Q·NSQY

QUANTUM SEQURITY

ACVP Test-Vector Report (June 2026, historical)

NIST Automated Cryptographic Validation Protocol Test Vector Replay

Document version: 1.0

Publication date: 2026-06-03

Software: QNSQY, June 2026 build

ACVP-Server: v1.1.0.42

ACVP-Server commit: `15c0f3deeefbfa8c...`

Publisher: Quantum Sequrity (QNSQY)

Contact: security@quantumsequrity.com

Disclaimer

This document presents byte-for-byte test-vector validation evidence. It does not claim, imply, or substitute for NIST certification, NIST approval, or FIPS 140-3 / CMVP validation. NIST does not certify products. QNSQY is not listed on NIST's Cryptographic Module Validation Program (CMVP) Validated Modules list. CMVP is on the QNSQY roadmap; no application has been submitted to NIST as of the date on this document. The evidence in this document corresponds to the algorithm-correctness portion of a CMVP submission, but the surrounding boundary, role, self-test, and operational requirements have not been independently evaluated.

Executive Summary

QNSQY is a post-quantum cryptography tool for protecting data, implementing NIST FIPS 203 (ML-KEM), FIPS 204 (ML-DSA), FIPS 205 (SLH-DSA), and the supporting hash, KDF, MAC, AEAD, and elliptic-curve primitives required for hybrid post-quantum cryptographic operations.

To demonstrate algorithm correctness, QNSQY maintains a test harness that replays the public NIST Automated Cryptographic Validation Protocol (ACVP) vectors against the same Rust crates included in the shipping binary. The harness operates entirely on public material: the ACVP-Server vector tree, the production crate set, and the harness source itself are all available under open licenses.

The 2026-06-03 sweep, executed against ACVP-Server v1.1.0.42 (commit `15c0f3deeefbfa8cb6cd32a99e1ca3b738c66bf0`), produced the following result:

Result	Count
Vectors passed	**18,703**
of which post-quantum (ML-KEM + ML-DSA + SLH-DSA)	1,998
of which HMAC (classical)	12,375
of which other classical (SHA-2/3, SHAKE, AES-GCM, Ed25519, X25519)	4,330
Vectors failed	**0**
Vectors skipped (documented)	**11,259**
of which real coverage gaps in upstream Rust crates	135
of which outside QNSQY algorithm scope	~11,124
Total vectors attempted	**29,962**
ACVP algorithm directories at 100% NIST coverage	**33 of 50**
Sweep wall-clock (single machine, varies by hardware)	719.66 seconds

A reviewer evaluating QNSQY as a post-quantum-cryptography product should focus on the 1,998 PQC-specific vectors (ML-KEM 180, ML-DSA 570, SLH-DSA 1,248), all of which pass. The remaining ~16,705 vectors

are classical primitives (HMAC, SHA-2, SHA-3, etc.) that QNSQY also exercises but that pre-date the post-quantum question.

Section 4 of this report itemizes every skip category with a reason and the upstream issue or design rationale.

---

1. Scope and Method

1.1 What ACVP Is

The NIST Automated Cryptographic Validation Protocol (ACVP) is the test-vector layer of NIST's Cryptographic Algorithm Validation Program. Each ACVP algorithm directory in the public `usnistgov/ACVP-Server` repository carries two JSON files: `prompt.json` containing the inputs (keys, messages, nonces, etc.) and `expectedResults.json` containing the answers the implementation is required to produce. ACVP validation is mechanical: a correct implementation reproduces the answers byte-for-byte; a divergence is a real bug.

ACVP is used by NIST-accredited Cryptographic and Security Testing Laboratories (CSTLs) as one component of a CMVP submission. The other CMVP components (Security Policy, operational testing on the cryptographic boundary, roles and services testing, self-test verification) are out of scope for this report.

