§ INDEPENDENT REVIEW · QUANTACIPHER SDK FAMILY

An honest audit of QuantaCipher

A second, independent review of the published `quantacipher-sdk`, `quantacipher-wasm`, `quantacipher` (PyPI), and `quantacipher-core` artifacts, their documentation, and platform. Prepared by QuantaKrypto under the fix-first disclosure we agreed: shared privately, corrected, then published.

The short version

The cryptographic fundamentals are sound. The random number generation, the authenticated encryption (AES-256-GCM), and the KEM-to-cipher construction are all done correctly. Nothing in this review exposes user data, and the "encryption happens locally" claim substantially holds.

The material issue is accuracy, not a broken cipher. The product advertises "NIST ML-KEM / FIPS 203, Kyber-1024" but ships pre-standard round-3 **Kyber-768**. That is measurable, and it matters for buyers who need NIST or CNSA 2.0 compliance, it will not pass a FIPS 203 test and will not interoperate with real ML-KEM, but it is a labelling and compliance gap, not a data-exposure risk.

One dependency hygiene item is worth acting on: the KEM crate is unmaintained and carries a known timing-side-channel advisory with no upstream fix. Practical exploitability is low in a local-only model, but it should be migrated. **A single change, moving to the maintained `ml-kem` crate, resolves the compliance gap and the advisory at once.**

What is already right

Sound cryptographic engineering

- CSPRNG throughout (OsRng / Web Crypto), no weak randomness
- AES-256-GCM authenticated encryption, fresh 96-bit nonce per message
- A fresh KEM shared secret per message, so key/nonce reuse is structurally impossible
- Exact length checks on keys, ciphertext, and nonce before use; strict base64

Honest posture, mostly

- The WASM engine has no network capability; only ciphertext (never plaintext or private keys) leaves the box
- The Python wheel ships a CycloneDX SBOM, above average for the category
- Runtime dependencies are minimal
- The "local-only encryption" claim holds, with the metadata caveat below

We lead with this because it is true, and because the fixes below are refinements of a fundamentally reasonable design, not a rescue.

Findings

ACCURACY / COMPLIANCE

Shipped algorithm does not match the advertised standard

Every README, the website, and the docs state "NIST ML-KEM (Kyber-1024) / FIPS 203". The only KEM dependency is `pqc_kyber 0.7.1`, which implements **round-3 CRYSTALS-Kyber** (last released 2023, before FIPS 203). Round-3 Kyber is not KAT- or wire-compatible with ML-KEM, and with no parameter feature set it resolves to **Kyber-768**, not 1024. We measured the published WASM at runtime: public key 1184 B, private key 2400 B, ciphertext 1088 B, an exact Kyber-768 match, while the core README's own size table claims 1568 / 3168 / 1568.

Impact: a real accuracy and compliance gap for a vendor selling on NIST language. It will fail a FIPS 203 conformance test and will not interoperate with standard ML-KEM. It does **not** put encrypted data at risk, round-3 Kyber-768 is a strong, unbroken post-quantum KEM.

Fix: migrate the KEM to the maintained `ml-kem` crate at the 1024 parameter set (true FIPS 203), rebuild the crate, WASM, and wheel, and either back the "ML-KEM / FIPS 203" claims or soften them until the migration ships.

ACCURACY

The runtime API reports a hardcoded, incorrect algorithm string

`generate_keypair()` returns an `algorithm` field hardcoded to the literal `"Kyber-1024"` (present in the WASM binary and the Python `.so`), regardless of what is actually compiled. Downstream code that reads this for policy, inventory, or attestation records a value that is wrong twice over.

Fix: derive the algorithm string from the compiled parameters (`KYBER_PUBLICKEYBYTES`) and add a build-time assertion.

SECURITY HYGIENE

Unmaintained KEM dependency with a known timing advisory

`cargo audit` flags `pqc_kyber 0.7.1` for **RUSTSEC-2023-0079 (KyberSlash)**, a secret-dependent division-timing side-channel in decapsulation, CVSS 7.4, with **no fixed version** (the crate is abandoned). It touches the one mode that holds a long-lived key.

Impact, honestly: in QuantaCipher's local-only model there is usually no remote attacker feeding ciphertexts and measuring decapsulation timing, so practical exploitability is low. The real issue is depending on an unmaintained crate with an open advisory in the product's core primitive.

Fix: the same `ml-kem` migration removes it (constant-time, maintained). Add `cargo audit` / `cargo deny` to CI.

SUPPLY-CHAIN TRUST

No public source repository; builds are unverifiable

All four packages declare `github.com/xaexaex/quantacipher` as their source; it returns **HTTP 404** on both the web and the API. No third party can diff or reproduce the shipped WASM and native binaries against source. There is also no build provenance on npm, PyPI, or crates.io, and both binaries embed a developer's local paths (built on a workstation, not CI).

