{c} \circ \hat{s}_1)$
19: $\quad \langle\langle cs_2 \rangle\rangle \leftarrow \mathrm{NTT}^{-1}(\hat{c} \circ \hat{s}_2)$
20: $\quad z \leftarrow y + \langle\langle cs_1 \rangle\rangle$ // signer's response
21: $\quad r_0 \leftarrow \mathrm{LowBits}(w - \langle\langle cs_2 \rangle\rangle)$
22: $\quad$ // LowBits applied componentwise
23: $\quad$ **if** $\|z\|_\infty \geq \gamma_1 - \beta$ **or** $\|r_0\|_\infty \geq \gamma_2 - \beta$ **then**
24: $\quad\quad (z, h) \leftarrow \perp$ // rejection 1: validity
25: $\quad$ **else**
26: $\quad\quad \langle\langle ct_0 \rangle\rangle \leftarrow \mathrm{NTT}^{-1}(\hat{c} \circ \hat{t}_0)$
27: $\quad\quad h \leftarrow \mathrm{MakeHint}(-\langle\langle ct_0 \rangle\rangle, w - \langle\langle cs_2 \rangle\rangle + \langle\langle ct_0 \rangle\rangle)$ // signer's hint.
28: $\quad\quad$ **if** $\|\langle\langle ct_0 \rangle\rangle\|_\infty \geq \gamma_2$ **or** number of 1's in $h > \omega$ **then**
29: $\quad\quad\quad (z, h) \leftarrow \perp$ // rejection 2: after MakeHint applied componentwise
30: $\quad\quad$ **end if**
31: $\quad$ **end if**
32: $\quad \kappa \leftarrow \kappa + \ell$ // increment counter
33: **end while**
34: $\sigma \leftarrow \mathrm{sigEncode}(\tilde{c}, z \bmod q, h)$
35: **return** σ

There are four rejection conditions in total, and they fall into two quite different categories. The two on line 23 are required for the scheme to be secure. Without them the distribution of z would depend on the private key s_1, and an attacker watching signatures could recover it. The rejection is exactly what makes the output distribution independent of the secret - it is the "Aborts" in Fiat-Shamir with Aborts.

The two on line 28 have nothing to do with security. ML-DSA publishes only the high part of t, dropping $d = 13$ low bits into t_0, and the signer must transmit hints so the verifier can reconstruct what was dropped. If an attempt needs more hints than the signature format reserves room for, the signature is perfectly secure but cannot be encoded, so it gets discarded. The fourth condition similarly protects the correctness of the hint mechanism.

Line	Condition	Character
23	$\|z\|_\infty \geq \gamma_1 - \beta$	security-mandatory
23	$\|r_0\|_\infty \geq \gamma_2 - \beta$	security-mandatory
28	$\#\{i : h_i = 1\} > \omega$	encoding (compression)
28	$\|c \cdot t_0\|_\infty \geq \gamma_2$	hint correctness (compression)

Table 1: The four rejection conditions of ML-DSA.Sign_internal.

The Dilithium [6] specification scopes this precisely: it describes its Equation (5) as "the probability that Step 21 passes", the line 23 checks only, and bounds the hint-weight rejection separately and only loosely, "between 1 and 2%", with no formula given. FIPS 204 reproduces the resulting number without that qualifier, stating that Table 1 gives "the expected repetitions in the rejection sampling loop" as a whole.

3 Correcting the Acceptance Probability

Write p for the probability that a single attempt is accepted. Because the checks happen in two groups, with the second evaluated only if the first passed:

$$\begin{aligned} p &= P(\text{line 23 passes}) \cdot P(\text{line 28 passes} \mid \text{line 23 passed}) \\ &= P_{norm} \cdot (1 - P_{ct_0}) \cdot (1 - P_{hint}) \end{aligned}$$

where P_{norm} is the probability of surviving both line 23 checks, P_{ct_0} the probability that $\|c \cdot t_0\|_\infty$ reaches γ_2, and P_{hint} the probability of producing more than ω hints. The three terms turn out to need three completely different techniques: exact counting, a closed-form distribution, and simulation.

4 The Security Checks

Line 23 rejects unless every coefficient of z and every coefficient of $r0$ is inside its bound. Both are handled by the same counting argument.

4.1 The z check

The condition is $\|z\|_\infty < \gamma_1 - \beta$, where $z = y + c \cdot s_1$. Take one coefficient $z_i = y_i + (c \cdot s_1)_i$. The only randomness is y_i, which $ExpandMask$ (Algorithm 34, [1]) draws uniformly from $2 \cdot \gamma 1$ possible values. The permitted range in which z_i may land in, is an open interval $(-(\gamma_1 - \beta), \gamma_1 - \beta)$, hence there are $2(\gamma_1 - \beta) - 1$ admissible values.

