Security Review Summary — v4

Subject: Keylay Encrypted QR Relay — index.html, server.js

Security page: security.html (keylay.org/security)

Audit scope: Cryptographic correctness, protocol claims, relay trust model, server hardening

Methodology: Manual source review, claim-vs-code differential analysis

Original review: April 2, 2026

This revision: August 2026 (R4)

Changes since v3: August 2026 code audit. Two security findings resolved in v0.8.0 (URL code exposure, exportable ephemeral key) plus a low-entropy-code warning; four low-severity findings disclosed as open with scheduled fixes (unauthenticated `format` field, fixed KDF salt, unbounded relayed decompression, one vendored library's provenance).

Reviewed against: app v0.8.0

Status: Internal review — not a formal third-party security audit

---

What was checked

Every security property stated in the Keylay security page was traced to its corresponding code path and evaluated independently. The review covered:

- Session code generation and entropy
- Channel identifier derivation (SHA-256 hashing)
- PBKDF2 key stretching and HMAC handshake authentication
- Ephemeral X25519 key pair generation and lifecycle
- HKDF session key derivation and participant binding
- AES-256-GCM message encryption, IV generation, and AAD construction
- Replay counter enforcement and counter-advance ordering
- Handshake state machine and concurrency handling
- Server-side rate limiting: payload cap, token bucket, per-IP cap, global cap, liveness sweep
- Session idle and max-lifetime timers (client and server)
- Pre-handshake message buffer cap
- Server-side logging and IP truncation
- Content Security Policy scope and claims

What was confirmed

All core cryptographic and operational claims check out against the deployed code:

- The relay sees only AES-GCM ciphertext and a derived channel hash — never plaintext or the raw session code.

- The HMAC handshake prevents relay key substitution without code knowledge.
- Forward secrecy holds as implemented. Ephemeral X25519 private keys are set to null immediately after session key derivation. Learning the code after a session ends does not decrypt past traffic.
- No session key is transmitted. Both peers derive the AES-256-GCM key locally from X25519 shared bits processed through HKDF, with the HKDF info field binding the key to the sorted pair of ephemeral public keys.
- Counter-bound AAD prevents replay of recorded ciphertext. The receive-side counter is advanced only after successful decryption.
- Unencrypted data messages are explicitly rejected by the client.
- The server correctly enforces two-peer session limits.
- No full IP addresses are logged — truncateIp() reduces every address to its /16 prefix before any log entry is written.
- Server enforces rate and size limits: maxPayload 256 KB, per-connection token bucket (burst 5, sustained 1 msg/s), per-IP cap of 10, global cap of 100.
- Sessions expire after 15 minutes idle or 60 minutes maximum, enforced on both server and client.

What was found and resolved — complete history

Revision 1 → 2

D-01: PBKDF2 iteration count mismatch — resolved.

security.html claimed the current deployment uses 100,000 iterations; target V1 is 300,000. The code had been using 300,000 all along. Security page updated.

D-02: Replay protection described as absent — resolved.

security.html claimed "Replay protection is not yet deployed." Counter-bound AAD and strict monotonic counter enforcement were already deployed. Security page updated.

D-03: Hello wire format described incorrectly — resolved.

security.html described hello as carried inside a data envelope. The code sends `{type:"hello", pubkey, sig}` directly. Security page updated.

F-01: Counter advanced before decryption — resolved.

The client was advancing its replay counter before decryption completed. A relay injecting a message with a fabricated counter value could consume the counter slot without a valid payload, causing subsequent legitimate messages to be rejected. Fixed: recvCounter is now assigned inside the try block, after `await decrypt()` returns successfully.

F-03: Server channel-hash logging — resolved.

The relay server was logging the channel hash (derived from the session code) and full message objects. Log output is now limited to message type and peer count.

F-07: Modular bias in code generator — resolved.

The 31-character alphabet does not divide evenly into 256, so the first eight characters were generated slightly more often. Rejection sampling now discards any byte value ≥ 248 before applying modulo 31. The generator is now correct by construction.

Revision 2 → 3

F-02: No server rate limiting or message size enforcement — resolved.

The relay now enforces maxPayload 256 KB (frame-layer rejection), a per-connection token bucket (burst 5, sustained 1 msg/s), per-IP connection cap of 10, and global connection cap of 100. A ping/pong liveness sweep runs every 30 seconds.

F-05: No Content Security Policy — resolved (partial).

CSP added in v0.7. connect-src limited to 'self' and wss://app.keylay.org; object-src, base-uri, and form-action all 'none'. The scope of protection is now accurately described in security.html (see D-05 below).

F-06: No server-side session expiry — resolved.

Sessions now expire on two independent timers: 15-minute idle (bumped on each data or hello message) and 60-minute maximum from first join. Both server and client enforce these timers independently. Key material is wiped on expiry via leaveChannel().

Client pre-handshake buffer cap — resolved.

messageBuffer is now bounded to MAX_BUFFERED_MESSAGES (32) with oldest-drop-on-overflow behavior.

Full IP addresses in server logs — resolved.

truncateIp() reduces every IP to its /16 prefix (first two octets for IPv4, first two hextets for IPv6) before any log entry is written.

D-04: Peer-key-change behavior changed; security.html not updated — resolved.

security.html stated: "If a peer reconnects with a different ephemeral public key while a session is active, the session key is cleared, counters are reset, and a full handshake is forced before data flows again." The code no longer does this — it silently ignores the incoming hello instead.

The change was deliberate: the prior reset-on-key-change behavior was a denial-of-service vector, since any peer knowing the code could force continuous session resets by sending hellos with fresh ephemeral keys. Silent ignore closes that vector. Legitimate reconnects now require a page reload. security.html has been updated to accurately describe the current behavior.

