XMSS for Ethereum PQ-Consensus, specification draft.

Abstract

As proposed in [DKKW25], XMSS is a promising post-quantum replacement for BLS [BLS01], the signature scheme used by Ethereum validators. The lack of native aggregation can be mitigated with a post-quantum snark. This technical note specifies an XMSS instance with the following properties:

- **NIST security level 1** [Nat16], ≈ 128-bit (resp. ≈ 64-bit) of classical (resp. quantum) security, in the Random Oracle model, ROM (resp. Quantum Random Oracle model, QROM).[1]
- **public key: 32 bytes**.
- **signature: 1208 bytes** (smaller than the 1280-byte minimum IPv6 MTU [DH17]).
- snark-friendly encoding, optimized for aggregation via leanVM [lea26], requiring 15 bits of grinding on average.
- 133 hashes per verification.

The key difference with [DKKW25] is the use of short hash digests, of 16 bytes, following SLH-DSA [Nat24] at level 1. The consequence is that the security proof takes place in the ROM, while [DKKW25] is secure in the Standard Model (as a cost of bigger hash digests, close to 32 bytes).

1 Definitions and notation

Definition 1.1 (Synchronized signature scheme)**.** A synchronized signature scheme with lifetime L is a tuple $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$, where Gen and Sig are randomized and Ver is deterministic:

$$\mathsf{Gen} \longrightarrow (pk, sk), \qquad \mathsf{Sig}(sk, ep, m) \longrightarrow \sigma \in \Sigma \cup \{\bot\}, \qquad \mathsf{Ver}(pk, ep, m, \sigma) \longrightarrow \{0,1\},$$

where Σ is the signature space, $m \in \{0,1\}^{\ell_{\text{msg}}}$, and $ep \in \{0, \dots, L-1\}$. Whenever (pk, sk) is output by Gen and $\mathsf{Sig}(sk, ep, m)$ returns $\sigma \neq \bot$, correctness requires $\mathsf{Ver}(pk, ep, m, \sigma) = 1$. For each key pair, $\mathsf{Sig}(sk, ep, \cdot)$ may be invoked at most once for any fixed epoch ep.

XMSS is a synchronized signature scheme in which epoch ep selects a Winternitz one-time key authenticated by leaf ep of a Merkle tree [HBG+18]. This specification sets $\ell_{\text{msg}} = 256$ and $L = 2^h = 2^{32}$.

Byte strings are concatenated with $\|$. Bits and integer encodings are little endian. $\mathsf{LE}_r(a)$ is the unsigned r-bit encoding of a. All indices are zero based.

[1] The classical bound is proved (Section 5); the quantum one is a target, not a proved statement (Section 5.2).

Symbol	Value	Meaning
n	128 bits	hash value and Merkle node length
ℓ_{p}	128 bits	public parameter length
ℓ_{t}	128 bits	tweak length
ℓ_{msg}	256 bits	message length
ℓ_{rnd}	192 bits	encoding randomness length
w	3	chunk size in bits
v	42	code length
T	195	target sum
h	32	Merkle tree height
L	2^{32}	lifetime
$A_{\max}$	2^{23}	maximum encoding attempts per signature

Let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{256}$ be a cryptographic hash function, and let $\mathsf{Truncate}_{\nu \text{ bits}}$ keep the first ν output bits. The tweakable hash $\mathsf{Th} : \mathcal{P} \times \mathcal{T} \times \mathcal{M} \to \mathcal{H}$, with $\mathcal{P} = \{0,1\}^{\ell_{\mathrm{p}}}$, $\mathcal{T} = \{0,1\}^{\ell_{\mathrm{t}}}$ and $\mathcal{H} = \{0,1\}^n$, is

$$\mathsf{Th}(P, tw, M) = \mathsf{Truncate}_{n \text{ bits}}(\mathsf{H}(tw \parallel P \parallel M)) .$$

Definition 1.2 (Tweak encoding)**.** For a one-byte type t and unsigned 32-bit integers p and j, define the 16-byte tweak

$$\mathsf{enc}(t, p, j) = \mathsf{LE}_8(t) \parallel \mathsf{LE}_{32}(p) \parallel \mathsf{LE}_{32}(j) \parallel 0^{56}.$$

