§ INDEPENDENT REVIEW · QUANTALABSS/QUANTACIPHER

QuantaCipher — post-quantum security review

QuantaChain / QuantaLabs · Application · Rust · 6 findings

The short version

Sound fundamentals — accuracy and compliance fixes required

The original review found sound cryptographic fundamentals and no data-exposure path, but identified accuracy, compliance and supply-chain issues. Five of six findings are now fixed. The ML-KEM migration and endpoint corrections were verified against published releases. Source and build provenance are now verifiable for the scoped SDK and WASM 2.0.6 releases from the official quantalabss/quantacipher repository. The gateway-metadata finding remains open.

What it gets right

Correct CSPRNG throughout

KEM keygen and encapsulation use OsRng, nonces use the AEAD's OsRng, and in WASM that is Web Crypto getRandomValues. No weak randomness anywhere.

Sound KEM/DEM construction

AES-256-GCM authenticated encryption, a fresh 96-bit random nonce per operation, and a fresh KEM shared secret per message, so key and nonce reuse are structurally impossible.

The local-only claim substantially holds

The WASM engine has no network capability. Plaintext and private keys are never transmitted; only ciphertext leaves the runtime, plus the metadata noted in the findings.

Careful input validation

Exact length checks on keys, ciphertext and nonce, and strict base64. The Python wheel even ships a CycloneDX SBOM, which is above average.

The fix that matters most

Migrate the KEM from pqc_kyber to the maintained ml-kem crate, at the 1024 parameter set

This single change makes the FIPS 203 ML-KEM claim true, closes the Kyber-768-versus-1024 discrepancy, and removes the unmaintained dependency carrying the KyberSlash advisory. Three of the findings below collapse into one migration. It has since been made: quantacipher-core 0.3.5 depends on ml-kem and the published WASM measures ML-KEM-1024.

Findings

F1 · Advertised algorithm does not match the shipped one

MEDIUM | ACCURACY / COMPLIANCE | FIXED

QuantaCipher advertises "NIST ML-KEM / FIPS 203 / Kyber-1024", but the shipped key sizes measure public key 1184 / secret key 2400 / ciphertext 1088 bytes — those are Kyber-768, and a pre-standard round-3 variant at that. It therefore fails FIPS 203 and will not interoperate with a real ML-KEM implementation. This is a correctness and compliance defect, not a data-exposure risk: the exchange itself is not broken.

EVIDENCE, MEASURED RATHER THAN INFERRED

Runtime execution of the published quantacipher-wasm 0.3.0 returned key sizes that are an exact Kyber-768 match, contradicting the product's own documented sizes. Root cause: pqc_kyber 0.7.1 with no kyber1024 feature, so KYBER_K defaults to 3. Separately, pqc_kyber 0.7.1 implements round-3 CRYSTALS-Kyber, last released in 2023, which is not FIPS 203 ML-KEM and is neither KAT- nor wire-compatible with it.

```
actual  pk=1184  sk=2400  ct=1088   → Kyber-768
docs    pk=1568  sk=3168  ct=1568   → Kyber-1024 (claimed)
```

IMPACT

Two true statements at once: the delivered security level is NIST Level 3, not the Level 5 advertised, and the algorithm is pre-standard Kyber rather than ML-KEM. For buyers who need FIPS 203 or CNSA 2.0 compliance, or interoperability with a real ML-KEM implementation, the product did not meet its claim. Data-exposure risk: none, Kyber-768 is unbroken.

FIX

Either advertise the parameter set actually shipped, or move to a standardised ML-KEM implementation (see the single recommended fix below). Re-measured on the published quantacipher-wasm 0.4.4: public key 1568 / secret key 3168 bytes, which are the FIPS 203 ML-KEM-1024 sizes, and the build identifies itself as 3.0.0-dual-mode-mlkem-fips203. The shipped parameter set now matches the advertised one.

`Release: quantacipher-wasm 0.4.4: https://www.npmjs.com/package/quantacipher-wasm/v/0.4.4`
`Release: quantacipher-core 0.3.5: https://crates.io/crates/quantacipher-core/0.3.5`

F2 · Unmaintained pqc_kyber crate carries the KyberSlash advisory

MEDIUM | SECURITY HYGIENE | FIXED

The dependency pqc_kyber is unmaintained and is subject to KyberSlash (RUSTSEC-2023-0079, CVSS 7.4) with no fix available. Practical exploitability is low in QuantaCipher's local-only usage model — the timing side-channel needs an adversary positioned to measure decapsulation — but shipping an unmaintained crate with an open advisory is not acceptable for a security product.