1.2 Crates Tested

The harness invokes the public RustCrypto crates that QNSQY's production build also links against. The version pins are mirrored in the standalone reproducer's `Cargo.toml` (published in `qnsqy-acvp-harness-v1.0.tar.gz`):

- `sha2` (FIPS 180-4)
- `sha3` (FIPS 202)
- `hmac` (FIPS 198-1)
- `aes-gcm` 0.10 (SP 800-38D)
- `ed25519-dalek` (RFC 8032)
- `x25519-dalek` (RFC 7748)
- `ml-kem` 0.2.3 (FIPS 203)
- `ml-dsa` 0.1.0-rc.7 (FIPS 204)
- `slh-dsa` 0.2.0-rc.1 (FIPS 205)
- `hbs-lms` 0.1.1 (SP 800-208, structural KAT only; ACVP vectors not available)
- `hkdf` (RFC 5869)
- `argon2` (RFC 9106)

The test harness pulls three dev-only crates that do not enter production: `sha1` (for the legacy HMAC-SHA1 vectors), `ml-kem` with the `deterministic` feature flag (for keyGen DRBG seeding), and `hybrid-array 0.4` (so the ml-dsa externalMu types unify with the harness call site). No production source file is modified.

1.3 Vector Provenance

The ACVP-Server repository at `https://github.com/usnistgov/ACVP-Server` is pinned to commit `15c0f3deeefbfa8cb6cd32a99e1ca3b738c66bf0` (release tag v1.1.0.42, snapshot date 2026-04-16). A sparse-checkout of `gen-val/json-files/` provides the full ACVP algorithm directory tree (169 directories, roughly 1 GiB on disk). The QNSQY harness exercises 50 of these.

1.4 Harness Location

The harness is part of QNSQY's test suite, organized as twelve loader modules plus a shared `mod` providing the JSON reader and the `Report` struct. Fifty test functions wire the loaders to algorithm directories. Each test function loads the directory's `prompt.json` and `expectedResults.json`, iterates the vectors, calls the production crate, compares the output to the expected value, and emits a single `ACVP_REPORT name=DIR total=T pass=P fail=F skip=S` line for parsing.

---

2. Results

2.1 Top-Line Numbers

The 2026-06-03 sweep produced:

```
test result: ok. 50 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out
```

with aggregate vector counts of 18,703 pass, 0 fail, 11,259 skip, 29,962 total.

2.2 Per-Family Breakdown

Family	Specification	Vectors passed	Vectors skipped	Coverage notes
HMAC	FIPS 198-1, RFC 4231	12,375	0	22 directories (every hash family, both ACVP schemas) at 100%
SHA-2	FIPS 180-4	2,947	bit-aligned subset	SHA-2-256 / 512 / 512-256 -1.0 at 100% (AFT, MCT-alternate, LDT up to 8 GiB)
SHA-3	FIPS 202	538	bit-aligned subset	SHA3-{224,256,384,512} byte-aligned AFT + MCT-standard + LDT
SHAKE	FIPS 202	756	2 MCT variable-output	SHAKE-128-FIPS202 at 100% (269 vectors)
ML-KEM	FIPS 203	180	60	keyGen 100%; encapDecap 105/165; 60 EncapsKeyCheck/ DecapsKeyCheck blocked by upstream crate
ML-DSA	FIPS 204	570	45	keyGen 100%; sigVer 100%; sigGen 315/360; 45 externalMu+randomized blocked by upstream crate
	FIPS 205	1,248	0	All 12 parameter sets at 100%, every flavor

Family	Specification	Vectors passed	Vectors skipped	Coverage notes
SLH-DSA				
Ed25519	FIPS 186-5, RFC 8032	9	19	Pure Ed25519 at 100%; Ed25519ph and Ed448 not used
X25519	RFC 7748	50	32	XECDH-SSC at 100%; cross-protocol keyVer not implemented
AES-GCM	SP 800-38D	30	30	Tag-length >= 96 bits at 100%; sub-floor vectors rejected by crate per §5.2.1.2
HKDF	RFC 5869	covered	2,000	RFC 5869 paths pass; 2,000 TwoStep KDA-HKDF skips (unused construction)
Total		**18,703**	**11,259**	33 of 50 directories at 100%

The full per-directory `ACVP_REPORT` lines, including individual pass / fail / skip totals for every directory tested, are recorded verbatim in `acvp-RUN-final.log`. The machine-readable summary (per-directory counts plus aggregate totals) is at `acvp-summary.json` and `acvp-summary.csv`.

