Annotated ML DSA Signature Algorithm

Jan-2026

The Annotated ML DSA Algorithms

Conversion Between Data Types

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

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

$b'_2 \leftarrow b_2$
if $b'_2 > 127$ **then**
  $b'_2 \leftarrow b'_2 - 128$ ▷ set the top bit of b'_2 to zero
end if
$z \leftarrow 2^{16} \cdot b'_2 + 2^8 \cdot b_1 + b_0$ ▷ $0 \leq z \leq z^{23} - 1$
  ▷ $\max(z) = 2^{16} \cdot (2^7 - 1) + 2^8 \cdot (2^8 - 1) + (2^8 - 1)$
  ▷ $\qquad = 2^{23} - 1$
if $z < q$ **then return** z ▷ rejection sampling
else return $\perp$
end if

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

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

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

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

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

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

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

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

$j \leftarrow 0$
ctx $\leftarrow$ G.Init() ▷ G $\doteq$ SHAKE128
ctx $\leftarrow$ G.Absorb(ctx, ρ)
while $j < 256$ **do**
  (ctx, $s : \mathbb{B}^3$) $\leftarrow$ G.Squeeze(ctx, 3)
  $\hat{a}_j \leftarrow$ CoeffFromThreeBytes($s[0], s[1], s[2]$)
  if $\hat{a}[j] \neq \perp$ **then**
    $j \leftarrow j + 1$
  end if
end while
return $\hat{a}$

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

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

$j \leftarrow 0$
ctx $\leftarrow$ H.Init() ▷ H ≐ SHAKE256
ctx $\leftarrow$ H.Absorb(ctx, ρ)
while $j < 256$ **do**
  $z : \mathbb{B}^1 \leftarrow$ H.Squeeze(ctx, 1)
  $z_0 \leftarrow$ CoeffFromHalfByte($z$mod16)
  $z_1 \leftarrow$ CoeffFromHalfByte(z/16)
  if $z_0 \neq \perp$ **then**
    $a_j \leftarrow z_0$
    $j \leftarrow j + 1$
  end if
  if $z_1 \neq \perp$ **and** $j < 256$ **then**
    $a_j \leftarrow z_1$
    $j \leftarrow j + 1$
  end if
end while
return a

Algorithm 6 ML-DSA.KeyGen_internal(ξ)

$(\rho, \rho', K) \leftarrow \mathrm{H}(\xi \,\|\, \mathrm{Integer2Bytes}(k, 1) \,\|\, \mathrm{Integer2Bytes}(\ell, 1), 128)$
▷ expand seed
$\hat{\mathbf{A}} \leftarrow$ ExpandA(ρ)
$(\mathbf{s}_1, \mathbf{s}_2) \leftarrow$ ExpandS(ρ')
$t \leftarrow \mathrm{NTT}^{-1}(\hat{\mathbf{A}} \circ \mathrm{NTT}(\mathbf{s}_1)) + \mathbf{s}_2$
$(\mathbf{t}_1, \mathbf{t}_0) \leftarrow$ Power2Round($\mathbf{t}$) ▷ compress t
▷ Power2Round is applied componentwise
$pk \leftarrow$ pkEncode($\rho, \mathbf{t}_1$)
$tr \leftarrow$ H(pk, 64)
$sk \leftarrow$ skEncode($\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0$)
return (pk, sk)

Annotated Algorithm 6 ML-DSA.KeyGen_internal(ξ)

$(\rho : \mathbb{B}^{32}, \rho' : \mathbb{B}^{64}, K : \mathbb{B}^{3}2) \leftarrow \mathrm{H}(\xi \,\|\, \mathrm{Integer2Bytes}(k, 1) \,\|\, \mathrm{Integer2Bytes}(\ell, 1), 128)$

$\triangleright$ expand seed

$\hat{\mathbf{A}} : T_q^{k \times \ell} \leftarrow \mathrm{ExpandA}(\rho)$

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

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

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

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

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

  $t_{q_1} \in [0, 1023]$ $\triangleright$ 10-bit value

  $t_{q_0} \in [-4095, 4096]$ $\triangleright$ $\bmod^{\pm} 2^d$, $d = 13$, $([-2^{12} + 1, 2^{12}])$

$\triangleright$ Power2Round is applied componentwise

$pk \leftarrow \mathrm{pkEncode}(\rho, \mathbf{t}_1)$

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

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

return (pk, sk)