EVIDENCE, MEASURED RATHER THAN INFERRED

cargo audit against the published dependency graph flagged RUSTSEC-2023-0079 in pqc_kyber 0.7.1. The crate's last release was in 2023 and no fixed version exists.

IMPACT

KyberSlash is a timing side channel in the key-recovery class, CVSS 7.4. Exploitability in a local-only usage model is low, since an attacker needs to be positioned to measure decapsulation. Shipping an unmaintained crate with an open advisory and no upgrade path is still not acceptable in a security product.

FIX

Migrate off pqc_kyber to a maintained implementation (see the single recommended fix below).

`Release: quantacipher-core 0.3.5: https://crates.io/crates/quantacipher-core/0.3.5`
`Dependency manifest: https://crates.io/api/v1/crates/quantacipher-core/0.3.5/dependencies`

F3 · Source and build provenance are now verifiable

MEDIUM | SUPPLY-CHAIN | FIXED

At the original review, the declared xaexaex/quantacipher repository returned 404 and published artifacts lacked verifiable build provenance. The official source is now public at quantalabss/quantacipher. On 12 September 2026, we verified GitHub build attestations for the JavaScript entry point and WASM binary extracted from the published @quantalabss/quantacipher-sdk and @quantalabss/quantacipher-wasm 2.0.6 npm packages.

EVIDENCE, MEASURED RATHER THAN INFERRED

Both npm 2.0.6 packages declare the official repository and gitHead 88fcab36615f7a368ef291d0355541f90af0b7b2. gh attestation verify succeeded for dist/src/index.js and quantacipher_wasm_bg.wasm, validating Sigstore signatures and SLSA provenance from .github/workflows/release.yml at refs/tags/v2.0.6, run 33234222573.

```
SDK index.js SHA-256
69e9b63fa1e195842e7daa2bd9c267d892e1f0f4e09af55e09db74a32a4a1ac7
WASM SHA-256
13865e4fb81f347bbcb82a353bd7647a4abad3a0413f086ea71030c1a93232cf
```

IMPACT

Consumers can inspect the official source and verify that the checked SDK and WASM files were built by its GitHub release workflow. The original source-availability and build-origin blocker is resolved for those artifacts; wildcard dependency selection still permits the installed WASM version to change.

FIX

The source/provenance issue is closed for the verified 2.0.6 SDK and WASM artifacts. Consumers can verify them with gh attestation verify --repo quantalabss/quantacipher. Dependency pinning remains a separate caveat: the SDK still declares @quantalabss/quantacipher-wasm as "*". Pin the installed WASM version and retain a lockfile. This check does not establish reproducible builds or verify the older unscoped npm, PyPI or crates.io releases.

```
Official source at verified commit: https://github.com/quantalabss/quantacipher/tree/88fcab36615f7a368ef291d0355541f90af0b7b2
Verified release workflow: v2.0.6: https://github.com/quantalabss/quantacipher/actions/runs/33234222573
SDK 2.0.6 registry metadata: https://registry.npmjs.org/@quantalabss/quantacipher-sdk/2.0.6
WASM 2.0.6 registry metadata: https://registry.npmjs.org/@quantalabss/quantacipher-wasm/2.0.6
```

F4 · Gateway metadata sent unencrypted, contradicting the zero-trust claim

MEDIUM | CLAIM ACCURACY | OPEN

QuantaCipher markets a zero-trust posture, but gateway metadata is transmitted unencrypted. The claim and the observed behaviour disagree; the metadata channel is not protected to the standard the product advertises.

EVIDENCE, MEASURED RATHER THAN INFERRED

The gateway call posts ciphertext and metadata as sibling fields in the same JSON body. The vendor's own HIPAA example passes userId and record type in that metadata.

IMPACT

The zero-trust claim is that the operator learns nothing. Whoever runs the gateway learns who is talking, about what kind of record, and when. The payload is protected and the pattern of use is not, and for the health-record example given in the docs the pattern is often the sensitive part.

FIX

Encrypt the metadata channel end-to-end, or scope the zero-trust claim to what is actually protected.

F5 · Runtime returns a hardcoded algorithm label

LOW ACCURACY FIXED

generate_keypair() returns a hardcoded algorithm: "Kyber-1024" string regardless of the parameters actually in use, so callers and telemetry are told the wrong algorithm at runtime. This compounds the advertised-vs-shipped mismatch above.

EVIDENCE, MEASURED RATHER THAN INFERRED

generate_keypair() returned a constant algorithm string in both the WASM build and the Python shared object, independent of the parameters actually compiled in.

IMPACT

Downstream policy engines, CBOM generators and attestation tooling record whatever the runtime tells them. A constant string means those records are wrong wherever the constant is wrong, and they are trusted precisely because they came from the runtime rather than from documentation.