Define

$$\begin{aligned}
\mathsf{tweak}_{\mathrm{chain}}(ep, i, k) &= \mathsf{enc}(0, 2^w i + k - 1, ep), && 0 \le i < v,\ 1 \le k \le 2^w - 1,\\
\mathsf{tweak}_{\mathrm{leaf}}(ep) &= \mathsf{enc}(1, 0, ep), &&\\
\mathsf{tweak}_{\mathrm{merkle}}(\ell, j) &= \mathsf{enc}(2, \ell, j), && 1 \le \ell \le h,\ 0 \le j < 2^{h-\ell},\\
\mathsf{tweak}_{\mathrm{enc}}(ep) &= \mathsf{enc}(3, 0, ep). &&
\end{aligned}$$

The target-sum code is [DKKW25]

$$\mathcal{C} = \left\{ (x_0, \ldots, x_{v-1}) \in \{0, \ldots, 2^w - 1\}^v : \sum_{i=0}^{v-1} x_i = T \right\}.$$

Define $\mathsf{IncEnc}(P, m, \rho, ep)$ by computing

$$D = \mathsf{Th}\Big(P, \mathsf{tweak}_{\mathrm{enc}}(ep), m \parallel \rho \parallel 0^{64}\Big) .$$

Write $D = D_0 \parallel D_1$, where D_0 and D_1 are $n/2$-bit strings. For $q \in \{0,1\}$, let $d_q \in \{0, \ldots, 2^{n/2} - 1\}$ be the integer whose $n/2$-bit little-endian encoding is D_q. For $0 \le r < v/2$, set

$$x_{qv/2+r} = \left\lfloor \frac{d_q}{2^{wr}} \right\rfloor \bmod 2^w.$$

Return $x = (x_0, \ldots, x_{v-1})$ if bit $wv/2 = 63$ of both d_0 and d_1 is zero and $x \in \mathcal{C}$; otherwise return $\perp$.

Definition 1.3 (Hash chain)**.** For a chain index i, epoch ep, start index k, number of steps s, and value $x \in \mathcal{H}$, define $\mathsf{Chain}_{\mathsf{Th},i,ep}(P,k,s,x)$ recursively by

$$\mathsf{Chain}_{\mathsf{Th},i,ep}(P,k,0,x) = x$$

and, for $s \geq 1$,

$$\mathsf{Chain}_{\mathsf{Th},i,ep}(P,k,s,x) = \mathsf{Th}(P, \mathsf{tweak}_{\mathrm{chain}}(ep,i,k+s), \mathsf{Chain}_{\mathsf{Th},i,ep}(P,k,s-1,x))\,.$$

The valid range is $0 \leq k \leq 2^w - 1$ and $0 \leq s \leq 2^w - 1 - k$.

The one-time public key and its Merkle leaf are

$$\begin{aligned} pk_{ep,i} &= \mathsf{Chain}_{\mathsf{Th},i,ep}(P,0,2^w-1,sk_{ep,i}), && 0 \leq i < v, \\ X_{0,ep} &= \mathsf{Th}\Big(P, \mathsf{tweak}_{\mathrm{leaf}}(ep), pk_{ep,0} \parallel \cdots \parallel pk_{ep,v-1}\Big)\,. \end{aligned}$$

2 Key generation

The algorithm Gen independently samples $P \xleftarrow{\$} \{0,1\}^{\ell_{\mathrm{P}}}$ and $sk_{ep,i} \xleftarrow{\$} \mathcal{H}$ for $0 \leq ep < L$ and $0 \leq i < v$. It computes and stores every chain value

$$C_{ep,i,k} = \mathsf{Chain}_{\mathsf{Th},i,ep}(P,0,k,sk_{ep,i}), \qquad 0 \leq k < 2^w.$$

Thus $C_{ep,i,0} = sk_{ep,i}$ and $C_{ep,i,2^w-1} = pk_{ep,i}$. For $1 \leq \ell \leq h$, define

$$X_{\ell,j} = \mathsf{Th}(P, \mathsf{tweak}_{\mathrm{merkle}}(\ell,j), X_{\ell-1,2j} \parallel X_{\ell-1,2j+1})\,, \qquad 0 \leq j < 2^{h-\ell}.$$

Set $root = X_{h,0}$ and return

$$pk = (root, P), \qquad sk = \left(P, (C_{ep,i,k})_{\substack{0 \leq ep < L,\, 0 \leq i < v, \\ 0 \leq k < 2^w}}, (X_{\ell,j})_{\substack{0 \leq \ell \leq h, \\ 0 \leq j < 2^{h-\ell}}}\right).$$

The public key is serialized as $root \parallel P$.

Remark 2.1 (Ideal precomputed secret key). This specification deliberately uses an ideal precomputed secret key. It stores the full Merkle tree and all intermediate one-time-signature chain values so that signing does not recompute them. Repeated access to such a value is a table read, not a fresh random-oracle query. An implementation may store or recompute those values as it sees fit, provided the sampled ones keep the distribution above.