Pseudorandom Sampling

The Key Generation Algorithm

The Annotated Key Generation Algorithm

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)

$(\rho, K, tr, s_1, s_2, t_0) = \text{skDecode}(sk)$
$\hat{\mathbf{s}}_1 \leftarrow \text{NTT}(s_1)$
$\hat{\mathbf{s}}_2 \leftarrow \text{NTT}(s_2)$
$\hat{\mathbf{t}}_0 \leftarrow \text{NTT}(t_0)$
$\hat{\mathbf{A}} \leftarrow \text{ExpandA}(\rho)$
$\mu \leftarrow \text{H}(\text{BytesToBits}(tr \,\|\, M', 64))$
$\rho'' \leftarrow \text{H}(K \,\|\, rnd \,\|\, \mu, 64)$
$\kappa \leftarrow 0$
$(\mathbf{z}, \mathbf{h}) \leftarrow \bot$
while $(\mathbf{z}, \mathbf{h}) = \bot$ **do**
  $\text{y} \in R_q^\ell \leftarrow \text{ExpandMask}(\rho'', \kappa)$
  $\text{w} \leftarrow \text{NTT}^{-1}(\hat{\mathbf{A}} \circ \text{NTT}(\text{y}))$
  $\text{w}_1 \leftarrow \text{HighBits}(\text{w})$
  ▷ HighBits is applied componentwise
  $\tilde{c} \leftarrow \text{H}(\mu \,\|\, \text{w1Encode}(\text{w}_1), \lambda/4)$
  $c \in R_q \leftarrow \text{SampleInBall}(\tilde{c})$
  $\hat{c} \leftarrow \text{NTT}(c)$
  $\langle\langle c\mathbf{s}_1 \rangle\rangle \leftarrow \text{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{s}}_1)$
  $\langle\langle c\mathbf{s}_2 \rangle\rangle \leftarrow \text{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{s}}_2)$
  $\mathbf{z} \leftarrow \mathbf{y} + \langle\langle c\mathbf{s}_1 \rangle\rangle$
  $\mathbf{r}_0 \leftarrow \text{LowBits}(\mathbf{w} - \langle\langle c\mathbf{s}_2 \rangle\rangle)$
  ▷ LowBits is applied componentwise
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **or** $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then** $(\mathbf{z}, \mathbf{h}) \leftarrow \bot$
  else
    $\langle\langle c\mathbf{t}_0 \rangle\rangle \leftarrow \text{NTT}^{-1}(\hat{c} \circ \hat{\text{t}}_0)$
    $\mathbf{h} \leftarrow \text{MakeHint}(-\langle\langle c\mathbf{t}_0 \rangle\rangle, \mathbf{w} - \langle\langle c\mathbf{s}_2 \rangle\rangle + \langle\langle c\mathbf{t}_0 \rangle\rangle)$
    ▷ MakeHint is applied componentwise
    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** $(\mathbf{z}, \mathbf{h}) \leftarrow \bot$
    end if
  end if
  $\kappa \leftarrow \kappa + \ell$
end while
$\sigma \leftarrow \text{sigEncode}(\tilde{c}, \mathbf{z} \bmod^{\pm} q, \mathbf{h})$
return σ