2.3 Directories at 100% NIST Coverage

The following 33 ACVP directories produced zero skip and zero fail; every published vector in each directory was reproduced byte-for-byte:

- HMAC-SHA-1, HMAC-SHA-2-224, HMAC-SHA-2-256, HMAC-SHA-2-384, HMAC-SHA-2-512, HMAC-SHA-2-512-224, HMAC-SHA-2-512-256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512 (each in both the -1.0 and -2.0 ACVP schemas: 22 directories, 12,375 vectors)
- ML-DSA-keyGen (75 vectors, 3 parameter sets)
- ML-DSA-sigVer (180 vectors, 3 parameter sets)
- ML-KEM-keyGen (75 vectors, 3 parameter sets)
- SLH-DSA-keyGen, SLH-DSA-sigGen, SLH-DSA-sigVer (1,248 vectors total, 12 parameter sets across every signatureInterface flavor)
- SHA-2-256 -1.0, SHA-2-512 -1.0, SHA-2-512/256 -1.0 (2,575 vectors, AFT + MCT-alternate + LDT to 8 GiB)
- SHAKE-128-FIPS202 (269 single-output vectors)
- XECDH-SSC RFC 7748 (50 X25519 self-consistency pair tests)

The remaining 17 algorithm directories contain at least one variant outside the implemented scope. None contains a failing vector.

3. Reproducibility

The sweep can be reproduced on any Linux workstation with a Rust toolchain. Realistic end-to-end time on a first run is 30 to 45 minutes: 5 to 15 minutes for the 667 MiB sparse clone in Step 1 (network-dependent), 5 to 10

minutes for `cargo build` to download and compile the dependency tree, and 10 to 15 minutes for the sweep itself. Subsequent runs with everything cached are 10 to 15 minutes.

3.1 Step 1. Stage the vector tree

```
mkdir -p /tmp/qnsqy-test-run/acvp-source
cd /tmp/qnsqy-test-run/acvp-source
git init
git remote add origin https://github.com/usnistgov/ACVP-Server.git
git config core.sparseCheckout true
echo "gen-val/json-files/" > .git/info/sparse-checkout
git fetch --depth 1 origin 15c0f3deeefbfa8cb6cd32a99e1ca3b738c66bf0
git checkout FETCH_HEAD
```

3.2 Step 2. Run the public reproducer

The internal build steps of the original report are not reproduced here. The standalone reproducer harness published on https://quantumsequrity.com/validation runs the same NIST vector sets against the same open-source crates; its README gives the commands.

If the reproduction produces different numbers, the cause is either drift in the upstream Rust crates (which we track), a different ACVP-Server commit (the pinned commit is the contract), or a real correctness bug. The QNSQY security contact accepts reproduction reports at `security@quantumsequrity.com`.

---

4. Skip Catalogue

All 11,259 skips are documented in this section. Skips are emitted by the harness with a reason string captured in the log; the categories below aggregate them.

4.1 Bit-Oriented Hash Inputs (approximately 5,000 vectors)

ACVP defines hash inputs at bit granularity by way of the `msgLen` field; many vectors carry `msgLen` not a multiple of 8. The Rust `sha2` and `sha3` crates accept byte-aligned input through their stable APIs. The harness skips any vector where `msgLen % 8 != 0`.

