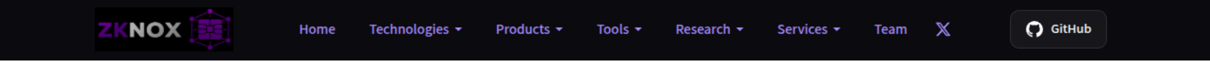


Post-Quantum Transaction Signatures

PQTS Breakout Room — ZKNOX contributions

Renaud Dubois, Simon Masson, Nicolas Bacca

Slides: github.com/ZKNoxHQ/Communications/pqts-breakout

February 4, 2026

What We Have Today — Full Operational Suite with Dapp Integration

Precompile EIPs (deployed on testnet):

- EIP-8051: ML-DSA (Dilithium) precompile
- EIP-8052: FN-DSA (Falcon) precompile
- Ethereum-optimized variants (Keccak PRNG)

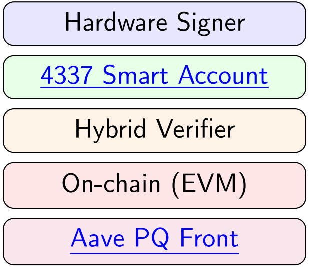
Signer implementations:

- ML-DSA on Ledger hardware wallet
- Falcon software signer
- PQ-BIP39 key derivation (zkProof of seed)

Hybrid & agile verifier (ERC-4337):

- ECDSA (k1/r1) + ML-DSA/FN-DSA
- ETH-optimized or NIST-native
- Modular, swappable verifiers

Full stack:



Hardware Signer

4337 Smart Account

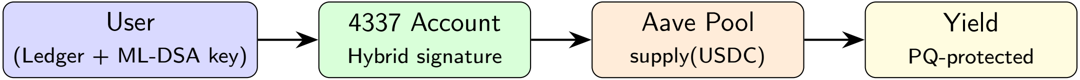
Hybrid Verifier

On-chain (EVM)

Aave PQ Front

Protecting High-Value Use Cases **Today**

Key insight: We don’t need precompiles to start protecting assets *now*.
Pure Solidity verification works — and at current gas prices, it’s affordable.



User
(Ledger + ML-DSA key)

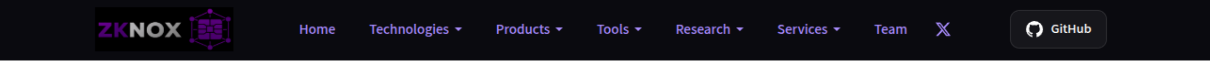
4337 Account
Hybrid signature

Aave Pool
supply(USDC)

Yield
PQ-protected

Scheme	Gas (Solidity)	Cost @0.55 gwei, ETH=$2322
ML-DSA (NIST)	13.0M gas	$16.60
ML-DSA-ETH	8.3M gas	$10.60
Falcon (NIST)	4.1M gas	$5.24
Falcon-ETH	1.6M gas	**$2.04**

Use cases (no precompile needed): governance contracts, treasury management, DeFi yield positions (Aave, Compound), multisig upgrades — high value, low frequency.



Why We Need Precompiles

With precompiles, PQ signature verification cost becomes comparable to ECDSA. The dominant cost of a PQ transaction is then the **UserOp handling itself**, not the cryptography.

NIST candidates (pros and cons)

Standardization since 2016... and ~~the winner is~~ the winners are:

- **Dilithium – ML-DSA**, based on lattices,
- **Falcon – FN-DSA**, based on lattices,
- SPHINCS – SLH-DSA, based on hash functions (big and expensive)

How to choose?

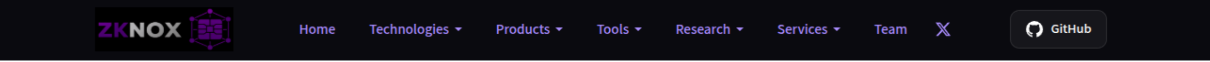
NIST candidates (pros and cons)

Standardization since 2016... and ~~the winner is~~ the winners are:

- **Dilithium – ML-DSA**, based on lattices,
- **Falcon – FN-DSA**, based on lattices,
- SPHINCS – SLH-DSA, based on hash functions (big and expensive)

How to choose?

	ML-DSA	**FN-DSA**
EIP	8051	8052
Public key	1312B	897B
Signature	2420B	666B
On-chain cost	13.0M gas (8.3M gas)	4.1M gas (1.6M gas)
(with Precompile)	4500 gas	3000 gas
Standardized?	FIPS 204	not yet (since 2 years)
Signer implementation	Easy, many	Tricky, floating point
Hardware integration	Done	High RAM requirements
Industrial integration	Passkey, Apple (soon)	Luna HSM (no memory constraint)
ZK variant	Possible	Overstretch attacks



EIPs 8051 and 8052

- **EIP 8051**: link
  - Two precompiles:
    - MLDSA: NIST-compliant with SHAKE256 (verification: 13.0 M gas, not far from the tx limit of 16M!).
    - MLDSA-ETH: replacement with a counter-mode Keccak PRNG (verification: 8.3M gas).
  - Test vector provided (generated from NIST reference implementation).
  - Integrated into a 4337 hybrid (MLDSA + ECDSA) account:

EIPs 8051 and 8052

- **EIP 8051**: link
  - Two precompiles:
    - MLDSA: NIST-compliant with SHAKE256 (verification: 13.0 M gas, not far from the tx limit of 16M!).
    - MLDSA-ETH: replacement with a counter-mode Keccak PRNG (verification: 8.3M gas).
  - Test vector provided (generated from NIST reference implementation).
  - Integrated into a 4337 hybrid (MLDSA + ECDSA) account:
- **EIP 8052**: link
  - Separation of the hash part and the polynomial arithmetic:
    - FALCON: NIST-compliant with SHAKE256 (verification: 4.1M gas).
    - ETH-FALCON: replacement with a counter-mode Keccak PRNG (verification: 1.6M gas).
  - Precompiles for FALCON-CORE and HASH-TO-POINT (one for Shake256, one for KeccakPRNG).
  - Test vector provided (generated from NIST reference implementation).
  - Integration in a 4337 account in progress...

Live Demo: Post-Quantum DeFi on Sepolia

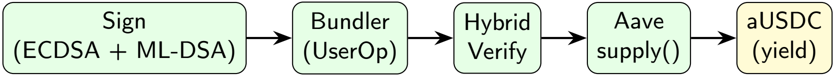
Demo: visionary-nougat-217eaa.netlify.app

What it does:

1. Connect with PQ-enabled signer
2. Hybrid signature (ECDSA + ML-DSA)
3. Supply USDC to Aave V3 (Sepolia)
4. Earn yield with quantum-safe keys

Stack:

- ERC-4337 smart account
- Modular hybrid verifier
- Pure Solidity PQ verification
- Standard Aave V3 interaction
- No protocol modification needed


Sign (ECDSA + ML-DSA)
Bundler (UserOp)
Hybrid Verify
Aave supply()
aUSDC (yield)