§ INDEPENDENT REVIEW · RELAYSTR/NDK

Dart NDK — post-quantum and licensing review

relaystr · SDK · Dart · 5 findings

The short version

Licensing and interoperability defects, all fixed and released

The most serious finding is not cryptographic. An MIT-licensed SDK linked a GPL-3.0-only crate into every dependent application, whether or not that application used post-quantum code at all. Alongside it, the quantum-secure signer implemented the pre-standard round-3 CRYSTALS submission rather than FIPS 204, so the keys and signatures it produced could not be verified by any standards-compliant implementation. Two memory-safety issues in the Rust FFI and one key-recoverability gap were found while reading the same code. All five were fixed upstream and are now shipped: pub.dev serves ndk 0.9.0, published on 20 August, and we confirmed against that published archive that crystals-dilithium is gone, fips204 is in its place, the free path zeroizes before releasing, the buffer helper no longer reconstructs a Vec whose capacity it cannot vouch for, and signing keys derive from the BIP-39 seed.

What it gets right

The cryptography itself was not the problem

The most serious finding is a licensing defect. No key recovery and no forgery were found in the signing path, and the Schnorr verifier behaved correctly throughout.

An FFI boundary clean enough to fix in one pass

Replacing the signature scheme, closing two memory-safety defects and adding seed derivation all landed in a single reviewable change across ten files. A boundary that tangled would have made this three risky changes instead of one.

Zeroize was already wired in where it mattered most

The crate already pulled zeroize and used it on several paths. The finding is a buffer that path missed, not an absent practice.

Fixed and merged upstream in full

All five findings resolved by the maintainers, with the migration off the GPL-3.0-only dependency taken in one pass and the hybrid encryption work following it the same morning.

The fix that matters most

Replace crystals-dilithium with fips204

One dependency swap closes both the licensing defect and the interoperability defect. It removes GPL-3.0-only object code from an MIT SDK, and it moves the signer from the pre-standard round-3 CRYSTALS submission to FIPS 204 ML-DSA, so signatures can be verified by standards-compliant implementations. Merged: the crate manifest now reads fips204 0.4.6, which is MIT OR Apache-2.0, with no crystals-dilithium anywhere in the graph.

Findings

F1 · GPL-3.0-only crate linked into every dependent application

HIGH | LICENSING / COMPLIANCE | FIXED

`packages/ndk/rust` depended on `crystals-dilithium`, which is GPL-3.0-only, while NDK ships under MIT. The crate was not optional at build time: `build.dart` compiled it unconditionally, and the ordinary Schnorr verifier lived in the same `src/lib.rs` as the quantum-secure signer, so an application that never touched post-quantum code still linked the GPL-3.0-only object code. Statically linking GPL-3.0-only code makes the combined binary a derivative work, obliging every downstream application to distribute under GPL-3.0. That is a hard problem for closed-source consumers, and GPL-3.0 is also widely held to be incompatible with App Store distribution terms.

EVIDENCE, MEASURED RATHER THAN INFERRED

packages/ndk/rust depended on crystals-dilithium, which is GPL-3.0-only, while NDK ships under MIT. build.dart compiled it unconditionally, and the ordinary Schnorr verifier lived in the same src/lib.rs as the quantum-secure signer.

IMPACT

An application that never touched post-quantum code still linked GPL-3.0-only object code. Statically linking it makes the combined binary a derivative work, obliging every downstream application to distribute under GPL-3.0. That is a hard problem for closed-source consumers, and GPL-3.0 is widely held to be incompatible with App Store distribution terms.

FIX

Replace the dependency with `fips204`, which is MIT OR Apache-2.0 and imposes none of that. Merged upstream in pull request #712.

```
Fix: commit 4100dfa: https://github.com/relaystr/ndk/commit/4100dfa
Pull request #712: https://github.com/relaystr/ndk/pull/712
Released: ndk 0.9.0 on pub.dev: https://pub.dev/packages/ndk/versions/0.9.0
```

F2 · Pre-standard Dilithium shipped as the quantum-secure signer

MEDIUM | INTEROPERABILITY / COMPLIANCE | FIXED

`crystals-dilithium` implements the round-3 CRYSTALS submission. NIST changed the algorithm during standardisation, so Dilithium and ML-DSA are different schemes with different wire formats. Keys and signatures produced by the shipped code could therefore be verified only by the same code. No FIPS 204 implementation could read them, and vice versa. For a signature scheme whose whole purpose is letting somebody else check your work, that is a defect rather than a preference.

EVIDENCE, MEASURED RATHER THAN INFERRED

The signer implemented the round-3 CRYSTALS submission rather than FIPS 204.

IMPACT

NIST changed the algorithm during standardisation, so Dilithium and ML-DSA are different schemes with different wire formats. Keys and signatures produced here could not be verified by any standards-compliant implementation, which defeats the point of using a standard.

