Annotated ML DSA Signature Algorithm

March-2026

The Annotated ML DSA Algorithms

Conversion Between Data Types

Algorithm 14 CoeffFromThreeBytes($b_0 : \mathbb{B}^1, b_1 : \mathbb{B}^1, b_2 : \mathbb{B}^1$) $\rightarrow z : Z_q \cup \perp$

Generates an element of $\{0, 1, 2, \ldots, q-1\} \cup \perp$.

1: $b_2' \leftarrow b_2$
2: **if** $b_2' > 127$ **then**
3: $\quad b_2' \leftarrow b_2' - 128$ $\hfill \triangleright$ set the top bit of b_2' to zero
4: **end if**
5: $z \leftarrow 2^{16} \cdot b_2' + 2^8 \cdot b_1 + b_0$ $\hfill \triangleright 0 \leq z \leq z^{23} - 1$
$\qquad \triangleright \max(z) = 2^{16} \cdot (2^7 - 1) + 2^8 \cdot (2^8 - 1) + (2^8 - 1)$
$\qquad \triangleright \qquad = 2^{23} - 1$
6: **if** $z < q$ **then return** z $\hfill \triangleright$ rejection sampling
7: **else return** $\perp$
8: **end if**

Algorithm 15 CoeffFromHalfByte($b : [0, 15]$) $\rightarrow z : [-\eta, \eta] \cup \perp$

Generates an element of $\{-\eta, -\eta + 1, \ldots, \eta\} \cup \{\perp\}$.
$\eta \in \{2, 4\}$.

1: **if** $\eta = 2$ **and** $b < 15$ **then return** $2 - (b \mod 5)$ $\hfill \triangleright$ rejection sampling from $\{-2, \ldots, 2\}$
$\qquad \triangleright$ case 1: ML-DSA-44 and ML-DSA-87
2: **else**
$\qquad \triangleright$ case 2: ML-DSA-65
3: $\quad$ **if** $\eta = 4$ **and** $b < 9$ **then return** $4 - b$ $\hfill \triangleright$ rejection sampling from $\{-4, \ldots, 4\}$
4: $\quad$ **else return** $\perp$
5: $\quad$ **end if**
6: **end if**

Pseudorandom Sampling

Algorithm 29 SampleInBall$(\rho : \mathbb{B}^{\lambda/4}) \rightarrow c : R_{[-1,0,1]}$

Samples a polynomial $c \in R$ with coefficients from $\{-1, 0, 1\}$ and *Hamming weight $\tau \leq 64$.*

 1: $c \leftarrow 0$
 2: $ctx \leftarrow H.\text{Init}()$ ▷ H $\doteq$ SHAKE256
 3: $ctx \leftarrow H.\text{Absorb}(ctx, \rho)$
         ▷ $len(\rho)$ is 32 bytes in ML-DSA-44, 48 in ML-DSA-65, and 64 in ML-DSA-87.
 4: $(ctx, s : \mathbb{B}^8) \leftarrow H.\text{Squeeze}(ctx, 8)$
 5: $h : \{0,1\}^{64} \leftarrow \text{BytesToBits}(s)$ ▷ h is a bit string of length 64
 6: **for** i from $256 - \tau$ to 255 **do**
         ▷ $\tau = 39, 49, 60$ in ML-DSA-44, ML-DSA-65, and ML-DSA-87, respectively.
 7:      $(ctx, j : \mathbb{B}^1) \leftarrow H.\text{Squeeze}(ctx, 1)$
 8:      **while** $j > i$ **do**   ▷ rejection sampling in $\{0, \dots, i\}$
 9:          $(ctx, j : \mathbb{B}^1) \leftarrow H.\text{Squeeze}(ctx, 1)$
10:      **end while**   ▷ j is a pseudorandom byte that is $\leq i$
11:      $c_i \leftarrow c_j$
         ▷ c_j is a smaller-degree coefficient, pseudorandomly selected.
         ▷ c_i is a larger degree coefficient. c_i receives the value of Cj.
12:      $c_j \leftarrow (-1)^{h[i+\tau-256]}$
         ▷ access pattern: $h[0], h[1], \dots, h[\tau]$ *where* $39 \leq \tau \leq 60$, *and* $h : \{0,1\}^{64}$.
         ▷ This pseudorandom shuffling is performed τ times in total.
13: **end for**
14: **return** c

Algorithm 30 RejNTTPoly$(\rho : \mathbb{B}^{34}) \rightarrow \hat{a} : T_q$

Samples a polynomial $\hat{a} \in T_q$.