Remark 2.2 (Seed derivation). The sampled secret values may instead be generated from a master secret using a pseudorandom function. For simplicity the theorem of Section 5 does not cover that case. One could for instance reuse H itself, under a fresh tweak type, and we expect the security proof to adapt easily.

Remark 2.3 (Restricted key generation). Key generation may be restricted to a smaller interval of epochs by replacing subtrees outside that interval with random nodes. The resulting key signs only epochs in that interval. The theorem of Section 5 does not cover this variant either, but we also expect the security proof to adapt easily.

3 Signing

$\mathsf{Sig}(sk, ep, m)$ returns $\perp$ if epoch ep has already been used and otherwise marks it used. It makes at most $A_{\max} = 2^{23}$ attempts. In each attempt it independently samples $\rho \xleftarrow{\$} \{0,1\}^{\ell_{\mathrm{rnd}}}$ and computes $x = \mathsf{IncEnc}(P, m, \rho, ep)$. If $x \neq \perp$, it immediately computes

$$\sigma_{\mathrm{OTS},i} = C_{ep,i,x_i}, \qquad 0 \le i < v,$$

and the authentication path

$$A_\ell = X_{\ell,\, \lfloor ep/2^\ell \rfloor \mathsf{xor} 1}, \qquad 0 \le \ell < h.$$

and returns

$$\sigma = (\rho, \sigma_{\mathrm{OTS}}, path_{ep}), \qquad path_{ep} = (A_0, \ldots, A_{h-1}),$$

serialized as $\sigma_{\mathrm{OTS},0} \,\|\, \cdots \,\|\, \sigma_{\mathrm{OTS},v-1} \,\|\, \rho \,\|\, A_0 \,\|\, \cdots \,\|\, A_{h-1}$. If all $A_{\max}$ attempts produce $x = \perp$, it returns $\perp$.

Consequently, signing queries the random oracle only when evaluating IncEnc, once per attempt. Reading the stored chain values and authentication path makes no random-oracle query.

4 Verification

$\mathsf{Ver}(pk, ep, m, \perp) = 0$. For $\sigma \neq \perp$, $\mathsf{Ver}(pk, ep, m, \sigma)$ parses $pk = (root, P)$ and $\sigma = (\rho, (y_i)_i, (A_\ell)_\ell)$, then:

1. Compute $x = \mathsf{IncEnc}(P, m, \rho, ep)$ and reject if $x = \perp$.
2. For every $0 \le i < v$, compute $pk'_{ep,i} = \mathsf{Chain}_{\mathsf{Th},i,ep}(P, x_i, 2^w - 1 - x_i, y_i)$.
3. Set $Z_0 = \mathsf{Th}(P, \mathsf{tweak}_{\mathrm{leaf}}(ep), pk'_{ep,0} \,\|\, \cdots \,\|\, pk'_{ep,v-1})$.
4. For $0 \le \ell < h$, let $j = \lfloor ep/2^{\ell+1} \rfloor$. If bit ℓ of ep is zero, set $Z_{\ell+1} = \mathsf{Th}(P, \mathsf{tweak}_{\mathrm{merkle}}(\ell + 1, j), Z_\ell \,\|\, A_\ell)$; otherwise set $Z_{\ell+1} = \mathsf{Th}(P, \mathsf{tweak}_{\mathrm{merkle}}(\ell + 1, j), A_\ell \,\|\, Z_\ell)$.
5. Accept if and only if $Z_h = root$.

5 Security

5.1 Classical security

Definition 5.1 (strong unforgeability in the ROM)**.** Let $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ be a synchronized signature scheme whose algorithms use a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{256}$. Consider the following game between a signer and an adversary $\mathcal{A}$ (an arbitrary probabilistic algorithm with unbounded running time and memory), with H sampled as a random oracle, that both signer and adversary can query. The signer first runs $(pk, sk) \leftarrow \mathsf{Gen}$ and gives pk to $\mathcal{A}$. The adversary may then adaptively take any of the following actions:

1. Query the random oracle on any input and receive its 256-bit output.
2. Submit a message $m \in \{0,1\}^{256}$ and an epoch $ep \in \{0, \ldots, L-1\}$ that it has not used in an earlier signing query, and receive $\sigma \leftarrow \mathsf{Sig}(sk, ep, m)$ from the signer.

