leanXMSS for Ethereum PQ-Consensus, specification draft

Abstract

leanXMSS is a stateful hash-based signature scheme: two layers of XMSS [8] of height 16 over WOTS$^+$C one-time keys [10, 5]. Each leaf of the top tree signs the root of one bottom tree, so a key signs at 2^{32} leaf indices.

- **Target: NIST security level 1** [12], with 127 bits of classical security proved in Lean (Section 6).
- **Public key: 32 bytes.**
- **Signature: 2404 bytes** ($4 + 1200 + 1200$: the leaf index and the two layers).
- **Verification: 186 permutations** of Keccak-f.
- **Threshold friendly** (Section 8).
- **Grinding:** $\approx 2^{20}$ **randomizer trials per signature** on average, one permutation each.

1 Overview

A key is two layers of Merkle trees of height $h = 16$, whose leaves are WOTS$^+$C one-time keys (Figure 1). The top layer is one tree, and its root is the public key. The bottom layer is 2^{16} trees, and key b of the top tree signs the root of bottom tree b. A leaf index $idx < 2^{32}$ names bottom key idx, in bottom tree $b = \lfloor idx/2^{16} \rfloor$.

To sign a message m at idx, the signer signs m with bottom key idx and the root of bottom tree b with top key b, and attaches each key's authentication path. The verifier recovers the bottom key's leaf from m and the signature, follows its path to the root of bottom tree b, does the same on the top layer, and accepts if it reaches the public key. The scheme is stateful: a leaf index may sign only one message.

2 Hashing and parameters

Write $\|$ for concatenation, 0^r for r zero bits, $\mathsf{LE}_r(z)$ for the r-bit little-endian encoding of an integer z, and LE^{-1} for its inverse. Let H be Keccak-256 [2], Ethereum's hash, which is the 256-bit hash of FIPS 202 [11] but for its padding's first byte, `0x01` where the standard has `0x06`. Write $\mathsf{H}_0, \mathsf{H}_1$ for the first and last 16 bytes of its output. With a 16-byte public parameter P and a 16-byte address A (Section 7),

$$\mathsf{Th}(P, \mathsf{A}, M) = \mathsf{H}_0(P \,\|\, \mathsf{A} \,\|\, M).$$

Symbol	Value	Meaning
v	58	chains per one-time key, of positions 0 to 3
T	106	sum of a codeword's digits
h	16	height of each layer's trees
$A_{\max}$	2^{32}	randomizer trials per one-time signature