This count does not depend on the value of $(c \cdot s_1)_i$. The per-coefficient probability is therefore the same whatever the secret key happens to be:

$$P(\text{coefficient of } z \text{ passes}) = \frac{2(\gamma_1 - \beta) - 1}{2\gamma_1}$$

This factor is **exact**. It is a counting argument over a distribution that is uniform by construction.

Lemma 1. *Let y_i be drawn uniformly from the $2\gamma_1$ values* `ExpandMask` *can produce for one coefficient, and let $z_i = y_i + (c \cdot s_1)_i$. Then*

$$\Pr[|z_i| < \gamma_1 - \beta] = \frac{2(\gamma_1 - \beta) - 1}{2\gamma_1},$$

independent of the value of $(c \cdot s_1)_i$.

Proof. The admissible values of y_i are those for which $z_i = y_i + (c \cdot s_1)_i$ lands in the open interval $(-(\gamma_1 - \beta),\ \gamma_1 - \beta)$ of width $2(\gamma_1 - \beta)$, i.e. a window of that width shifted by $-(c \cdot s_1)_i$ within the $2\gamma_1$ values y_i ranges over. Since $\beta = \tau\eta$ and $(c \cdot s_1)_i$ is a sum of τ terms each bounded by η, $|(c \cdot s_1)_i| \leq \beta$, so the shifted window never slides off either end of the y_i range and always contains exactly $2(\gamma_1 - \beta) - 1$ integer values, regardless of $(c \cdot s_1)_i$.

4.2 The r_0 check

The condition is $\|r_0\|_\infty < \gamma_2 - \beta$, with $r_0 = LowBits(w - c \cdot s_2)$ taking one of $2 \cdot \gamma_2$ possible values. The identical count gives

$$P(\text{coefficient of } r_0 \text{ passes}) = \frac{2(\gamma_2 - \beta) - 1}{2\gamma_2}$$

This factor is **not** exact. It assumes the low bits of $w - c \cdot s_2$ are uniformly distributed over their residues. They are a deterministic function of A, y, c and s_2 - so this is a modeling assumption (a heuristic). It is resting on the expectation that modular reduction of a pseudorandom quantity scrambles the low-order bits. (Decompose also treats $r - r_0 = q - 1$ as a special case, which breaks r_0's uniformity at that one point even if the scrambling is perfect - negligible, since it is one value out of $q \approx 2^{23}$.)

4.3 Putting them together

An attempt passes line 23 only if *every* coefficient passes. One violation anywhere kills the attempt. There are $256 \cdot l$ coefficients in z and $256 \cdot k$ in r_0, treated as independent, so:

$$P_{norm} = \left[\frac{2(\gamma_1 - \beta) - 1}{2\gamma_1}\right]^{256\ell} \cdot \left[\frac{2(\gamma_2 - \beta) - 1}{2\gamma_2}\right]^{256k}$$

The compounding effect may seem counterintuitive. Each coefficient is rejected with probability well under 0.1% - between roughly one in 1,200 and one in 4,300. But there are between two and four thousand coefficients, and all must pass at once. Since $(1-x)^N \approx e^{-Nx}$ and $Nx \approx 1.4$, survival collapses to about a quarter. The same compounding explains Equation (5)'s error: a per-coefficient discrepancy of only $1/(2\gamma)$ - a few parts per million - accumulates over thousands of coefficients into roughly 0.5-1% in P_{norm}.

Variant	P_{norm}	**Equation (5)**	**Overstatement**
ML-DSA-44	0.23272	0.23502	1.0%
ML-DSA-65	0.19539	0.19631	0.5%
ML-DSA-87	0.25809	0.25961	0.6%

Table 2: Probability of passing the line 23 validity checks, per attempt. The exact product is compared against Equation (5) of the Dilithium (v3.1) [6] specification, which approximates it.

Two separate problems have now been isolated. Equation (5) is an *approximation* of P_{norm}, good to about a percent - the exact product fixes that precision issue. But Equation (5) is also *only* P_{norm}, whereas FIPS 204 presents it as the whole loop; that is a scoping issue, and fixing it needs the two remaining terms.

5 The Hint-Correctness Check

This is the smallest of the three terms. It comes before P_{hint} because both depend on the distribution of a coefficient of $c \cdot t_0$, which is derived here.

5.1 What the check does

ML-DSA compresses its public key. The full value $t = A \cdot s_1 + s_2$ is split by *Power2Round* (Algorithm 35 of FIPS 204 [1]) into a high part t_1, which is published, and a low part t_0, which is not. The verifier therefore cannot compute the high bits of $A \cdot z - c \cdot t$ directly, and the signer sends hints recording where the missing $c \cdot t_0$ term causes a carry across a high-bit boundary.