3. Terminate with a claimed forgery (m^*, ep^*, σ^*).

The adversary wins if $\mathsf{Ver}(pk, ep^*, m^*, \sigma^*) = 1$ and the signer did not return σ^* in response to a signing query for (m^*, ep^*), meaning:

- if the adversary never queried a signature for (m^*, ep^*);
- or it did, but the signer's answer was not σ^*.

Call $\mathcal{A}$ q-bounded if the experiment makes at most q random-oracle queries on every execution, counting those of key generation, signing, and the final verification of the claimed forgery. We say that SIG has x bits of classical strong unforgeability in the ROM if every $q \geq 1$ and every q-bounded $\mathcal{A}$ satisfy

$$\Pr[\mathcal{A} \text{ wins}] \leq \frac{q}{2^x}.$$

Theorem 5.2 (XMSS security)**.** *The XMSS specified above has 127 bits of classical strong unforgeability in the ROM.*

A formal Lean4 proof of this theorem, using the VCVio framework [TDW+26], can be found in `./formal/xmss/`. The claim is stated in `XmssSecurity/Statement.lean`, discharged at the root in `XmssSecurity.lean`, and derived under `XmssSecurity/Proof/`. The theorem depends only on Lean's three standard axioms.

Proof sketch. TODO □

Remark 5.3 (Why not 128 bits?). The following classical attack demonstrates that security $\leq 127,3$ bits. The same idea also applies to SLH-DSA.

1. **Get a signature.** Ask for a signature and read its encoding x.
2. **Find a nearby encoding.** For a new message at the same epoch, try 2^{121} distinct randomness values. Seek an encoding that decreases one digit x_i by one and increases another digit x_j by one, leaving the rest unchanged. The sum stays 195. At least 28 digits are nonzero, and at least 15 are below 7, so there are at least $28 \cdot 14 = 392$ choices of (i, j) with $i \neq j$. Each choice specifies one 128-bit digest. A trial therefore succeeds with probability at least $392/2^{128}$, and independent trials give
$$\Pr[\text{encoding found}] \geq 1 - \left(1 - \frac{392}{2^{128}}\right)^{2^{121}} \geq 1 - \exp\left(-\frac{392 \cdot 2^{121}}{2^{128}}\right) \approx 0.953.$$
3. **Go back one chain step.** For the selected chain i, try 2^{124} distinct preimages of its signature value under the preceding chain step. A random guess has two ways to succeed: guess the genuine predecessor, or find another input with the same hash, each contributing about 2^{-128}. The guesses contain the genuine predecessor with probability $2^{124}/2^{128} = 1/16$. Otherwise, they find a collision 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.** Use the preimage on chain i and hash forward once on chain j. Reuse the other signature values and the authentication path.

The total success probability is at least about $0.953 \cdot 0.1193 \approx 0.114$, so

$$\frac{q}{\Pr[\text{success}]} \lesssim \frac{2^{121} + 2^{124}}{0.953 \cdot 0.1193} \approx 1.24 \cdot 2^{127} \approx 2^{127.3}.$$

5.2 Quantum security

Only the classical bound of Section 5 is proved. We expect ≈ 64 quantum bits, NIST category 1, which FIPS 205 assigns to the SLH-DSA parameter sets with a 128-bit digest [Nat24]. The XMSS layers of SPHINCS+ have a QROM bound against an adversary that does not choose the signed messages [HK22], and hash-then-sign reduces chosen messages to random ones [GHHM21]. This remains a conjecture: the first bound is proved for the WOTS+ encoding rather than for a target sum, and the second reduction induces a small loss of around 1.5 bits.

6 Completeness

Definition 6.1 (Completeness)**.** With H modeled as a random oracle, the completeness error of SIG is the worst-case probability that the signer fails,

$$\varepsilon_{\mathsf{SIG}} = \max_{ep,\,m} \Pr[\sigma = \perp \mid (pk, sk) \leftarrow \mathsf{Gen},\ \sigma \leftarrow \mathsf{Sig}(sk, ep, m)]\,,$$

over the oracle and the randomness of Gen and Sig. The scheme is ε-complete if $\varepsilon_{\mathsf{SIG}} \le \varepsilon$.