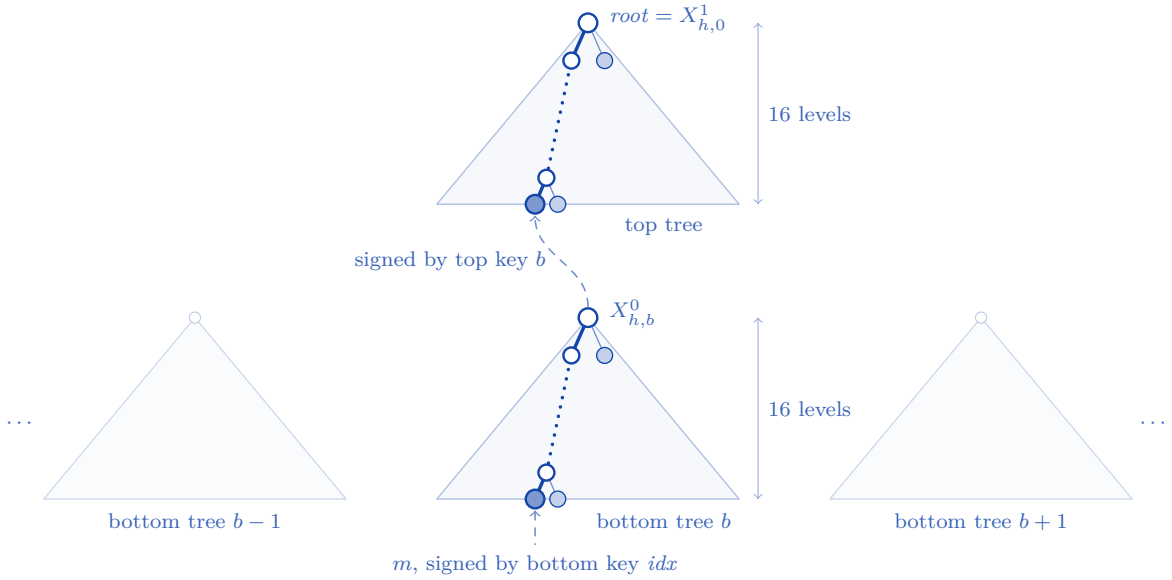


Figure 1: A signature at leaf index idx, in bottom tree $b = \lfloor idx/2^{16} \rfloor$, whose leaves are keys $b\,2^{16}$ to $(b+1)2^{16}-1$. On each layer, the verifier recomputes the signing key (dark) and its path to the root (bold) from the siblings the signature carries (shaded, one per level, the 14 middle levels dotted).

The secret key is a 32-byte seed S_0. Every secret derives from $S = \mathsf{H}(\texttt{leanXMSS} \parallel S_0)$, the hash of the scheme's 8-byte ASCII name then the seed, so that no other scheme derives from the same bytes, and $P = \mathsf{Th}(0^{128}, \mathsf{A}_{\text{param}}, S)$. A message m is 32 bytes. The layers are $\ell = 0$ (bottom) and $\ell = 1$ (top). A leaf index $idx < 2^{32}$ uses key idx at the bottom, which signs $M = m$, and key $b = \lfloor idx/2^h \rfloor$ at the top, which signs the 16-byte root $M = X^0_{h,b}$ of bottom tree b (Section 4).

3 WOTS$^+$C

A one-time key is v hash chains of 4 positions. It signs a 16-byte digest that is a codeword, revealing each chain at the position its digit names. The chains of key e on layer ℓ are, for $i < v$ and $\mu < 3$,

$$C_{i,0} = \mathsf{H}_{i \bmod 2}(P \parallel \mathsf{A}_{\text{prf}}(\ell, e, \lfloor i/2 \rfloor) \parallel S), \qquad C_{i,\mu+1} = \mathsf{Th}(P, \mathsf{A}_{\text{chain}}(\ell, e, \mu), C_{i,\mu} \parallel \mathsf{LE}_{128}(i)),$$

and its leaf is $X^\ell_{0,e} = \mathsf{Th}(P, \mathsf{A}_{\text{leaf}}(\ell, e), C_{0,3} \parallel \cdots \parallel C_{v-1,3})$.

Codewords. A 16-byte digest D has the digits $x_i = \lfloor \mathsf{LE}^{-1}(D)/4^i \rfloor \bmod 4$ for $i < 64$. It is a codeword when $x_v = \cdots = x_{63} = 0$ and $\sum_{i<v} x_i = T$. Distinct codewords are incomparable, each having a digit smaller than the other's, so the chain values revealed for one codeword never sign another. This replaces the Winternitz checksum.

Signing. $\mathsf{Ots.sign}(\ell, e, M)$ computes

$$R_0 = \mathsf{Th}(P, \mathsf{A}_{\text{rnd}}(\ell, e), S \parallel M), \qquad \rho_c = \mathsf{LE}_{128}(\mathsf{LE}^{-1}(R_0) + c \bmod 2^{128}),$$
$$D_c = \mathsf{Th}(P, \mathsf{A}_{\text{msg}}(\ell, e), M \parallel \rho_c),$$

takes the least $c < A_{\max}$ for which D_c is a codeword x, and returns $\rho = \rho_c$ and the chain values $y_i = C_{i,x_i}$, or $\perp$ if no $c < A_{\max}$ does. It is deterministic. The layout is chosen so that every trial

costs exactly one permutation: ρ_c is a sum rather than a hash, and the input is one block, 80 bytes on the bottom layer and 64 on the top.

Recovering the leaf. $\mathsf{Ots.leaf}(\ell, e, M, \rho, y)$ recomputes D, returns $\perp$ unless it is a codeword x, walks each y_i from position x_i to position 3, and hashes the ends into the leaf.

4 Trees

On layer ℓ, for $1 \le \lambda \le h$,

$$X^{\ell}_{\lambda,j} = \mathsf{Th}\big(P, \mathsf{A}_{\mathrm{node}}(\ell, \lambda, j), X^{\ell}_{\lambda-1,2j} \,\|\, X^{\ell}_{\lambda-1,2j+1}\big).$$

The top layer is one tree, with root $X^1_{h,0}$. The bottom layer is 2^h trees whose node indices are global: level λ has $2^{2h-\lambda}$ nodes, bottom tree b holds those with $\lfloor j/2^{h-\lambda} \rfloor = b$, and its root is $X^0_{h,b}$ (Figure 1).