The verifier does not know t_0, so the value it computes is off by $c \cdot t_0$. One hint bit per coefficient can correct this only if $c \cdot t_0$ is small enough that it never carries a coefficient more than one high-bits boundary away. Dilithium's Lemma 1 [6] makes this precise: UseHint recovers the signer's high bits exactly when the perturbation has infinity norm at most $\alpha/2 = \gamma_2$, so each coefficient of $c \cdot t_0$ must stay within that bound. So before trusting the hint, the signer checks $\|c \cdot t_0\|_\infty < \gamma_2$, and if the check fails it discards the attempt, because the signature might not verify.

5.2 The distribution of a coefficient of $c \cdot t_0$

Power2Round defines $t_0 = t \bmod^{\pm} 2^d$, where $\bmod^{\pm}$ is the centred representative. So the coefficients of t_0 lie in

$$(-2^{d-1},\ 2^{d-1}] = (-4096,\ 4096] \quad \text{for } d = 13$$

That is $2^d = 8192$ consecutive integers, **centred on zero**. The centring makes the distribution essentially zero-mean, which is what lets $c \cdot t_0$ be treated as a zero-mean sum. Assuming these coefficients are uniform over their 8192 values, the variance of a uniform over N consecutive integers is $(N^2 - 1)/12$, so

$$\mathrm{Var}(t_{0,i}) = \frac{2^{2d} - 1}{12} \approx 5{,}592{,}405 \qquad \sigma(t_{0,i}) \approx 2365$$

The challenge polynomial c, produced by *SampleInBall* (Algorithm 29 of FIPS 204 [1]), has exactly τ non-zero coefficients, each $+1$ or -1, the rest zero. So each coefficient of the product $c \cdot t_0$ is a signed sum of exactly τ coefficients of t_0.

Two properties follow:

- **Its spread.** Each coefficient of $c \cdot t_0$ adds up τ independent coefficients of t_0, so their variances add:

  $$\mathrm{Var} = \tau \cdot \frac{2^{2d} - 1}{12} \qquad \sigma = \sqrt{\mathrm{Var}} \approx 2^d \sqrt{\tau/12}$$

  That is 14768, 16554 and 18318 for the three parameter sets - a typical coefficient of $c \cdot t_0$ is in the tens of thousands. The threshold γ_2 is 6.4, 15.8 and 14.3 times larger than that, so the check rarely fails. But σ alone does not say how rarely.

- **Its range.** Each term is at most $2^{d-1} = 4096$ in absolute value, so

  $$|(c \cdot t_0)_i| \le \tau \cdot 2^{d-1}$$

which is 159744, 200704 and 245760. For ML-DSA-65 and ML-DSA-87 this maximum is below $\gamma_2 = 261888$: the check cannot fail, and $P_{ct_0} = 0$ exactly. For ML-DSA-44 the maximum exceeds $\gamma_2 = 95232$, so the tail has to be computed.

5.3 Use of Irwin-Hall distribution

Yan, Luo and Wang [5] give this same closed form for $\|c \cdot t_0\|_\infty$ using the Irwin-Hall distribution, including the 7.335×10^{-9} value for ML-DSA-44 and the observation that the check is inactive for the other two parameter sets.

Most distributions require numerical integration to evaluate the fraction of probability lying below a threshold x. The Irwin-Hall distribution instead has a closed form:

$$P(\mathrm{IH}_n \leq x) = \frac{1}{n!} \sum_{j=0}^{\lfloor x \rfloor} (-1)^j \binom{n}{j} (x-j)^n$$

Here n is the number of values summed, which is τ, the number of non-zero coefficients in the challenge, and x is the fixed threshold of interest. A coefficient of $c \cdot t_0$ *is* a sum of uniform values, τ coefficients of t_0, added with $\pm$ signs, so the Irwin-Hall distribution applies directly, up to the discrete-to-continuous correction below. For ML-DSA-65 and ML-DSA-87, the threshold lies past the end of the distribution's support, so the probability is exactly zero.

Two adjustments are needed before the formula applies. First, the signs of c can be ignored: $t_0 \in (-2^{d-1}, 2^{d-1}]$ is symmetric about zero up to its single boundary point, a 1-in-2^d effect negligible at this precision, so negating it leaves its distribution essentially unchanged. Second, Irwin-Hall is defined for uniforms on [0,1], whereas the coefficients of t_0 are uniform on an interval of width 2^d. Dividing through by that width converts between the two, up to a shift of $\frac{1}{2}$ per term, which puts the threshold at $x = \frac{\gamma_2}{2^d} + \frac{\tau}{2}$ in Irwin-Hall units.