D-05: CSP confirmation overstated XSS protection — resolved.

security.html stated: "A successful XSS injection no longer has unrestricted access to the DOM, session key variables, and decrypted payloads." With `script-src 'unsafe-inline'` present in the

deployed policy, an injected inline script can still execute and read in-memory cryptographic state. The claim was false for DOM and JS state access.

What the CSP correctly restricts: connect-src limits where the page can send data, so an injected script cannot exfiltrate to an arbitrary external host. object-src, base-uri, and form-action are all 'none'. These are real protections — but they do not prevent an injected script from reading sessionKey, hmacKey, or decrypted plaintext from memory. security.html has been updated to accurately describe both what the CSP does and does not protect, and to disclose the 'unsafe-inline' limitation.

F-04: Unauthenticated role claiming — disclosed.

Any connected peer can send `{type: "claim"}` to take the sender role, demoting the current sender to receiver. This is within the stated threat model (code knowledge equals membership) and is now documented in security.html.

Revision 3 → 4 (August 2026 code audit, app v0.8.0)

An in-depth code audit reviewed the deployed client against every claim on this page. Two security findings were fixed before v0.8.0 shipped; four low-severity findings are disclosed as open with scheduled fixes.

S1: Session code exposed via the URL — resolved in v0.8.0.

The app read and pre-filled a session code from a `?code=` URL parameter. A code in the URL rides the HTTP request to the app host and can land in access logs, browser history, or the Referer header — contradicting the property that the raw code never touches the server. The URL parameter was removed; the code is entered manually only. Severity: medium.

S2: Ephemeral private key exportable — resolved in v0.8.0.

The per-session X25519 private key was generated extractable. Nothing exported it, but extractability let a successful XSS export the live private key, and forward secrecy relied on nulling the reference rather than browser enforcement. The key is now generated non-extractable. Severity: low–medium.

S6: No warning on low-entropy typed codes — resolved in v0.8.0.

Length was enforced but entropy was not, so a user could type an obviously weak code (e.g. all-same or sequential characters). A soft, non-blocking warning now nudges toward the generator. Severity: informational.

S3: Envelope `format` field not authenticated — OPEN, planned for protocol v2.

The GCM additional authenticated data binds the counter but not the `format` field; a relay can flip it and the client still decrypts and runs the relay-chosen branch. Demonstrated in a harness. Fix (bind `format` + channel identifier + protocol version into the AAD) is wire-breaking and batched into protocol v2. Severity: low, contained.

S11: Fixed PBKDF2 salt enables multi-target amplification — OPEN, planned for protocol v2.

The salt is a global constant, so key-stretching for a given code is identical across all sessions; a relay logging many handshakes can sweep the keyspace once against all captured sessions rather than per session. Fix (derive the salt per session from the channel identifier) is wire-breaking and batched with S3. Severity: low.

S4: Relayed decompression not output-bounded — OPEN, scheduled next release.

A relayed BBQR compressed payload is inflated with no output ceiling; a small crafted frame can expand enormously (≈1000× measured). Member-gated and download-triggered, but a compromised co-signer could crash a counterpart's tab. Fix: cap inflate output and reject implausible ratios. Severity: low.

S5: One vendored library lacks verifiable provenance — OPEN, scheduled next release.

Two of three inlined libraries hash to their published upstream builds; the QR encoder does not, so the "audit-equals-runtime" guarantee is unprovable for it at its pinned version. No tampering evidence. Fix: reconcile to the canonical build (or document build inputs) and add to a provenance table. Severity: low.

What remains open

Four findings from the August 2026 audit are open, all low-severity and contained within the stated threat model:

S3 — unauthenticated `format` field and **S11 — fixed KDF salt.** Both fixes are wire-breaking and batched into the next protocol revision (protocol v2).

S4 — unbounded relayed decompression and **S5 — one vendored library's provenance.** Both scheduled for the next release.

The following are tracked as longer-term hardening, not bugs in the current implementation:

Inline scripts not hash-pinned under CSP. Replacing `'unsafe-inline'` with explicit sha256 hash entries for each inline block requires a build step to compute and recompute hashes on every edit. Until this is done, the CSP protects exfiltration paths but not in-memory state from a successful XSS.

meta-tag CSP limitations. frame-ancestors, report-uri, and sandbox are not honored in a `<meta>` CSP. Setting `frame-ancestors 'none'` via an HTTP response header at the static-hosting layer is recommended to prevent clickjacking.

PBKDF2 is not memory-hard. Argon2id or scrypt would be more resistant to GPU-accelerated offline attacks against the session code, but neither is available in the browser's native crypto and vendoring a WASM implementation conflicts with the protocol-crypto no-dependencies rule; deferred pending native support. Raising the generated code length is the cheaper interim lever.

What this review is and is not

This review is a thorough manual analysis with direct access to the source. It constitutes a detailed technical review but not a formal third-party security audit. No symbolic verification, fuzzing, or independent reproducibility testing was performed.

The full technical report, including annotated code paths, severity ratings, and the complete finding history from Revisions 1–4, is published at keylay.org/Keylay_Security_Review_v4.pdf.

Review date: August 2026 (R4 — app v0.8.0. Resolved in v0.8.0: S1 URL code exposure, S2 exportable ephemeral key, S6 low-entropy-code warning. Disclosed open, scheduled: S3 unauthenticated `format` field and S11 fixed KDF salt (protocol v2); S4 unbounded relayed decompression and S5 vendored-library provenance (next release). security.html updated to match. Prior: R3 April 2026 — all then-open items resolved; R1/R2 April 2026.)