FIX

Derive the reported algorithm from the implementation in use rather than a constant. generate_keypair() on 0.4.4 returns algorithm "ML-KEM-1024", which agrees with the measured key sizes. The label is no longer contradicted by the artifact.

`Release: quantacipher-wasm 0.4.4: https://www.npmjs.com/package/quantacipher-wasm/v/0.4.4`

F6 · Default endpoints do not resolve

LOW OPERATIONAL FIXED

The default gateway host api.quantacipher.com returns NXDOMAIN, and the Python SDK defaults to http://localhost. Out of the box the client points at endpoints that do not exist, so any default-configured deployment fails or silently talks to the wrong host.

EVIDENCE, MEASURED RATHER THAN INFERRED

The TypeScript SDK's default gateway, api.quantacipher.com, did not resolve in DNS. The Python SDK defaulted to http://localhost.

IMPACT

A deployment issue rather than a cryptographic one, but it blocks independent verification of the upstream path and means the documented default configuration does not work as shipped.

FIX

Ship working defaults, or fail closed with a clear configuration error when no endpoint is set. quantacipher-sdk 1.4.3 defaults to https://quantacipher.com/api/v1/ingest, which resolves, and rejects any non-HTTPS gateway except localhost. The dead api.quantacipher.com default is gone.

`Release: quantacipher-sdk 1.4.3: https://www.npmjs.com/package/quantacipher-sdk/v/1.4.3`

F7 · One migration resolves the compliance gap and the advisory together

INFO | RECOMMENDED FIX | ADVISORY

Migrating from pqc_kyber to the maintained ml-kem crate resolves both major threads at once: it moves QuantaCipher onto a standardised, FIPS 203-aligned ML-KEM implementation (closing the advertised-vs-shipped compliance gap) and off the unmaintained crate carrying the KyberSlash advisory. This is the single highest-leverage change.

IMPACT

Three findings, one migration. Worth reading as a single piece of work rather than a backlog.

FIX

Adopt the ml-kem crate for key encapsulation and re-run the parameter and interoperability checks.

`Release: quantacipher-core 0.3.5: https://crates.io/crates/quantacipher-core/0.3.5`

Automated scan (qScan v0.7.0)

PACKAGE	FILES SCANNED	FINDINGS
quantacipher-sdk	3 (5 with minified)	0
quantacipher-wasm	6	0
quantacipher-core	7	0
quantacipher (PyPI)	16	0

Read this correctly. qScan detects classical, quantum-vulnerable cryptography: RSA, ECDH, ECDSA, legacy TLS. A post-quantum library correctly shows none, which is the expected result and not a clean bill of health for everything else. The findings in this review come from measurement, from dependency advisories, and from comparing claims against artifacts, all of which are complementary to what qScan does.

Merged upstream

Release: quantacipher-wasm 0.4.4: https://www.npmjs.com/package/quantacipher-wasm/v/0.4.4

Release: quantacipher-core 0.3.5: https://crates.io/crates/quantacipher-core/0.3.5

Dependency manifest: https://crates.io/api/v1/crates/quantacipher-core/0.3.5/dependencies

Official source at verified commit:
https://github.com/quantalabss/quantacipher/tree/88fcab36615f7a368ef291d0355541f90af0b7b2

Verified release workflow: v2.0.6: https://github.com/quantalabss/quantacipher/actions/runs/33234222573

SDK 2.0.6 registry metadata: https://registry.npmjs.org/@quantalabss/quantacipher-sdk/2.0.6

WASM 2.0.6 registry metadata: https://registry.npmjs.org/@quantalabss/quantacipher-wasm/2.0.6

Release: quantacipher-sdk 1.4.3: https://www.npmjs.com/package/quantacipher-sdk/v/1.4.3

Scope & method

Artifacts reviewed

- quantacipher-sdk@1.2.0 (npm)
- quantacipher-wasm@0.3.0 (npm)
- quantacipher@0.2.0 (PyPI, sdist and wheel)
- quantacipher-core@0.2.0 (crates.io), plus the docs and platform

Method

Package checksums verified against registry metadata. qScan v0.7.0 on each package. Runtime execution of the published WASM to measure real key and ciphertext sizes. cargo audit and cargo tree across the dependency graph. Source read of the Rust core and the bindings. DNS and HTTP checks on the gateway and the repository. Two independent reviewers.

A point-in-time review of published artifacts. It does not cover the closed-source gateway service, the platform back end, or anything not shipped in the four packages above. Shared privately first, under a fix-first disclosure arrangement.

Reviewers

Krzysztof Cywiński

León Acosta