The one remaining approximation is modelling the discrete sum of t_0 coefficients by the continuous Irwin-Hall distribution, whose error we do not bound. Evaluating its CDF in exact rational arithmetic (below) makes that evaluation exact.

Variant	Threshold x	Irwin-Hall support	P_{ct_0}
ML-DSA-44	31.125	[0, 39]	7.33e-09
ML-DSA-65	56.469	[0, 49]	0
ML-DSA-87	61.969	[0, 60]	0

Table 3: P_{ct_0}, computed via the Irwin-Hall distribution. For ML-DSA-65 and ML-DSA-87 the threshold falls outside the support of the distribution, so the tail is exactly zero.

For ML-DSA-65 and ML-DSA-87 the threshold lands beyond the support of the distribution, 56.47 against a maximum of 49, and 61.97 against a maximum of 60, and the tail beyond the support of a distribution is exactly zero. For those two parameter sets, the condition cannot fire at all, consistent with the ACVP test-vector specification's restriction of the check to ML-DSA-44 [7].

For ML-DSA-44, adding the $256k$ per-coefficient tail probabilities gives 7.334967×10^{-9}; computing 1 minus the probability that every coefficient stays within the threshold - exact only under an assumed independence between coefficients - gives 7.334961×10^{-9} instead. The two agree to six significant figures, two more than Yan, Luo and Wang's reported 7.335×10^{-9} [5], so the simpler sum is used.

In theory, this check could be skipped for ML-DSA-65 and -87, but in practice omitting it creates a deviation from FIPS 204.

6 The Hint-Weight Rejection

Line 27 of the algorithm computes a vector of hints, one bit per coefficient, so h holds $256 \cdot k$ bits. Line 28 rejects when the number of 1 bits exceeds ω - the number of hint positions the signature format reserves space for. Line 28 is only reached once line 23 has already passed, so N and its distribution are implicitly conditioned on that. Writing N for that count:

$$P_{hint} = P(N > \omega \mid \text{line 23 passed})$$

Unlike P_{norm} and P_{ct_0}, which ask whether *any* coefficient crosses a threshold, P_{hint} asks whether the *count* of hint-producing coefficients exceeds ω. That is a tail of a sum of non-identically distributed indicators across thousands of coefficients, which does not factorize. This term has no closed form.

6.1 Condition to fire single hint

N is a count of individual coefficients that fired, so computing anything about it starts with a single one. Here we analyse how likely is a hint at one position.

MakeHint(z, r) returns 1 when HighBits(r) differs from HighBits$(r + z)$. Line 27 invokes it as

$$h = \text{MakeHint}(-c \cdot t_0,\ w - c \cdot s_2 + c \cdot t_0)$$

so $r = w - c \cdot s_2 + c \cdot t_0$ and $r + z = w - c \cdot s_2$. A hint at position i therefore fires exactly when adding $(c \cdot t_0)_i$ changes the high bits of $(w - c \cdot s_2)_i$ - that is, when its low part is pushed out of $(-\gamma_2,\ \gamma_2]$.

That low part is r_0, already checked on line 23. Lines 25-28 live inside the `else` branch, reached only when line 23 already passed. So by the time hints are

computed, r_0 is not free to take any value in $(-\gamma_2,\ \gamma_2]$; it is already confined to $|r_0| < \gamma_2 - \beta$.

With that constraint in place, the probability that a single coefficient fires can be computed directly. Take one coefficient and write $a = |(c \cdot t_0)_i|$, $a > 0$, and consider first the case $(c \cdot t_0)_i = +a$. The hint fires when adding it pushes the low part out of $(-\gamma_2, \gamma_2]$:

$$r_0 + a > \gamma_2 \qquad \text{i.e.} \qquad r_0 > \gamma_2 - a$$

So the question is how many values of r_0 satisfy that. Unconditionally r_0 could be anywhere in $(-\gamma_2, \gamma_2]$ - but line 23 has already rejected everything with $|r_0| \geq \gamma_2 - \beta$, so the largest value still in play is $\gamma_2 - \beta - 1$. The firing values are therefore the integers in a window with a lower edge at $\gamma_2 - a + 1$ and an upper edge at $\gamma_2 - \beta - 1$. Counting that window:

$$(\gamma_2 - \beta - 1) - (\gamma_2 - a + 1) + 1 = a - \beta - 1$$

That count, $a - \beta - 1$, is for the case $(c \cdot t_0)_i = +a$; once $a \leq \beta + 1$ it is zero, and no hint is possible at all. The mirror case $(c \cdot t_0)_i = -a$ pushes r_0 out of the interval on the other side, $r_0 - a \leq -\gamma_2$, and the identical argument against that boundary gives a count of $a - \beta$ instead - one larger, because $(-\gamma_2, \gamma_2]$ is open at $-\gamma_2$ but closed at γ_2, so the two boundaries are not interchangeable.

