POSTERN LABS · SECURITY AUDIT

# ◇ Bloch-SIS-PoW Protocol — Security Audit

Ownerless post-quantum PoW base layer: crypto, consensus, reorg, P2P, shielded pool

2 CRITICAL | 5 HIGH | 3 MEDIUM | 6 LOW

Scope: `gitlab.com/blochsispow-group/BlochSISPoW-project` · 16 findings · Opus 4.8 auditors · 2026-07-15

**Nature of this document.** Internal, AI-assisted adversarial code review. It is **NOT** a third-party security audit and confers no assurance. Every finding was verified against source (file:line). The system is a disclosed **unaudited, pre-mainnet beta** running a relaxed **k=4** PoW that is openly documented as trivially forgeable — that known posture is not itself a finding. Fix criticals/highs before the corresponding feature is relied upon.

✓ **Remediation status (post-audit).** Every critical and high below has been FIXED in code and adversarially re-verified (C1 bounded network decode; C2 + nullifier length binds; H2 stratum validate_structure; H3 transport seal-once; H4 UTF-8-safe PEX; and the reorg DAG-ordering consensus-halt found during fix-verification). Fixes are merged into the protocol's main branch and this project's public demo nodes are being redeployed with them; other node operators must update + rebuild + restart to receive them (Bitcoin-model upgrade — nobody is forced; see posternlabs.com/security). One residual is scheduled for hardening: a narrow reorg concurrency race. Still UNAUDITED — the contracted third-party audit governs.

**Verified sound / no defect found.** Hybrid ML-DSA-65 ‖ Falcon-1024 signature (genuinely AND-secure — both legs verified, suite-committing envelope, no malleability, no missing-leg bypass); seeded-RNG fork (ChaCha20-from-seed, RAII guard, correct restore); SIS gate verification (correct header/nonce binding, fixed-size input, no shortcut beyond the disclosed posture); block/tx/shielded wire parsers (all counts/lengths bounded before allocation, strict trailing-byte rejection); sighash (domain- + chain-id-separated); transport handshake auth (Kyber768 + signed transcripts, constant-time compare); shielded u128 checked value math.

## Findings (16) — ranked by severity

### CRITICAL Remote single-packet OOM — untrusted gossip decoded with bincode config::standard() (no length limit)

`src/network/mod.rs:412`

**Scenario.** config::standard() sets NoLimit; Vec/String decoders allocate from the attacker's varint length prefix before reading payload. gossipsub max_transmit_size(4 MiB) bounds the payload, not the declared length: a ~20-byte NewBlock/NewTransaction/PeerExchange frame can declare 2^40 → ~1 TB allocation → allocator abort. Unauthenticated, single packet, whole-node crash, network-wide. (Second-pass verdict: REAL — confirmed against bincode 2.0.1 internals.)

**Fix.** bincode::config::standard().with_limit::<{4*1024*1024}>(); bound every Vec/String length field before trusting it.

### CRITICAL Shielded note counterfeiting — check_spend never binds public.out_commitments.len() to witness outputs.len() (latent)

`crates/coherence-core/src/lib.rs:194`

**Scenario.** The balance loop iterates only w.outputs and asserts public.out_commitments.get(i)==out.commitment(); it never checks the lengths are equal. A prover submits a balanced witness in=[100],out=[100],fee=0 but sets public.out_commitments=[cm(100), cm(1e9)]; index 0 matches, 100==100, Ok; the extra 1e9 note is appended to the tree as spendable → money from nothing. Latent (verifier=RejectAll, SP1 backend stubbed) but this is the FROZEN circuit statement that goes live when shielded verification is wired. (Second-pass verdict: REAL — the ZK statement itself is unsound.)

**Fix.** if public.out_commitments.len()!=w.outputs.len(){return Err(..)} plus the symmetric nullifiers.len()==inputs.len() bind.

### HIGH Symmetric gap: check_spend never binds public.nullifiers.len() to witness inputs.len()

`crates/coherence-core/src/lib.rs:180`

**Scenario.** Same class as C2 on the input side: extra committed nullifiers at indices >= inputs.len() are never validated, letting a prover mark arbitrary nullifiers spent (griefing / denial of others' notes). Latent behind RejectAll but frozen in the circuit.

**Fix.** Add if public.nullifiers.len()!=w.inputs.len(){return Err(..)} before the loop.

### HIGH Stratum SIS-PoW gate bypass — accept_block_cb routes blocks into accept_block() with NO validate_structure()

`src/main.rs:1046`

**Scenario.** accept_block() never calls validate_pow/validate_merkle/validate_coinbase; it relies on callers. The gossip path and rpc_submit_block call validate_structure(); the stratum accept_block_cb does NOT. Stratum blocks carry pow_solution=Vec::new(), and the only PoW check on that path is the 80-byte header hashcash — NOT the Module-SIS gate. A miner on the node's stratum server finds a cheap header hashcash (no SIS solution) → the node accepts a SIS-less block into its DAG/UTXO, forks itself off-network, enabling cheap local reorg + double-spend against services reading that node. (Second-pass verdict: REAL — a MISSING validation, distinct from the disclosed weak-k=4.)

**Fix.** Call block.validate_structure() at the top of accept_block_cb (mirror rpc_submit_block), or move the structural checks inside accept_block so no caller can skip them.