Theorem 6.2 (XMSS completeness)**.** *The XMSS specified above is 2^{-256}-complete in the ROM. Consequently a single key answers all $L = 2^{32}$ of its epochs with a signature, except with probability at most 2^{-224}.*

Proof. Fix ep and m. As ep is unused, $\mathsf{Sig}(sk, ep, m)$ returns $\perp$ only by exhausting its $A_{\max}$ attempts. Each attempt makes one oracle query,

$$I(\rho) = \mathsf{tweak}_{\mathrm{enc}}(ep) \parallel P \parallel m \parallel \rho \parallel 0^{64}, \qquad D = \mathsf{Truncate}_{n\ \mathrm{bits}}(\mathsf{H}(I(\rho)))\,,$$

and succeeds or not according to D alone: it succeeds exactly when $D \in \mathcal{D}$, where

$$\mathcal{D} = \{D \in \{0,1\}^n : \text{bit 63 of } d_0 \text{ and of } d_1 \text{ is zero, and } x \in \mathcal{C}\}$$

is a fixed subset of $\{0,1\}^n$, reading d_0, d_1 and x off D as in IncEnc. Put $\alpha = |\mathcal{D}|/2^n$.

The count. As $wv + 2 = n$, the digits of x and the two padding bits partition the bit positions of D, so zeroing the padding bits puts $\mathcal{D}$ in bijection with the $x \in \{0, \dots, 2^w - 1\}^v$ summing to T. Counting those is classical [Com74]:

$$|\mathcal{D}| = \sum_{b=0}^{\lfloor T/2^w \rfloor} (-1)^b \binom{v}{b} \binom{T - b2^w + v - 1}{v - 1}.$$

At $w = 3$, $v = 42$ and $T = 195$,

$$|\mathcal{D}| = 1153918537723868278134400324454475 2, \qquad \alpha = 2^{-14.8479\ldots} \ge 2^{-15} + 2^{-128}.$$

One attempt. The map I is injective, and no query of Gen reaches its image, Gen using tweak types 0, 1 and 2, not 3. So a ρ not drawn before is a fresh oracle point, making D uniform on $\{0,1\}^n$ and independent of the history, and the attempt succeeds with probability exactly α.

Exhausting the attempts. Let $\mathcal{F}_a$ be the event that the first a attempts all fail. The signer then samples a new ρ, uniformly at random and independently of the earlier attempts. This ρ equals one of the a earlier ones with probability at most $a2^{-\ell_{\mathrm{rnd}}}$, and otherwise the attempt fails with

probability $1-\alpha$, so $\Pr[\mathcal{F}_{a+1} \mid \mathcal{F}_a] \le 1-\alpha+a2^{-\ell_{\mathrm{rnd}}}$. Multiplying over $0 \le a < A_{\max}$, and since $A_{\max}2^{-\ell_{\mathrm{rnd}}} = 2^{-169} \le 2^{-128}$,

$$\Pr[\sigma=\perp] = \Pr[\mathcal{F}_{A_{\max}}] \le \left(1-\alpha+A_{\max}2^{-\ell_{\mathrm{rnd}}}\right)^{A_{\max}} \le \left(1-2^{-15}\right)^{A_{\max}}.$$

As $\left(1-2^{-15}\right)^{2^{15}} \le 2^{-1}$ and $A_{\max} = 2^{15}\cdot 2^8$, this gives $\varepsilon_{\mathsf{SIG}} \le 2^{-2^8} = 2^{-256}$. Finally $\mathsf{tweak}_{\mathrm{enc}}(ep)$ carries the epoch, so calls at distinct epochs query disjoint oracle points and the bound holds for each of a key's L epochs; a union bound over them gives the second claim. □

References

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 514–532, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[Com74] Louis Comtet. *Advanced Combinatorics: The Art of Finite and Infinite Expansions*. D. Reidel Publishing Company, Dordrecht, 1974.

[DH17] Stephen E. Deering and Robert M. Hinden. Internet protocol, version 6 (IPv6) specification. Technical Report RFC 8200, Internet Engineering Task Force, 2017. URL: https://www.rfc-editor.org/rfc/rfc8200, doi:10.17487/RFC8200.

[DKKW25] 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. URL: https://eprint.iacr.org/2025/055, doi:10.62056/aey7qjp10.

[GHHM21] 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. URL: https://eprint.iacr.org/2020/1361, doi:10.1007/978-3-030-92062-3_22.

[HBG+18] 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. URL: https://www.rfc-editor.org/rfc/rfc8391, doi:10.17487/RFC8391.

[HK22] 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. URL: https://eprint.iacr.org/2022/346, doi:10.1007/978-3-031-22972-5_1.

[lea26] leanEthereum. leanvm. Software and technical documentation, 2026. URL: https://github.com/leanEthereum/leanVM.

[Nat16] 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.

[Nat24] National Institute of Standards and Technology. Stateless hash-based digital signature standard. Technical Report FIPS 205, National Institute of Standards and Technology, 2024. doi:10.6028/NIST.FIPS.205.

[TDW+26] 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. URL: https://eprint.iacr.org/2026/899.