$$\underbrace{\overbrace{{\scriptstyle -(\gamma_2-\beta)+1\ \cdots\ -\gamma_2+a}}^{\text{hint fires } (-a)} \;\Big|\; \overbrace{{\scriptstyle -\gamma_2+a+1\ \cdots\ \gamma_2-a}}^{\text{no hint}} \;\Big|\; \overbrace{{\scriptstyle \gamma_2-a+1\ \cdots\ \gamma_2-\beta-1}}^{\text{hint fires } (+a)}}_{2(\gamma_2-\beta)-1 \text{ values permitted by line 23}}$$

$t_0 \in (-2^{d-1}, 2^{d-1}]$ is symmetric about zero except for its single excluded boundary value (Section 5) - one value out of 2^d, negligible at this precision - so the sign of $(c \cdot t_0)_i$ is, to the same precision, a fair coin flip given its magnitude a.

The probability that a single coefficient fires therefore follows by averaging the two counts, stated formally below.

Lemma 2. *Condition on line 23 of Algorithm 1 having accepted, so that r_0 is uniform over the $2(\gamma_2-\beta)-1$ integers in $(-(\gamma_2-\beta), \gamma_2-\beta)$. Let $a = |(c \cdot t_0)_i| > 0$ for one coefficient. Then*

$$P(\textit{hint at } i \mid a, \textit{line 23 of Algorithm 1 passed}) = \frac{\max(0,\ a - \beta - \frac{1}{2})}{2(\gamma_2 - \beta) - 1}$$

Proof. **Case 1:** $(c \cdot t_0)_i = +a$, the hint fires iff $r_0 + a > \gamma_2$, that is $r_0 > \gamma_2 - a$. But conditioning on the fact that line 23 passed, r_0 can't be bigger than $\gamma_2 - \beta - 1$.

Hence, value r_0 takes is one of $(\gamma_2-\beta-1)-(\gamma_2-a+1)+1 = a-\beta-1$ possible values. The other escape, $r_0 + a \leq -\gamma_2$ is unreachable, since $r_0 > -(\gamma_2 - \beta) > -\gamma_2 - a$. **Case 2:** $(c \cdot t_0)_i = -a$, the hint fires iff $r_0 - a \leq -\gamma_2$, that is $r_0 \leq -\gamma_2 + a$. But conditioning on the fact that line 23 passed, r_0 can't be smaller than -$(\gamma_2 - \beta) + 1$. Hence, value r_0 takes is one of $(-\gamma_2 + a) - (-(\gamma_2 - \beta) + 1) + 1 = a - \beta$ possible values. The other escape, $r_0 - a > \gamma_2$, is unreachable, since $r_0 \leq \gamma_2 - \beta - 1 < \gamma_2 + a$. Combining **Case 1** and **Case 2** and noticing that $t_0 \in (-2^{d-1}, 2^{d-1}]$ except for its single boundary value 2^{d-1}, the sign of $(c \cdot t_0)_i$ is a fair coin flip. The probability that the hint fires is therefore the average of the two counts, divided by the $2(\gamma_2 - \beta) - 1$ possible r_0 values.

$$P(\text{hint at } i \mid a) = \frac{\frac{1}{2}\left[(a-\beta-1) + (a-\beta)\right]}{(2(\gamma_2 - \beta) - 1)} = \frac{a - \beta - \frac{1}{2}}{2(\gamma_2 - \beta) - 1}.$$

When $a \leq \beta$, no admissible r_0 satisfies either case, so the probability is 0, giving the $\max(0, \cdot)$ above.

As a consequence, no hint is possible unless $|(c \cdot t_0)_i|$ exceeds β - which is why the mean hint count sits well below ω in every parameter set. Ignoring line 23 and using $\frac{a}{2\gamma_2}$ instead is the unconditioned model Yan, Luo and Wang [5] use for this term; the table below compares the two on shared (c, t_0) draws.

Variant	**Conditioned**	**Unconditioned**	**Overstatement**
ML-DSA-44	0.01454 ± 0.00005	0.01644 ± 0.00006	13.1%
ML-DSA-65	0.00379 ± 0.00003	0.00500 ± 0.00003	31.9%
ML-DSA-87	0.00776 ± 0.00004	0.00922 ± 0.00004	18.8%

Table 4: Conditioned (this note) vs. unconditioned [5] P_hint, compared on shared (c, t0) draws. 5,000,000 trials per parameter set.

The unconditioned model overstates P_{hint} by 13 to 32 percent across the three parameter sets - the gap depends on how much of the conditioned window line 23 has already consumed, which varies by parameter set.