Fix: publish the repository (the full source already ships inside the PyPI sdist, so openness costs nothing), build in CI with provenance (npm `--provenance`, PyPI Trusted Publishing), and tag releases.

OPERATIONAL

The default gateway does not resolve

The TypeScript SDK defaults to `https://api.quantacipher.com/v1/ingest`, which is **NXDOMAIN** (the apex `quantacipher.com` is live; only the `api` host is missing). The Python SDK defaults to `http://localhost:4000`, a developer default shipped as a release. So default installs of the gateway calls fail.

Fix: point both SDKs at the real production HTTPS endpoint, reject non-HTTPS for non-localhost, add request timeouts.

CLAIM ACCURACY

"The gateway only ever sees ciphertext" is not quite true

The gateway call posts `{ ciphertext, metadata, timestamp }`. The `metadata` is sent unencrypted, and the SDK's own examples put identifying values there (`userId: 'U-001'`, `type: 'hipaa_audit'`). For HIPAA use, that is exactly where re-identifiable data would leak.

Fix: encrypt metadata, or state precisely what the gateway observes rather than "only ciphertext".

HARDENING

Missing KDF / AAD binding / zeroization, and a vulnerable pyo3

Best-practice hardening, none of these is a practical break: the raw KEM shared secret is used directly as the AES key with no HKDF and no domain separation; the payload header and KEM ciphertext are not bound into the GCM AAD (the framing is malleable); secret buffers are never zeroized (despite Vault Mode being documented as "deterministically drops the private key"); and the published wheel pins `pyo3 0.20.3`, which carries two advisories. The SDK also depends on `quantacipher-wasm: "*"`, a wildcard on the crypto engine.

Fix: HKDF-SHA256 with a context label; bind the header + KEM ciphertext as AAD; `zeroize` secret buffers; bump `pyo3 ≥ 0.29`; pin the WASM version.

DESIGN HONESTY

"Vault Mode" is irreversible sealing marketed for record retention

Vault Mode encrypts to an ephemeral key that is then discarded, so the data is permanently unrecoverable, functionally, secure deletion. It is marketed for "HIPAA audit logs and compliance records", but HIPAA audit controls require records to remain examinable; permanently unreadable records satisfy retention only on paper. Not insecure, but the threat model should be stated plainly.

Fix: for tamper-evidence use a signed hash chain or timestamped digest; document Vault Mode as irreversible so no one relies on it for records they must later read.

Measured parameter set, per artifact

ARTIFACT	CLAIMED	ACTUALLY SHIPPED	HOW WE KNOW
quantacipher-wasm 0.3.0	Kyber-1024 / ML-KEM	round-3 Kyber-768	Runtime measurement: pk 1184 / sk 2400 / ct 1088 B (exact Kyber-768)
quantacipher-core 0.2.0	Kyber-1024 / FIPS 203	round-3 Kyber-768	`pqc_kyber 0.7.1`, no `kyber1024` feature → KYBER_K = 3
quantacipher 0.2.0 (PyPI)	Kyber-1024	round-3 Kyber-768	Core source byte-identical to crates.io; SBOM lists `pqc_kyber 0.7.1`
quantacipher-sdk 1.2.0 (npm)	NIST ML-KEM	round-3 Kyber-768	Pure wrapper; all crypto delegated to the WASM engine above

Automated scan

QuantaKrypto `qScan 0.7.0` ran clean (0 findings) across all four packages. That is expected and not a clean bill of health: qScan detects **classical** quantum-vulnerable cryptography, and this is a PQC library, so the meaningful signal here comes from dependency-advisory scanning (`cargo audit`) and the claim-versus-artifact checks above, not from qScan. Raw qScan output for each package accompanies this report.

The one fix, and how we can help

The single highest-value change

Migrate `pqc_kyber` → `ml-kem` at the 1024 parameter set. One dependency swap in `quantacipher-core` makes the FIPS 203 / ML-KEM claim true, delivers the Level 5 security you advertise, and removes the KyberSlash advisory, all at once. Ship it as a versioned v2 payload so existing ciphertexts still decrypt.

Where QuantaKrypto fits

- **Sieve** conformance-test the KEM against FIPS 203 (the concrete before/after)
- **crypto-agility.json** published on your site, declaring current vs target posture
- **qScan** in CI to gate the classical surfaces (gateway, dashboard)
- The **QRAMM** readiness assessment, and claiming your public repos for an automated badge + an on-chain audit certificate

Reviewers

Krzysztof Cywiński

Security review · initial audit pass

linkedin.com/in/krzysztofcywinski
github.com/krzysztof-cywinski

León Acosta

QuantaKrypto · independent review & methodology

linkedin.com/in/leonacosta
github.com/quantakrypto