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

$(\rho : \mathbb{B}^{32},\ K : \mathbb{B}^{32},\ tr : \mathbb{B}^{64},\ s_1 : R_m^{\ell},\ s_2 : R_m^{k},\ t_0 : R_t^{k}) = \text{skDecode}(sk)$
  $m \in [-2, 2]$ *if* ML-DSA-44 *or* ML-DSA-87 ▷ $\eta = 2$
  $m \in [-4, 4]$ *otherwise* ▷ $\eta = 4$
  $t \in [-2^{12}+1, 2^{12}-1]$ ▷ $d = 13$
$\hat{\mathbf{s}}_1 : T_q^{\ell} \leftarrow \text{NTT}(s_1)$
$\hat{\mathbf{s}}_2 : T_q^{k} \leftarrow \text{NTT}(s_2)$
$\hat{\mathbf{t}}_0 : T_q^{k} \leftarrow \text{NTT}(t_0)$
$\hat{\mathbf{A}} : T_q^{k\times\ell} \leftarrow \text{ExpandA}(\rho)$
$\mu : \mathbb{B}^{64} \leftarrow \text{H}(\text{BytesToBits}(tr \,\|\, M'), 64))$
$\rho'' : \mathbb{B}^{64} \leftarrow \text{H}(K \,\|\, rnd \,\|\, \mu, 64)$
$\kappa : \text{u16} \leftarrow 0$ ▷ unsigned 16-bit integer
$(\mathbf{z} : R^{l}_{mod^{\pm}q},\ \mathbf{h} : R_2^{k}) \leftarrow \bot$
while $(\mathbf{z}, \mathbf{h}) = \bot$ **do** ▷ Rejection sampling loop
  $\mathbf{y} : R_q^{\ell} \leftarrow \text{ExpandMask}(\rho'', \kappa)$ ▷ Sample a fresh mask y mixing ρ'' and new κ
  $\mathbf{w} : R_q^{k} \leftarrow \text{NTT}^{-1}(\hat{\mathbf{A}} \circ \text{NTT}(\mathbf{y}))$
  $\mathbf{w}_1 : R_{m_1}^{k} \leftarrow \text{HighBits}(\mathbf{w})$
   $m_1 \in [0, 21]$ *if* ML-DSA-44
   $m_1 \in [0, 15]$ *if* ML-DSA-65 *or* ML-DSA-87
  ▷ HighBits is applied componentwise
  $\tilde{c} : \mathbb{B}^{\lambda/4} \leftarrow \text{H}(\mu \,\|\, \text{w1Encode}(\mathbf{w}_1), \lambda/4)$ ▷ λ is the collission strength of $\tilde{c}$
   ▷ $\lambda/4 = 32$ (ML-DSA-44) or 48 (ML-DSA-65) or 64 (ML-DSA-87)
  $c : R_q \leftarrow \text{SampleInBall}(\tilde{c})$ ▷ polynomial in R with coefficients from {-1, 0, 1}
  $\hat{c} : T_q \leftarrow \text{NTT}(c)$
  $\langle\langle c\mathbf{s}_1 \rangle\rangle : R_q^{\ell} \leftarrow \text{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{s}}_1)$ ▷ $\hat{c}$ is multiplied with each of $\hat{\mathbf{s}}_1^{0}, \dots, \hat{\mathbf{s}}_1^{l-1}$
  $\langle\langle c\mathbf{s}_2 \rangle\rangle : R_q^{k} \leftarrow \text{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{s}}_2)$ ▷ $\hat{c}$ is multiplied with each of $\hat{\mathbf{s}}_2^{0}, \dots, \hat{\mathbf{s}}_2^{k-1}$
  $\mathbf{z} : R^{l}_{mod^{\pm}q} \leftarrow \mathbf{y} + \langle\langle c\mathbf{s}_1 \rangle\rangle$ ▷ $\mathbf{z} : R^{l}_{mod^{\pm}q} \leftarrow \mathbf{y} : R_q^{\ell} + \langle\langle c\mathbf{s}_1 \rangle\rangle : R_q^{\ell}$
  $\mathbf{r}_0 : R_q^{k} \leftarrow \text{LowBits}(\mathbf{w} - \langle\langle c\mathbf{s}_2 \rangle\rangle)$
  ▷ LowBits is applied componentwise
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **or** $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then** $(\mathbf{z}, \mathbf{h}) \leftarrow \bot$
  else
    $\langle\langle c\mathbf{t}_0 \rangle\rangle : T_q^{k} \leftarrow \text{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{t}}_0)$ ▷ $\hat{c}$ is multiplied with each of $\hat{\mathbf{t}}_0^{0}, \dots, \hat{\mathbf{t}}_0^{k-1}$
    $\mathbf{h} : R_2^{k} \leftarrow \text{MakeHint}(-\langle\langle c\mathbf{t}_0 \rangle\rangle, \mathbf{w} - \langle\langle c\mathbf{s}_2 \rangle\rangle + \langle\langle c\mathbf{t}_0 \rangle\rangle)$
    ▷ MakeHint is applied componentwise
    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** $(\mathbf{z}, \mathbf{h}) \leftarrow \bot$
    end if
  end if
  $\kappa \leftarrow \kappa + \ell$
end while
$\sigma \leftarrow \text{sigEncode}(\tilde{c}, \mathbf{z} \bmod^{\pm} q, \mathbf{h})$
return σ