Affected directories: SHA-2 / SHA-3 / SHAKE bit-oriented variant subsets. **Rationale**: QNSQY's CLI, GUI, and MCP boundaries accept whole-byte input only. A sub-byte feeder would not exercise any production code path. **Status**: Out of scope for the byte-aligned production API.

4.2 SP 800-56C TwoStep KDA-HKDF (2,000 vectors)

NIST SP 800-56C defines a TwoStep KDF (extract-then-expand with a distinct `fixedInfo` encoding) separate from RFC 5869 HKDF. QNSQY uses the RFC 5869 HKDF construction; its ACVP vectors all pass.

Status: Unused construction.

4.3 ML-KEM EncapsKeyCheck / DecapsKeyCheck (60 vectors)

FIPS 203 §7 defines `ML-KEM-EncapsKeyCheck` and `ML-KEM-DecapsKeyCheck` as key-validation functions. The `ml-kem 0.2.3` crate does not expose these as standalone functions; the crate's own `kem.rs:58` carries the comment "Should we verify that the provided `h` value is valid?", acknowledging the gap.

Status: Upstream Rust crate API gap. Tracked at `RustCrypto/KEMs`.

4.4 ML-DSA externalMu + Randomized (45 vectors)

The `ml-dsa 0.1.0-rc.7` crate exposes `raw_sign_mu(mu, rnd)` as `pub(crate)`, unreachable from outside the crate. The deterministic externalMu path (which uses the publicly exposed signing API with externalMu disabled and `rnd = 0`) passes; the externalMu+randomized variant requires an internal entry point.

Status: Upstream Rust crate API gap. Tracked at `RustCrypto/signatures`.

4.5 AES-GCM tagLen Below 96 Bits (30 vectors)

`aes-gcm 0.10` enforces the SP 800-38D §5.2.1.2 minimum tag-length floor (96 bits) at the type level. NIST publishes vectors at 32, 64, 96, 104, 112, 120, and 128-bit tag lengths. The four sub-96-bit variants are unreachable through the safe API. QNSQY uses 128-bit tags exclusively.

Status: Legacy IPsec/IKE configurations only. Not used by QNSQY. Compliant with NIST §5.2.1.2 guidance.

4.6 XECDH-keyVer Cross-Protocol (32 vectors)

XECDH-keyVer exercises behavior across X25519 / X448 boundaries. QNSQY uses X25519 only. The XECDH-SSC RFC 7748 self-consistency tests (50 pair tests) pass.

Status: Out of scope for X25519-only deployment.

4.7 Ed25519ph and Ed448 (19 vectors)

Pure Ed25519 (RFC 8032 with empty context) passes its ACVP vectors. Ed25519ph (pre-hashed Ed25519) and Ed448 are not used by QNSQY.

Status: Unused variants.

4.8 SHAKE MCT Variable-Output (2 vectors)

ACVP defines a Monte Carlo Test for SHAKE with variable output lengths drawn from a state machine. QNSQY uses single-output SHAKE, which the harness covers at 100% (SHAKE-128-FIPS202 269/269).

Status: Harness scope, not a correctness gap.

4.9 AES-192 (variable)

QNSQY uses AES-256 exclusively. Any vector group with `keyLen = 192` is skipped.

Status: Scope.

The skip categories sum to 11,259. Per-line skip reasons are recorded in `acvp-RUN-final.log` and are grep-able by the substring `SKIP:`.

5. Adjacent Validation

This report documents one evidence stream (ACVP test-vector replay). QNSQY runs four additional validation programs.

5.1 Curated KAT Suite

A 391-test curated Known Answer Test suite is part of QNSQY's test suite and runs on every test run. It includes 57 tests plus 525 ACVP JSON cases covering ML-KEM, ML-DSA (NIST ACVP), SHA-2 (FIPS 180-4), SHA-3 (FIPS 202), Ed25519 (RFC 8032), X25519 (RFC 7748), HMAC (RFC 4231), HKDF (RFC 5869), AES-GCM (SP 800-38D cases 15 / 17 / 18), XChaCha20 (draft-xchacha A.3.1), and Argon2id (RFC 9106 §5.3). Approximately 300 additional structural and round-trip tests cover SLH-DSA, HQC, Falcon, LMS, BLAKE3, and Shamir Secret Sharing.