 1: $j \leftarrow 0$
 2: $ctx \leftarrow G.\text{Init}()$ ▷ G $\doteq$ SHAKE128
 3: $ctx \leftarrow G.\text{Absorb}(ctx, \rho)$
 4: **while** $j < 256$ **do**
 5:      $(ctx, s : \mathbb{B}^3) \leftarrow G.\text{Squeeze}(ctx, 3)$
 6:      $\hat{a}[j] \leftarrow \text{CoeffFromThreeBytes}(s[0], s[1], s[2])$
 7:      **if** $\hat{a}[j] \neq \bot$ **then**
 8:          $j \leftarrow j + 1$
 9:      **end if**
10: **end while**
11: **return** $\hat{a}$

Algorithm 31 RejBoundedPoly($\rho : \mathbb{B}^{66}$) $\rightarrow a : R_{[-\eta,\eta]}$

Samples an element $a \in R$ with coefficients in $[-\eta, \eta]$ computed via rejection sampling from ρ.

1: $j \leftarrow 0$
2: $\text{ctx} \leftarrow \text{H.Init}()$ $\triangleright$ H $\doteq$ SHAKE256
3: $\text{ctx} \leftarrow \text{H.Absorb}(\text{ctx}, \rho)$
4: **while** $j < 256$ **do**
5:   $z : \mathbb{B}^1 \leftarrow \text{H.Squeeze}(\text{ctx}, 1)$
6:   $z_0 \leftarrow \text{CoeffFromHalfByte}(z \bmod 16)$
7:   $z_1 \leftarrow \text{CoeffFromHalfByte}(z/16)$
8:   **if** $z_0 \neq \bot$ **then**
9:    $a_j \leftarrow z_0$
10:    $j \leftarrow j + 1$
11:   **end if**
12:   **if** $z_1 \neq \bot$ **and** $j < 256$ **then**
13:    $a_j \leftarrow z_1$
14:    $j \leftarrow j + 1$
15:   **end if**
16: **end while**
17: **return** a

The Key Generation Algorithm

Algorithm 6 ML-DSA.KeyGen_internal(ξ)

1: $(\rho, \rho', K) \leftarrow \text{H}(\xi \,\|\, \text{Integer2Bytes}(k, 1) \,\|\, \text{Integer2Bytes}(\ell, 1), 128)$
2:                     $\triangleright$ expand seed
3: $\widehat{\mathbf{A}} \leftarrow \text{ExpandA}(\rho)$
4: $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow \text{ExpandS}(\rho')$
5: $\mathbf{t} \leftarrow \text{NTT}^{-1}(\widehat{\mathbf{A}} \circ \text{NTT}(\mathbf{s}_1)) + \mathbf{s}_2$
6: $(\mathbf{t}_1, \mathbf{t}_0) \leftarrow \text{Power2Round}(\mathbf{t})$ $\triangleright$ compress $\mathbf{t}$
7:           $\triangleright$ Power2Round is applied componentwise
8: $pk \leftarrow \text{pkEncode}(\rho, \mathbf{t}_1)$
9: $tr \leftarrow \text{H}(pk, 64)$
10: $sk \leftarrow \text{skEncode}(\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$
11: **return** (pk, sk)

The Annotated Key Generation Algorithm

Annotated Algorithm 6 ML-DSA.KeyGen_internal($\xi : \mathbb{B}^{32}$)