### HIGH PQ transport poll_write re-seals and retransmits the same buffer under backpressure

`src/transport/stream.rs:319`

**Scenario.** On a resume call after Poll::Pending, the inline path seals buf a second time (fresh counter N+1) and writes copy #2; the receiver opens both frames and delivers plaintext twice → yamux desync → connection torn down. Remotely inducible by a peer applying ordinary TCP backpressure. Latent (transport not yet libp2p-wired) but triggers on routine backpressure once live.

**Fix.** Seal a frame from buf exactly once; on resume only flush the existing pending frame and report its byte count.

### HIGH Remote panic — byte-slicing attacker-supplied PEX strings on non-UTF-8 boundaries

`src/network/mod.rs:512`

**Scenario.** info!/debug! log &addr_str[..len.min(60/80)] where addr_str is attacker-controlled PeerExchange.peers. &s[..n] panics if n is not a char boundary; a /dns4/<unicode-host>/ multiaddr straddling byte 60/80 panics the swarm event-loop task → node isolated.

**Fix.** addr_str.chars().take(N).collect::<String>() or addr_str.get(..n) with a fallback; never byte-slice untrusted strings.

### HIGH H1 VERDICT: NOT REAL (not exploitable) — reorg re-validation u64 .sum()

`src/reorg.rs:302`

**Scenario.** The first pass flagged out_sum: u64 = tx.outputs.iter().map(|o|.value).sum() as an overflow-to-inflation path. The focused second pass VERIFIED it is NOT reachable: upstream per-output value bounds (each < MAX_MONEY) and the count caps make a u64 wrap unreachable in validate_fork_block_state. Kept here as an audited-and-cleared item; still worth using checked_add/u128 for defense-in-depth and consistency with the extension path.

**Fix.** Optional hardening: try_fold(checked_add) or accumulate in u128, matching src/main.rs:1811.

### MEDIUM Untrusted-nonce length triggers a panic on keyfile decrypt (all 3 at-rest formats)

`crates/bloch-crypto/src/wallet/encryption.rs`

**Scenario.** The keyfile decrypt path slices/parses a nonce from the file without length-guarding it; a malformed/truncated keystore file → panic on decrypt (all three at-rest formats affected). Local file-tamper DoS; not remote, but crashes the wallet on a corrupted/hostile keystore.

**Fix.** Length-check the nonce/ciphertext framing and return a decrypt error instead of slicing/panicking.

### MEDIUM No reorg depth cap; reorg is non-transactional

`src/reorg.rs:160`

**Scenario.** Doc claims depth <= CHECKPOINT_DEPTH=1000 but nothing enforces it; to_rollback/to_apply are unbounded. A mid-reorg rollback error leaves storage in acknowledged partial state with no recovery. Partially mitigated by accept_block rejecting blocks at/under finalized_height.

**Fix.** Reject plans with to_rollback.len() > CHECKPOINT_DEPTH before executing; make rollback+apply a single atomic WriteBatch.

### MEDIUM bits_to_target fails OPEN (returns Target::MAX) on an out-of-range exponent

`crates/bloch-sis-pow/src/difficulty.rs:96`

**Scenario.** Wrong direction (should fail hardest). MITIGATED at consensus: accept_block recomputes expected_bits via pow::next_bits and rejects any mismatch, so header-supplied bits are never trusted. Defense-in-depth only.

**Fix.** Return Target::MIN on invalid compact-bits encodings.

LOW **Encrypted keystore written with default (world-readable) file permissions**

`crates/bloch-crypto/src/wallet/encryption.rs`

**Scenario.** The at-rest encrypted keystore is written without restricting mode (0600); on a multi-user host other local users can read the ciphertext (still AEAD-sealed, but unnecessary exposure).

**Fix.** Create the keystore with 0600 (OpenOptions.mode) on Unix.

LOW **i32::MIN.abs() overflow in verify/encode helpers**

`crates/bloch-sis-pow/src/verify.rs:103`

**Scenario.** i32::MIN.abs() overflows (debug panic / release wraps negative so >B passes). Only reachable via pub verify/verify_regime that bypass decode_s (which constrains coeffs to [-2,2]); the surviving i32::MIN still faces the true-value residual check, so not a standalone forgery.

**Fix.** v.unsigned_abs() > B as u32.

LOW **k=0 guardrail hole disables the SIS gate**

`crates/bloch-sis-pow/src/verify.rs:95`

**Scenario.** residual_regime_nontrivial(0,..) returns true, leaving only hashcash. Not used by shipped configs.

**Fix.** Assert residual_coeffs >= 1.

LOW **getrandom::fill(buf).expect() panics across the extern "C" FFI boundary (UB on unwind)**

`crates/pqcrypto-internals/src/lib.rs:171`

**Scenario.** Not attacker-triggerable, but a panic across a C FFI boundary is UB.

**Fix.** Return the C error code instead of expect().

LOW **verify(now_unix=0) silently skips the validity-window check (expired certs verify)**

`src/stratum_v2/cert.rs:108`

**Scenario.** A 0 sentinel disables the cert expiry check. Not currently called in-tree.

**Fix.** Use Option<u32> instead of a 0 sentinel.

LOW **codec: Option<NoiseCodec> — the None (plaintext) path is a latent auth-bypass on a shipped type**

`src/stratum_v2/session.rs:82`

**Scenario.** The dispatcher threads Option<NoiseCodec> reaching accept_block; the None/plaintext path is test-only but shipped.

**Fix.** Make the production type take NoiseCodec non-optionally.