The path of key e is $\pi = (\pi_0, \ldots, \pi_{h-1})$ with $\pi_\lambda = X^{\ell}_{\lambda,\lfloor e/2^\lambda \rfloor \oplus 1}$, where $\oplus 1$ flips the low bit. $\mathsf{Tree.fold}(\ell, e, X, \pi)$ starts from $V_0 = X$ and sets $V_{\lambda+1} = \mathsf{Th}(P, \mathsf{A}_{\mathrm{node}}(\ell, \lambda+1, \lfloor e/2^{\lambda+1} \rfloor), V_\lambda \,\|\, \pi_\lambda)$, the two children swapped when bit λ of e is one. It returns V_h.

5 The scheme

Key generation. Sample a uniform 32-byte S_0 and return $pk = (X^1_{h,0}, P)$ and $sk = S_0$.

Signing. $\mathsf{Sig}(sk, idx, m)$, with $b = \lfloor idx/2^h \rfloor$:

1. bottom layer: $(\rho^0, y^0) = \mathsf{Ots.sign}(0, idx, m)$, and the path π^0 of key idx;
2. top layer: $(\rho^1, y^1) = \mathsf{Ots.sign}(1, b, X^0_{h,b})$, and the path π^1 of key b;
3. return $\sigma = (idx, \rho^0, y^0, \pi^0, \rho^1, y^1, \pi^1)$, or $\perp$ if either $\mathsf{Ots.sign}$ failed.

A leaf index must never sign two different messages: their two codewords would reveal every chain at the smaller of their two positions. The top key b signs only $X^0_{h,b}$, so it signs one message, however many leaf indices of bottom tree b are used.

Serialization. The public key is $root \,\|\, P$. A layer's signature is $\rho \,\|\, y_0 \,\|\, \cdots \,\|\, y_{v-1} \,\|\, \pi_0 \,\|\, \cdots \,\|\, \pi_{h-1}$, $16(1 + v + h) = 1200$ bytes. The signature is $\mathsf{LE}_{32}(idx)$, the bottom layer's, then the top layer's: $4 + 1200 + 1200 = 2404$ bytes.

Verification. $\mathsf{Ver}((root, P), m, \sigma)$ rejects unless σ is 2404 bytes, parses it as idx, (ρ^0, y^0, π^0) and (ρ^1, y^1, π^1), and computes

$$X = \mathsf{Tree.fold}\big(0, idx, \mathsf{Ots.leaf}(0, idx, m, \rho^0, y^0), \pi^0\big),$$
$$X' = \mathsf{Tree.fold}\big(1, b, \mathsf{Ots.leaf}(1, b, X, \rho^1, y^1), \pi^1\big).$$

It accepts exactly when neither $\mathsf{Ots.leaf}$ fails and $X' = root$.

Verification cost. A k-byte input costs $\lfloor k/136 \rfloor + 1$ permutations of Keccak-f[1600]. Verification costs, per layer, 1 for the digest, $3v - T = 68$ for the chains, 8 for the leaf and 16 for the path: 93 per layer, 186 in all.

Signing cost. Building a tree costs $212 \cdot 2^h - 1$ permutations: per key, 29 for the chain starts, $3v = 174$ chain steps and 8 for the leaf, then $2^h - 1$ nodes. Key generation builds the top tree, and the signer builds each bottom tree when it first signs in it. A digest is a codeword with probability

$$p = [z^T](1 + z + z^2 + z^3)^v / 2^{128} \approx 2^{-20},$$

so an encoding takes about 2^{20} trials on average, one permutation each, and all $A_{\max}$ trials fail with probability below $\exp(-2^{11})$. A signature needs one encoding per layer, but the top layer's depends only on the bottom tree, so the signer grinds it once per bottom tree: amortized, a signature costs about 2^{20} trials.