The conditioned column above and the final P_{hint} estimate below are independent Monte Carlo runs at different trial counts - 5,000,000 here, 20,000,000 there - so their point estimates differ slightly; for ML-DSA-87 they are 0.00776 ± 0.00004 against 0.00785 ± 0.00002, about two combined standard errors apart, consistent with sampling noise.

There is no simple formula for $P(N > \omega)$. N is a sum of $256 \cdot k$ indicator variables, but three features rule out the standard routes to a closed form: the per-coefficient probabilities differ (ruling out a binomial tail), those probabilities are themselves random since $c \cdot t_0$ varies with the key and the challenge (ruling

out a fixed-probability model), and all 256 coefficients of one $c \cdot t_0$ polynomial share the same t_0 and c, so they are correlated rather than independent (ruling out a Poisson-binomial treatment).

A normal approximation to N, using its true mean and variance, underestimates the tail:

Variant	**Mean N**	σ	**True**	**Normal**	**Underestimate**
ML-DSA-44	63.06	7.74	0.01454	0.01211	16.7%
ML-DSA-65	38.23	6.12	0.00379	0.00239	36.9%
ML-DSA-87	56.75	7.45	0.00776	0.00590	24.0%

Table 5: Simulated P_{hint} vs. a mean/variance-matched normal approximation. 5,000,000 trials per parameter set.

A normal with the same mean and variance underestimates the tail by 17 to 37 percent, because N is right-skewed. Instead we estimate P_{hint} by simulation.

6.2 Monte Carlo

Each trial simulates one signing attempt that has already passed line 23:

1. Sample c: τ positions among 256, each ± 1, rest zero
2. Sample t_0: k polynomials of 256 coefficients, uniform on $(-2^{d-1},\ 2^{d-1}]$
3. Compute $c \cdot t_0$ in $\mathbb{Z}[X]/(X^{256}+1)$
4. For each of the $256 \cdot k$ coefficients, compute its hint probability and draw the Bernoulli outcome
5. N = number of hints; record whether $N > \omega$

P_{hint} is the fraction of trials in which N exceeded ω. The product in step 3 is computed by negacyclic FFT, values stay at or below $\tau \cdot 2^{d-1}$ - at most 245,760, for ML-DSA-87's $\tau = 60$ - far under $q/2$, so no modular reduction is needed and double precision is amply accurate.

Variant	**Mean hint count**	ω	P_{hint}
ML-DSA-44	63.1	80	0.01458 ± 0.00003
ML-DSA-65	38.2	55	0.00380 ± 0.00001
ML-DSA-87	56.8	75	0.00785 ± 0.00002

Table 6: P_{hint}, estimated by Monte Carlo. 20,000,000 trials per parameter set; the stated uncertainty is the binomial standard error.

The mean hint counts sit comfortably below ω in every case - the parameter sets were chosen that way - and it is only the upper tail of the count that spills over. In a separate, earlier 100-million-trial run the simulation recorded zero occurrences of the $\|c \cdot t_0\|_\infty \geq \gamma_2$ condition for ML-DSA-65 and ML-DSA-87, and exactly one for ML-DSA-44 - consistent with the P_{ct_0} figures above, where that condition is identically zero for the first two and on the order of 10^{-9} per signature for the third. (This check does not depend on the hint-probability formula, so it is unaffected by which run produced it; the 20-million-trial run behind Table 6 above recorded zero such occurrences for all three, also consistent with that rate.)

7 Results and Verification

Combining P_{norm}, P_{ct_0} and P_{hint} from the previous sections gives the per-attempt acceptance probability and the corrected expected iteration count for each parameter set:

Variant	P_{norm}	P_{ct_0}	P_{hint}	p	$\frac{1}{p}$
ML-DSA-44	0.23272	7.3e-09	0.01458	0.22933	4.360
ML-DSA-65	0.19539	0	0.00380	0.19465	5.137
ML-DSA-87	0.25809	0	0.00785	0.25606	3.905

Table 7: Per-attempt results, combining the three terms derived above.

The derived values agree with a real measurement done by using [8] to 0.14%, 0.06% and 0.05%, in each case well within the measurement's own standard error. That is consistent with the modelling assumptions, the two uniformity heuristics and the independence assumptions in the Monte Carlo. For ML-DSA-44, even a scope-corrected P_{norm} that keeps Equation (5)'s approximation (mean 4.318) is already 4 standard errors from the measured 4.354. For ML-DSA-65 the test is weaker, the raw FIPS 204 figure of 5.1 is only 3.3 standard errors away, and that standard error is itself an underestimate (see below).

