hashsigs-solidity

audited.xyz
by zack.eth

Date
Apr 08, 2026 09:52 UTC

Commit
e542347
7 files, 877 lines

Summary

The hashsigs-solidity project implements the Winternitz One-Time Signature Plus (WOTS+) scheme as a Solidity library, providing key generation, signing, and verification functions alongside supporting cryptographic primitives including hash chain computation, a pseudorandom function, base-w conversion, and checksum calculation. The library follows the XMSS RFC 8391 specification with keccak256 as its hash function and a Winternitz parameter of w=16, yielding correctly derived constants: 64 message chunks, 3 checksum chunks, and a chain length of 16. All functions are declared `pure`, meaning the library carries no state and cannot directly interact with on-chain assets, a deliberate architectural choice that reflects its role as a cryptographic building block rather than a standalone contract.

The security mechanisms employed in this codebase are well-considered for a cryptographic library. Domain separation in the PRF uses the `0x03` prefix as specified by RFC 8391, randomization elements are deterministically derived from the public seed, and the base-w encoding with checksum computation correctly ensures signature integrity. The cryptographic flow — secret key segments hashed through chains to produce public key segments, which are then hashed together into the final public key — faithfully implements the WOTS+ construction. The `sign` and `generateKeyPair` functions are explicitly documented as reference implementations not intended for on-chain use, which is appropriate given that private key material passed via calldata would be publicly visible on the blockchain.

The overall security posture of this library is strong. The pure library pattern with no state variables and no external calls eliminates entire classes of smart contract vulnerabilities, including reentrancy, access control bypasses, storage manipulation, and oracle dependencies. The test suite covers key generation, signing, verification, and cross-validation against deterministic test vectors, providing substantial confidence in the correctness of the cryptographic implementation. No vulnerabilities that could lead to fund loss, signature forgery, or verification bypass were identified during this audit.

Conclusion

The hashsigs-solidity codebase demonstrates a disciplined and correct implementation of the WOTS+ cryptographic scheme. Parameter derivation from the security parameter n=32 is accurate, domain separation in the pseudorandom function adheres to the RFC specification, and the base-w encoding with checksum computation is faithfully implemented. The decision to expose all cryptographic operations as `pure` functions within a stateless library reflects sound architectural judgment, as it constrains the attack surface to input validation and algorithmic correctness rather than the broader state management concerns typical of Solidity contracts.

The security posture of this library benefits substantially from its minimal design surface. By avoiding state, external calls, and token interactions, the codebase sidesteps the vulnerability classes that account for the majority of losses in smart contract exploits. Areas for continued vigilance include ensuring that integrating contracts understand the one-time nature of WOTS+ signatures and that private key material is never exposed on-chain — constraints that are well-documented in the existing codebase but that downstream consumers must respect.

The library is suitable for deployment as a cryptographic component within larger systems. Integrators should use off-chain implementations for any operations involving private key material and should consider restricting the visibility of internal helper functions to reduce the public API surface available to external contracts. No vulnerabilities affecting signature security, verification integrity, or fund safety were identified in this audit.