5.2 Install and Functional Smoke Test

The May 2026 release was installed from its real package (`.deb` , `.rpm` , `.exe`) on 15 platforms (Ubuntu 22.04 through 25.10, Debian 12 and 13 trixie, Fedora 40 through 44, AlmaLinux 10, Windows 10 and 11 via Wine win10/win11). The Free-tier 11-test suite (password encrypt / decrypt, SHA round-trip, ML-DSA-44 keygen / sign / verify, ML-KEM-512 recipient round-trip, BLAKE3) passes on every platform. The Business-tier 8-test suite (FN-DSA-1024, HQC-256, LMS-SHA256-H10W2, SLH-DSA-128s, ML-KEM-1024) passes on every supported Linux platform. Not supported (glibc 2.34 or older): Debian 11, Rocky 9, AlmaLinux 9. macOS pipeline pending.

5.3 Fuzz Harnesses

`cargo-fuzz` harnesses in QNSQY's test suite target the data-format parsers (QSPG v2, QSDN v2, recipient slot block), the header decoder, and the password derivation path. They are exercised during pre-release manual sweeps. Continuous fuzzing on a public OSS-Fuzz hookup is on the roadmap.

5.4 Internal Multi-Agent Security Audits

QNSQY undergoes multi-agent internal security audits on each major release. Internal audit rounds since 2026-03 have covered crypto core, billing API, payment API, CLI, and infrastructure on a rolling basis. Per-finding details are tracked internally to avoid telegraphing partially mitigated weaknesses; cadence and scope are published in this report. Independent third-party audit is on the roadmap.

---

6. Limitations

The following limitations are stated for the avoidance of any inference that this report substitutes for them.

FIPS 140-3 / CMVP Validation: QNSQY is not FIPS 140-3 validated. QNSQY is not listed on the CMVP Validated Modules list. CMVP typically takes 6 to 18 months and costs tens of thousands of dollars through a NIST-accredited CSTL. CMVP is on the QNSQY roadmap; no application has been submitted to NIST as of the date on this document. The ACVP evidence in this document is the algorithm-correctness piece a CMVP lab would consult, and is not a substitute for the full CMVP process.

SOC 2, ISO 27001, HITRUST: QNSQY does not hold any of these organizational or operational certifications and has not pursued them as of the publication date. These are separate from cryptographic algorithm validation.

Real Coverage Gaps (135 vectors): 30 AES-GCM tagLen-below-96 vectors, 45 ML-DSA externalMu+randomized vectors, and 60 ML-KEM EncapsKeyCheck / DecapsKeyCheck vectors are real coverage gaps caused by upstream Rust crate API limits (described in detail in Section 4). A CMVP reviewer would treat these as open items, not as scope decisions. The 30 AES-GCM cases below the SP 800-38D §5.2.1.2 floor are a NIST safety floor and unlikely to ever be reached by a safe public API.

Independent Third-Party Audit: Not yet engaged. Internal multi-agent audit does not substitute for an independent firm.

Algorithm vs Deployment Correctness: ACVP confirms cryptographic primitives produce NIST's expected outputs. It does not prove the surrounding orchestration (key handling, file format, billing, MCP, GUI) is bug-free. The audit history in Section 5.4 covers that surface; the limitation is that it is internal.