Variant	**Derived**	**Measured**	**FIPS 204**
ML-DSA-44	4.360	4.354 ± 0.009	4.25
ML-DSA-65	5.137	5.134 ± 0.010	5.1
ML-DSA-87	3.905	3.903 ± 0.008	3.85

Table 8: Comparison against the figures published in FIPS 204 Table 1 and against a real mldsa-ref measurement.

"Measured" is a real end-to-end run with 64 keys, 3,125 signatures each. The standard error above treats each signature as independent, which understates the true uncertainty since P_{hint} depends on the key through t_0.

Given p, the number of attempts until acceptance is geometrically distributed, and the full distribution follows directly, with no further derivation or assumption beyond what already goes into p.

Iterations	**ML-DSA-44**	**ML-DSA-65**	**ML-DSA-87**
1	22.93%	19.46%	25.61%
2	40.61%	35.14%	44.66%
3	54.23%	47.77%	58.83%
4	64.72%	57.93%	69.37%
5	72.81%	66.12%	77.21%
6	79.05%	72.72%	83.05%
7	83.85%	78.03%	87.39%
8	87.56%	82.30%	90.62%
9	90.41%	85.75%	93.02%
10	92.61%	88.52%	94.81%
11	94.30%	90.76%	96.14%
12	95.61%	92.56%	97.13%

Table 9: Probability of completing the signing process within a given number of iterations.

Every variant reaches at least 90% completion within 11 iterations, but the tail runs long: ML-DSA-65 needs 22 iterations to reach 99%, against an expected 5.14 - more than four times as many. For anyone budgeting worst-case latency or energy on a constrained device, the tail matters more than the mean.

Target	**ML-DSA-44**	**ML-DSA-65**	**ML-DSA-87**
90%	9	11	8
95%	12	14	11
99%	18	22	16

Table 10: Iterations required to reach a given probability of completing the signing process.

8 The Appendix C Loop Bound

FIPS 204 lets implementations bound the number of loop iterations, and Appendix C derives the smallest bound whose probability of being reached is at most 2^{-256}.

It solves $(\frac{5.1-1}{5.1})^n \leq 2^{-256}$, using the rounded Table 1 figure 5.1 (two significant digits), and obtains $n = 814$.

NIST's errata [2] replaces the input with the corrected rounded figure, 5.14 (three significant digits), and by the same method obtains $n = 821$. Both figures round the expected iteration count, not the per-iteration acceptance probability, before taking the ceiling - and the direction of the effect is not fixed: the full-precision mean is 5.1375, so 5.1 rounds below it and understates n (814 instead of 820), while 5.14 rounds above it and overstates n (821 instead of 820). Carrying the full-precision mean through the same inequality instead gives $n = 820$ - one iteration tighter than the errata, and the value used in the table below.

An implementation capping at 814, the pre-errata figure, reaches the cap with probability $2^{-254.2}$ - harmless in practice, since all of 814, 820 and 821 are far beyond any reachable computation, but short of the stated 2^{-256} guarantee. Implementations that bound the loop should use 821, the errata value; 820 is the tighter figure under key averaging only, and neither is a per-key guarantee.

Quantity	**Value**
Binding acceptance probability	0.19465
Full-precision bound*	820
FIPS 204 Appendix C (pre-errata)	814
Failure probability at 814	2^-254.2
Failure probability at 820	2^-256.1

Table 11: Corrected loop bound. ML-DSA-65 is the binding case, having the lowest acceptance probability. *) Not the recommended figure - see text.

Status Against the Errata. NIST's errata [2], dated 31 July 2026, already records both corrections: the Table 1 Repetitions row becomes 4.36 / 5.14 / 3.91, matching the values derived above, and the Table 3 loop bound becomes 821. The errata states the cause as line 28 of Algorithm 7 not having been accounted for, but gives neither a derivation nor an account of why its own figure differs from the full-precision value obtained here. The one open point is the loop bound itself: for the key-averaged p computed here, 820 rather than 821 meets the stated 2^{-256} target at full precision, for the reason given in Section 8 - though, as noted there, 820 is not a per-key guarantee. The other three rows of Table 3 concern unrelated loops (`RejBoundedPoly`, `RejNTTPoly`, `SampleInBall`) and are unaffected either way.

9 Conclusion

The error in FIPS 204 Table 1 is small in magnitude, affecting only runtime, but it illustrates a general hazard in how numbers travel between documents. The derivation yields two general observations:

- P_{norm}'s z-coefficient factor is exact counting; its r_0-coefficient factor rests on a uniformity heuristic. P_{ct_0} is a closed-form distribution whose bounded support does most of the work. P_{hint} is a correlated count with no tractable closed form, and requires simulation.
- The single largest correction to P_{hint} came from observing that lines 25-28 sit inside an `else` branch, so the low part r_0 had already been constrained by line 23. That is a fact about control flow and it moves the answer by 13 to 32 %.