1: $(\rho : \mathbb{B}^{32}, \rho' : \mathbb{B}^{64}, K : \mathbb{B}^{32}) \leftarrow \text{H}(\xi \,\|\, \text{Integer2Bytes}(k,1) \,\|\, \text{Integer2Bytes}(\ell,1), 128)$

2:   ▷ expand seed

3: $\widehat{\mathbf{A}} : \mathbb{T}_q^{k \times \ell} \leftarrow \text{ExpandA}(\rho)$

4: $(\mathbf{s}_1 : R_m^{\ell}, \mathbf{s}_2 : R_m^{k}) \leftarrow \text{ExpandS}(\rho')$

         $m \in [-2, 2]$ *if* ML-DSA-44 or ML-DSA-87   ▷ $\eta = 2$

         $m \in [-4, 4]$ *otherwise*   ▷ $\eta = 4$

5: $\mathbf{t} : R_q^{k} \leftarrow \text{NTT}^{-1}(\widehat{\mathbf{A}} \circ \text{NTT}(\mathbf{s}_1)) + \mathbf{s}_2$

6: $(\mathbf{t}_1 : R_{q_1}^{k}, \mathbf{t}_0 : R_{q_0}^{k}) \leftarrow \text{Power2Round}(\mathbf{t})$ ▷ compress $\mathbf{t}$

         $\mathbf{t}_1 \in [0, 1023]$ ▷ 10-bit value

         $\mathbf{t}_0 \in [-4095, 4096]$ ▷ $\text{mod}^{\pm} 2^d$, $d = 13$, $([-2^{12} + 1, 2^{12}])$

7:                                 ▷ Power2Round is applied componentwise

8: $pk \leftarrow \text{pkEncode}(\rho, \mathbf{t}_1)$

9: $tr : \mathbb{B}^{64} \leftarrow \text{H}(pk, 64)$

10: $sk \leftarrow \text{skEncode}(\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$

11: **return** (pk, sk)

The Annotated Signature Algorithm

The ML-DSA Signing (Internal) algorithm named **ML-DSA.Sign_internal** in FIPS 204 standard is reproduced below for reference. The line numbers and the pseudocode matches exactly with the original version.

Algorithm 7 ML-DSA.Sign_internal(sk, M', rnd)

1: $(\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0) \leftarrow \text{skDecode}(sk)$
2: $\widehat{\mathbf{s}}_1 \leftarrow \text{NTT}(\mathbf{s}_1)$
3: $\widehat{\mathbf{s}}_2 \leftarrow \text{NTT}(\mathbf{s}_2)$
4: $\widehat{\mathbf{t}}_0 \leftarrow \text{NTT}(\mathbf{t}_0)$
5: $\widehat{\mathbf{A}} \leftarrow \text{ExpandA}(\rho)$
6: $\mu \leftarrow \text{H}(\text{BytesToBits}(tr \| M'), 64)$
7: $\rho'' \leftarrow \text{H}(K \| rnd \| \mu, 64)$
8: $\kappa \leftarrow 0$
9: $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
10: **while** $(\mathbf{z}, \mathbf{h}) = \perp$ **do**
11:     $\mathbf{y} \in R_q^\ell \leftarrow \text{ExpandMask}(\rho'', \kappa)$
12:     $\mathbf{w} \leftarrow \text{NTT}^{-1}(\widehat{\mathbf{A}} \circ \text{NTT}(\mathbf{y}))$
13:     $\mathbf{w}_1 \leftarrow \text{HighBits}(\mathbf{w})$
14: $\triangleright$ HighBits is applied componentwise
15:     $\tilde{c} \leftarrow \text{H}(\mu \| \text{w1Encode}(\mathbf{w}_1), \lambda/4)$
16:     $c \in R_q \leftarrow \text{SampleInBall}(\tilde{c})$
17:     $\widehat{c} \leftarrow \text{NTT}(c)$
18:     $\langle\!\langle c\mathbf{s}_1 \rangle\!\rangle \leftarrow \text{NTT}^{-1}(\widehat{c} \circ \widehat{\mathbf{s}}_1)$
19:     $\langle\!\langle c\mathbf{s}_2 \rangle\!\rangle \leftarrow \text{NTT}^{-1}(\widehat{c} \circ \widehat{\mathbf{s}}_2)$
20:     $\mathbf{z} \leftarrow \mathbf{y} + \langle\!\langle c\mathbf{s}_1 \rangle\!\rangle$
21:     $\mathbf{r}_0 \leftarrow \text{LowBits}(\mathbf{w} - \langle\!\langle c\mathbf{s}_2 \rangle\!\rangle)$
22: $\triangleright$ LowBits is applied componentwise
23:     **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **or** $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then** $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
24:     **else**
25:         $\langle\!\langle c\mathbf{t}_0 \rangle\!\rangle \leftarrow \text{NTT}^{-1}(\widehat{c} \circ \widehat{\mathbf{t}}_0)$
26:         $\mathbf{h} \leftarrow \text{MakeHint}(-\langle\!\langle c\mathbf{t}_0 \rangle\!\rangle, \mathbf{w} - \langle\!\langle c\mathbf{s}_2 \rangle\!\rangle + \langle\!\langle c\mathbf{t}_0 \rangle\!\rangle)$
27:                                     $\triangleright$ MakeHint is applied componentwise
28:         **if** $\|\langle\!\langle c\mathbf{t}_0 \rangle\!\rangle\|_\infty \geq \gamma_2$ **or** the number of 1's in $\mathbf{h}$ is greater than ω **then** $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
29:         **end if**
30:     **end if**
31:     $\kappa \leftarrow \kappa + \ell$
32: **end while**
33: $\sigma \leftarrow \text{sigEncode}(\tilde{c}, \mathbf{z} \bmod^{\pm} q, \mathbf{h})$
34: **return** σ