6 Security

Model H as one shared random oracle. The challenger generates a key pair and gives pk to a classical probabilistic adversary, which may adaptively query the hash and request signatures $\mathsf{Sig}(sk, idx, m)$ on messages m at leaf indices idx of its choice, never the same leaf index twice. It finally returns (m^*, σ^*) and wins exactly when $\mathsf{Ver}(pk, m^*, \sigma^*) = 1$ and no signing request for m^* returned σ^*: *strong unforgeability*. A query budget q bounds every hash call of the complete experiment: key generation, signing, the adversary's queries and the final verification, repeated inputs included.

Theorem 1. *For every $q \geq 1$ and every adversary with budget q, the adversary wins with probability at most $q/2^{127}$.*

The theorem is proved in Lean 4 over VCVio [14], with only Lean's three standard axioms, in `formal/lean_xmss/` (`XmssSecurity.leanXmss_has_127_bits_of_classical_security`, of statement `LeanXmss.LeanXmssSecurityStatement`), for the scheme of this document byte for byte: its hash inputs, its codewords and its signer. `formal/lean_xmss/PROOF.md` outlines the route: once the seed is eliminated, a win implies a bad event on one of the two layers, a hash call that hits an honest output from another input or guesses a chain value not yet revealed, and each call raises one with probability at most 2^{-127}.

Using a key. The theorem covers a signer that uses each leaf index at most once, which the signer's state must enforce. Signing is deterministic, so signing the same message again at the same leaf index returns the same signature, but the game counts no run that repeats a leaf index as a win, so the theorem does not cover it. Two different messages at one leaf index break the key (Section 5).

Tightness. The following classical attack shows that the security is at most about 127.2 bits.

1. **Get a signature** at a leaf index idx, and read its bottom codeword x.
2. **Find a nearby codeword.** For a new message m^* at idx, try 2^{120} distinct randomizers, seeking a codeword that decreases one digit x_i by one and increases another digit x_j by one. The digits are at most 3 and sum to $T = 106$, so at least 36 are nonzero and at least 23 are below 3: there are at least $36 \cdot 22 = 792$ choices of (i, j), each one 128-bit digest, and

$$\Pr[\text{codeword found}] \geq 1 - \left(1 - \frac{792}{2^{128}}\right)^{2^{120}} \geq 1 - \exp\left(-\frac{792}{2^8}\right) \approx 0.955.$$

3. **Go back one chain step.** Try 2^{124} distinct values, seeking a preimage of y_i under the chain step from position $x_i - 1$ of chain i. They contain the genuine C_{i,x_i-1} with probability $1/16$; otherwise one of them hits y_i with probability about $1 - e^{-1/16}$. Thus

$$\Pr[\text{preimage found}] \approx \frac{1}{16} + \frac{15}{16}\left(1 - e^{-1/16}\right) \approx 0.1193.$$

4. **Forge.** Replace y_i by the preimage and y_j by its next chain step, keeping the rest of the signature.

Key generation and signing add a negligible number of calls, so

$$\frac{q}{\Pr[\text{success}]} \approx \frac{2^{120} + 2^{124}}{0.955 \cdot 0.1193} \approx 1.17 \cdot 2^{127} \approx 2^{127.2}.$$

The top layer admits the same attack, on the root of a bottom tree of the adversary's own, at the same cost.

Quantum security. Only the classical bound is proved. We expect about 64 quantum bits, NIST category 1, which FIPS 205 assigns to the SLH-DSA parameter sets with a 128-bit digest [13]. The hypertree of leanSPHINCS$^+$, of which these two layers are a variant, has a QROM bound against an adversary that does not choose the signed messages [9], and hash-then-sign reduces chosen messages to random ones [7]. This remains a conjecture: the first bound is proved for the WOTS$^+$ encoding rather than for a target sum, and the second reduction loses around 1.5 bits.

7 Address encoding

Addresses keep the hash inputs of a key distinct. An address is $\mathsf{enc}(\tau, \mu, hi, lo) = \mathsf{LE}_{32}(lo) \,\|\, \mathsf{LE}_{24}(hi) \,\|\, \mathsf{LE}_8(\tau + 32\mu) \,\|\, 0^{64}$, with type $\tau < 32$ and $\mu < 8$. The top layer sets bit 23 of hi: $hi_\ell(a) = a + 2^{23}\ell$.