Limitations. The split between line 23 and line 28 makes $p = P_{norm} \cdot P(\text{line 28 passes} \mid \text{line 23 passed})$ exact, by the definition of conditional probability. The assumption is in expanding that second factor as $(1 - P_{ct_0})(1 - P_{hint})$, which requires ct0-failure and hint-failure to be independent of each other, given that line 23 has passed. For two events with known individual probabilities, however they are correlated, their joint probability cannot differ from the independence product by more than the smaller of the two - here, P_{ct_0}. So the resulting error in p is at most $P_{norm} \cdot P_{ct_0}$, about 1.7×10^{-9} for ML-DSA-44, far below the precision this derivation otherwise targets. P_{hint} itself has no closed form and is estimated by Monte Carlo, with a small residual statistical uncertainty propagating into the final figures.

Key dependence. P_{hint}, and therefore p and everything derived from it including the loop bound in Section 8, is averaged over keys: the Monte Carlo redraws t_0 on every trial, since P_{hint} depends on the key through t_0. Write p_{key} for the per-key acceptance probability and n for the candidate loop bound (e.g. 820); Section 8 chooses n so that $(1-\mathbb{E}[p_{\text{key}}])^n \leq 2^{-256}$, but the quantity that actually governs the failure rate - the probability that a signing run under a randomly chosen key still has not produced a signature after n iterations - is $\mathbb{E}[(1-p_{\text{key}})^n]$, not $(1-\mathbb{E}[p_{\text{key}}])^n$. Since $(1-p)^n$ is convex in p, Jensen's inequality gives $\mathbb{E}[(1-p_{\text{key}})^n] \geq (1-\mathbb{E}[p_{\text{key}}])^n$ always: the key-averaged calculation can only understate the true population failure probability, by an amount that depends on the key-to-key variance of p_{key}. In our end-to-end run in Section 7 (64 keys, 3,125 signatures each), the per-key P_{hint} for ML-DSA-65 has a standard deviation of about 24% of its mean (bootstrap 90% CI, roughly 14-30% given only 64 keys), estimated as the excess of the observed per-key spread over binomial sampling noise. A key one standard deviation above the mean already misses the target: 820 iterations give $2^{-255.8}$ for that key, not 2^{-256}, short of the stated bound. Every figure in this note is key-averaged, not a per-key guarantee; 820 should be read as a model-based estimate under that averaging, not a proven bound on $\mathbb{E}[(1 - p_{\text{key}})^n]$, and a full per-key analysis is left for future work.

Every number in this derivation comes from the `mldsa_rejection` package. The package is archived unmodified alongside this note [9].

Acknowledgments This work has received funding from the Chips Joint Undertaking (Chips JU) under grant agreement No. 101252658-2 (uPQComing – Enhancing Cyber-Resilience for the Upcoming Post-Quantum Era). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Chips JU.

10 References

[1] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard," U.S. Department of Commerce, Federal Information Processing Standards Publication FIPS 204, Aug. 2024. doi: 10.6028/NIST.FIPS.204.

[2] National Institute of Standards and Technology, "FIPS 204 potential updates." Jul. 31, 2026. Available: https://csrc.nist.gov/files/pubs/fips/204/final/docs/fips-204-potential-updates.xlsx

[3] H. Becker and M.-J. O. Saarinen, "ML-DSA signing failure rate." NIST pqc-forum mailing list, Jul. 2026. Available: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/BF7a9So7-1Y/m/uKFsXZx6BAAJ

[4] M. Hamburg and Q. Xiao, "Test vectors for the Dilithium $c \cdot t_0$ check." NIST pqc-forum mailing list, Mar. 2024. Available: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/G8Zf0hC-uu0/m/Kb3qNJb0AwAJ

[5] L. Yan, M. Luo, and M. Wang, "Speedup signing: Pre-rejection sampling towards dilithium," *Cybersecurity*, vol. 8, no. 1, p. 10, 2025, doi: 10.1186/s42400-024-00325-6.

[6] S. Bai *et al.*, "CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation (Version 3.1)." Feb. 2021. Available: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf

[7] C. Celi *et al.*, "ACVP ML-DSA JSON specification." Internet-Draft draft-celi-acvp-ml-dsa, 2026. Available: https://pages.nist.gov/ACVP/draft-celi-acvp-ml-dsa.html

[8] K. Kwiatkowski, "MLDSA-ref: A reference implementation of ML-DSA digital signature." Oct. 2026. Available: https://github.com/kriskwiatkowski/mldsa-ref

[9] K. Kwiatkowski, "Rejection rate of ML-DSA signing." Zenodo, 2026. doi: 10.5281/zenodo.23278885.