Annotated Algorithm 7 ML-DSA.Sign_internal(*sk, M', rnd*)

1: $(\rho : \mathbb{B}^{32}, K : \mathbb{B}^{32}, tr : \mathbb{B}^{64}, \mathbf{s}_1 : R_m^\ell, \mathbf{s}_2 : R_m^k, \mathbf{t}_0 : R_t^k) = \text{skDecode}(sk)$
        $\triangleright m \in [-2, 2]$ *if* ML-DSA-44 *or* ML-DSA-87 $\triangleright \eta = 2$
        $\triangleright m \in [-4, 4]$ *otherwise* $\triangleright \eta = 4$
        $\triangleright t \in [-2^{12} + 1, 2^{12} - 1]$ $\triangleright d = 13$

2: $\widehat{\mathbf{s}}_1 : \mathbb{T}_q^\ell \leftarrow \text{NTT}(\mathbf{s}_1)$

3: $\widehat{\mathbf{s}}_2 : \mathbb{T}_q^k \leftarrow \text{NTT}(\mathbf{s}_2)$

4: $\widehat{\mathbf{t}}_0 : \mathbb{T}_q^k \leftarrow \text{NTT}(\mathbf{t}_0)$

5: $\widehat{\mathbf{A}} : \mathbb{T}_q^{k \times \ell} \leftarrow \text{ExpandA}(\rho)$

6: $\mu : \mathbb{B}^{64} \leftarrow \text{H}(\text{BytesToBits}(tr \,\|\, M'), 64)$

7: $\rho'' : \mathbb{B}^{64} \leftarrow \text{H}(K \,\|\, rnd \,\|\, \mu, 64)$

8: $\kappa : \text{u16} \leftarrow 0$ $\triangleright$ unsigned 16-bit integer

9: $(\mathbf{z} : R_{mod^\pm q}^l, \mathbf{h} : R_2^k) \leftarrow \perp$

10: **while** $(\mathbf{z}, \mathbf{h}) = \perp$ **do** $\triangleright$ Rejection sampling loop

11:     $\mathbf{y} : R_q^\ell \leftarrow \text{ExpandMask}(\rho'', \kappa)$ $\triangleright$ Sample a fresh mask $\mathbf{y}$ mixing ρ'' and new κ

12:     $\mathbf{w} : R_q^k \leftarrow \text{NTT}^{-1}(\widehat{\mathbf{A}} \circ \text{NTT}(\mathbf{y}))$