$\mathsf{A}_{\mathrm{prf}}(\ell, e, j)$	$\mathsf{enc}(0, 0, hi_\ell(j), e)$	$\mathsf{A}_{\mathrm{param}}$	$\mathsf{enc}(5, 0, 0, 0)$
$\mathsf{A}_{\mathrm{chain}}(\ell, e, \mu)$	$\mathsf{enc}(1, \mu, hi_\ell(0), e)$	$\mathsf{A}_{\mathrm{rnd}}(\ell, e)$	$\mathsf{enc}(7, 0, hi_\ell(0), e)$
$\mathsf{A}_{\mathrm{leaf}}(\ell, e)$	$\mathsf{enc}(2, 0, hi_\ell(0), e)$	$\mathsf{A}_{\mathrm{msg}}(\ell, e)$	$\mathsf{enc}(12, 0, hi_\ell(0), e)$
$\mathsf{A}_{\mathrm{node}}(\ell, \lambda, j)$	$\mathsf{enc}(3, 0, hi_\ell(\lambda), j)$		

8 Threshold signing

A key can be held by n operators, any t of whom can sign. This section sketches one way to do it, after PRAWNS [3]. It is an example, not part of the scheme.

Idea. The bottom keys derive their secrets from a context-aware threshold PRF F [3]: any t operators can evaluate it, and $t - 1$ cannot predict its outputs. Each chain starts at an output of F, and the setup publishes the chain's two middle values, each masked with an output of F, and its end. A WOTS$^+$C signature reveals one value per chain, which the signers obtain by evaluating F once, or read if it is the end, with no MPC. Signing takes two rounds: one for the randomizer base R_0, derived from an output of F, and one for the chain values. The top keys only sign bottom roots, so the setup signs them.

Setup. Either a trusted dealer computes everything in the clear for a lifetime of L bottom trees, or the operators run MPC [4, 6] at key generation and once per bottom tree.

Costs. The public data is 48 bytes per chain: 182 MB per bottom tree, or 7.3 GB per year of 12-second slots. With MPC, a bottom tree costs its $2^h \cdot 3v \approx 1.1 \cdot 10^7$ chain steps. Each is one Keccak-f[1600] permutation: 24 rounds, each with one AND gate per state bit in χ, 38400 in all, so a bottom tree takes $4.4 \cdot 10^{11}$ AND gates. At one bit sent per AND gate [1], each operator sends about 55 GB per bottom tree, or 70 KB/s with 12-second slots.

References

[1] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[2] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference, version 3.0, 2011.

[3] Dan Boneh, Benedikt Bünz, Justin D., Yavor Litchev, and Kamilla Nazirkhanova. PRAWNS: Threshold hash-based signatures from threshold PRFs. Cryptology ePrint Archive, Paper 2026/2142, 2026.

[4] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *Theory of Cryptography Conference (TCC)*, 2005.

[5] Justin Drake, Dmitry Khovratovich, Mikhail Kudinov, and Benedikt Wagner. Hash-based multi-signatures for post-quantum Ethereum. *IACR Communications in Cryptology*, 2(1):13, 2025.

[6] Vipul Goyal and Yifan Song. Malicious security comes free in honest-majority MPC. In *Advances in Cryptology: CRYPTO 2020*, 2020.

[7] Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the QROM. In *Advances in Cryptology: ASIACRYPT 2021, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 637–667, 2021.

[8] Andreas Hülsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: eXtended Merkle Signature Scheme. Technical Report RFC 8391, Internet Research Task Force, 2018.

[9] Andreas Hülsing and Mikhail Kudinov. Recovering the tight security proof of SPHINCS+. In *Advances in Cryptology: ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 3–33, 2022.

[10] Andreas Hülsing, Mikhail Kudinov, Eyal Ronen, and Eylon Yogev. SPHINCS+C: Compressing SPHINCS+ with (almost) no cost. In *IEEE Symposium on Security and Privacy*, 2023.

[11] National Institute of Standards and Technology. SHA-3 standard: Permutation-based hash and extendable-output functions. Federal Information Processing Standards Publication 202, U.S. Department of Commerce, 2015.

[12] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. NIST PQC Call for Proposals, 2016. Section 4.A.5, p. 16.

[13] National Institute of Standards and Technology. Stateless hash-based digital signature standard. Technical Report FIPS 205, National Institute of Standards and Technology, 2024.

[14] Devon Tuma, Quang Dao, James Waters, Alexander Hicks, and Nicholas Hopper. VCVio: Verified cryptography in lean via oracle effects and handlers. Cryptology ePrint Archive, Paper 2026/899, 2026.