7. References

7.1 NIST Standards

- NIST FIPS 203, "Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM)", August 2024. https://doi.org/10.6028/NIST.FIPS.203
- NIST FIPS 204, "Module-Lattice-Based Digital Signature Algorithm (ML-DSA)", August 2024. https://doi.org/10.6028/NIST.FIPS.204
- NIST FIPS 205, "Stateless Hash-Based Digital Signature Algorithm (SLH-DSA)", August 2024. https://doi.org/10.6028/NIST.FIPS.205
- NIST FIPS 180-4, "Secure Hash Standard (SHA-2)". https://doi.org/10.6028/NIST.FIPS.180-4
- NIST FIPS 202, "SHA-3 Standard". https://doi.org/10.6028/NIST.FIPS.202
- NIST FIPS 198-1, "Keyed-Hash Message Authentication Code (HMAC)". https://doi.org/10.6028/NIST.FIPS.198-1
- NIST SP 800-38D, "AES-GCM". https://doi.org/10.6028/NIST.SP.800-38D
- NIST SP 800-208, "Stateful Hash-Based Signatures". https://doi.org/10.6028/NIST.SP.800-208
- NIST Automated Cryptographic Validation Protocol Project. https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program
- NIST Cryptographic Module Validation Program (Validated Modules list). https://csrc.nist.gov/projects/cryptographic-module-validation-program

7.2 IETF Specifications

- RFC 7748, "Elliptic Curves for Security" (X25519, X448). https://datatracker.ietf.org/doc/html/rfc7748
- RFC 8032, "Edwards-Curve Digital Signature Algorithm (EdDSA)". https://datatracker.ietf.org/doc/html/rfc8032

- RFC 5869, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)". https://datatracker.ietf.org/doc/html/rfc5869
- RFC 4231, "Test Vectors for HMAC-SHA2". https://datatracker.ietf.org/doc/html/rfc4231
- RFC 9106, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications". https://datatracker.ietf.org/doc/html/rfc9106

7.3 Source Repositories

- usnistgov/ACVP-Server (canonical NIST vector tree; pinned commit `15c0f3deeefbfa8cb6cd32a99e1ca3b738c66bf0`). https://github.com/usnistgov/ACVP-Server
- QNSQY ACVP Reproducer Harness: https://quantumsequrity.com/validation-artifacts/qnsqy-acvp-harness-v1.0.tar.gz (standalone, self-contained; depends only on public RustCrypto crates; SHA-256 checksum at the same path with `.sha256` suffix)
- The harness archive contains the 12 loader modules and 50 test functions referenced throughout this report
- QNSQY product source is proprietary and not shared; the harness above is the algorithm-correctness reproducer used by NIST and external auditors

7.4 QNSQY Companion Materials

- Validation page: https://quantumsequrity.com/validation
- Raw sweep log: https://quantumsequrity.com/validation-artifacts/acvp-RUN-final.log
- Machine-readable summary: https://quantumsequrity.com/validation-artifacts/acvp-summary.json
- CSV summary: https://quantumsequrity.com/validation-artifacts/acvp-summary.csv
- Blog write-up: https://quantumsequrity.com/blog/nist-acvp-validation

---

8. Contact and Verification

For verification of the numbers in this report, vulnerability disclosure, methodology questions, or NIST coordination:

Purpose	Contact
Security correspondence	security@quantumsequrity.com
General inquiries	hello@quantumsequrity.com
Sales and procurement	sales@quantumsequrity.com
Security advisories	https://quantumsequrity.com/.well-known/security.txt
GPG public key	See `/.well-known/security.txt`

Security reports are acknowledged within 72 hours.

When corresponding with NIST or third-party auditors, please reference:

- Document version: 1.0

- Publication date: 2026-06-03
- Software: QNSQY, June 2026 build
- ACVP-Server commit: `15c0f3deeefbfa8cb6cd32a99e1ca3b738c66bf0` (v1.1.0.42)
- QNSQY ACVP Reproducer Harness: https://quantumsequrity.com/validation-artifacts/qnsqy-acvp-harness-v1.0.tar.gz

---

Signed for the publisher

Quantum Sequrity (QNSQY) security@quantumsequrity.com 2026-06-03

End of report.