13:     $\mathbf{w}_1 : R_{m_1}^k \leftarrow \text{HighBits}(\mathbf{w})$
        $\triangleright m_1 \in [0, 21]$ *if* ML-DSA-44
        $\triangleright m_1 \in [0, 15]$ *if* ML-DSA-65 or ML-DSA-87

14:                                     $\triangleright$ HighBits is applied componentwise

15:     $\tilde{c} : \mathbb{B}^{\lambda/4} \leftarrow \text{H}(\mu \,\|\, \text{w1Encode}(\mathbf{w}_1), \lambda/4)$ $\triangleright \lambda$ is the collision strength of $\tilde{c}$
        $\triangleright \lambda/4 = 32$ (ML-DSA-44) or 48 (ML-DSA-65) or 64 (ML-DSA-87)

16:     $c : R_q \leftarrow \text{SampleInBall}(\tilde{c})$ $\triangleright$ polynomial in R with coefficients from $\{-1, 0, 1\}$

17:     $\widehat{c} : T_q \leftarrow \text{NTT}(c)$

18:     $\langle\!\langle c\mathbf{s}_1 \rangle\!\rangle : R_q^\ell \leftarrow \text{NTT}^{-1}(\widehat{c} \circ \widehat{\mathbf{s}}_1)$ $\triangleright \widehat{c}$ is multiplied with each of $\widehat{\mathbf{s}}_1^0, \ldots, \widehat{\mathbf{s}}_1^{\ell-1}$

19:     $\langle\!\langle c\mathbf{s}_2 \rangle\!\rangle : R_q^k \leftarrow \text{NTT}^{-1}(\widehat{c} \circ \widehat{\mathbf{s}}_2)$ $\triangleright \widehat{c}$ is multiplied with each of $\widehat{\mathbf{s}}_2^0, \ldots, \widehat{\mathbf{s}}_2^{k-1}$

20:     $\mathbf{z} : R_{mod^\pm q}^\ell \leftarrow \mathbf{y} + \langle\!\langle c\mathbf{s}_1 \rangle\!\rangle$ $\triangleright \mathbf{z} : R_{mod^\pm q}^\ell \leftarrow \mathbf{y} : R_q^\ell + \langle\!\langle c\mathbf{s}_1 \rangle\!\rangle : R_q^\ell$

21:     $\mathbf{r}_0 : R_q^k \leftarrow \text{LowBits}(\mathbf{w} - \langle\!\langle c\mathbf{s}_2 \rangle\!\rangle)$

22:                                   $\triangleright$ LowBits is applied componentwise

23:     **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **or** $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then** $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$

24:     **else**

25:         $\langle\!\langle c\mathbf{t}_0 \rangle\!\rangle : T_q^k \leftarrow \text{NTT}^{-1}(\widehat{c} \circ \widehat{\mathbf{t}}_0)$ $\triangleright \widehat{c}$ is multiplied with each of $\widehat{\mathbf{t}}_0^0, \ldots, \widehat{\mathbf{t}}_0^{k-1}$

26:         $\mathbf{h} : R_2^k \leftarrow \text{MakeHint}(-\langle\!\langle c\mathbf{t}_0 \rangle\!\rangle, \mathbf{w} - \langle\!\langle c\mathbf{s}_2 \rangle\!\rangle + \langle\!\langle c\mathbf{t}_0 \rangle\!\rangle)$

27:                                 $\triangleright$ MakeHint is applied componentwise

28:         **if** $\|\langle\!\langle c\mathbf{t}_0 \rangle\!\rangle\|_\infty \geq \gamma_2$ **or** the number of 1's in $\mathbf{h}$ is greater than ω **then** $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$

29:         **end if**

30:     **end if**

31:     $\kappa \leftarrow \kappa + \ell$

32: **end while**

33: $\sigma \leftarrow \text{sigEncode}(\tilde{c}, \mathbf{z} \bmod^\pm q, \mathbf{h})$

34: **return** σ