FIX

Move to FIPS 204 ML-DSA and pin a cross-implementation test vector. The fix includes one that checks an ML-DSA-87 public key byte for byte against an independent library from the same derived seed. Merged upstream in pull request #712.

```
Fix: commit 4100dfa: https://github.com/relaystr/ndk/commit/4100dfa
Pull request #712: https://github.com/relaystr/ndk/pull/712
Released: ndk 0.9.0 on pub.dev: https://pub.dev/packages/ndk/versions/0.9.0
```

F3 · Undefined behaviour in the buffer free path

MEDIUM | MEMORY SAFETY | FIXED

`write_buffer` leaked a `Vec` recording only its length, while `qs_free_buffer` reconstructed it with `Vec::from_raw_parts(data, len, len)`. That is undefined behaviour whenever capacity exceeds length. It held for every value passed at the time, so this was a latent trap rather than a live bug: the next contributor building an output with `push` or `extend` would have introduced heap corruption in the free path, with no compiler diagnostic to warn them.

EVIDENCE, MEASURED RATHER THAN INFERRED

The buffer free path reconstructed a Vec with Vec::from_raw_parts using a length and capacity that did not match the original allocation.

IMPACT

Undefined behaviour in the allocator. It may appear to work until it does not, and the failure surfaces far from the cause.

FIX

Use `into_boxed_slice`, which reallocates to the exact size so length and capacity always agree. Merged upstream in pull request #712.

```
Fix: commit 4100dfa: https://github.com/relaystr/ndk/commit/4100dfa
Pull request #712: https://github.com/relaystr/ndk/pull/712
Released: ndk 0.9.0 on pub.dev: https://pub.dev/packages/ndk/versions/0.9.0
```

F4 · Signing keys could not be restored from a mnemonic

MEDIUM | KEY MANAGEMENT | FIXED

Quantum-secure keys were random-only, with no derivation path, so losing the key lost the identity permanently. There was no way to restore a signer from the seed phrase the user already held.

EVIDENCE, MEASURED RATHER THAN INFERRED

The quantum-secure signing keys were generated independently of the account's seed phrase, with no derivation path and no export route.

IMPACT

A user who lost the device lost the identity, with a mnemonic in hand that could not restore it. Fixed by deriving the keys from the existing BIP-39 seed.

FIX

Derive from the 64-byte BIP-39 seed via HKDF-SHA256, making the post-quantum key a sibling of the secp256k1 key rather than a child, so breaking secp256k1 does not reach it and one mnemonic restores both. A 32-byte secp256k1 private key must be rejected as derivation input, since deriving from it would be circular. Merged upstream in pull request #712.

```
Fix: commit 4100dfa: https://github.com/relaystr/ndk/commit/4100dfa
Pull request #712: https://github.com/relaystr/ndk/pull/712
Released: ndk 0.9.0 on pub.dev: https://pub.dev/packages/ndk/versions/0.9.0
```

F5 · Freed secret-key buffers were not zeroized

LOW | ZEROIZATION | FIXED

`qs_free_buffer` deallocated without wiping. These buffers carry secret keys, and a freed but unwiped secret remains recoverable from a core dump or a swap file.

EVIDENCE, MEASURED RATHER THAN INFERRED

Secret-key buffers were freed without being zeroized first.

IMPACT

Key material stays in released memory until something else happens to overwrite it. Lower severity than the others here, and still the kind of thing a signing library is expected to get right.

FIX

Zeroize before deallocating. Merged upstream in pull request #712.

```
Fix: commit 4100dfa: https://github.com/relaystr/ndk/commit/4100dfa
Pull request #712: https://github.com/relaystr/ndk/pull/712
Released: ndk 0.9.0 on pub.dev: https://pub.dev/packages/ndk/versions/0.9.0
```

Automated scan (qScan v0.12.0)

PACKAGE	FILES SCANNED	FINDINGS
ndk 0.8.3 (pub.dev)	475 scanned, 448 analyzed	5 high: 4 ECDH in NIP-04, 1 secp256k1 dependency

Five results, and none of them is a defect in the SDK. Four are ECDH in the NIP-04 implementation and one is the secp256k1 dependency in the Rust crate. Nostr identity is secp256k1 by protocol definition and NIP-04 is a published NIP, so an SDK that implements the protocol correctly will show exactly this. What the scan does measure honestly is exposure: NIP-04 direct messages are encrypted with classical ECDH, and anyone recording relay traffic today can decrypt them once a cryptographically relevant quantum computer exists. That is harvest-now-decrypt-later against a public, permanently archived transport, and it is the reason the post-quantum work in this SDK matters. The readiness score of 56 reflects the protocol's position, not the quality of this implementation.

Merged upstream

Fix: commit 4100dfa: https://github.com/relaystr/ndk/commit/4100dfa

Pull request #712: https://github.com/relaystr/ndk/pull/712

Released: ndk 0.9.0 on pub.dev: https://pub.dev/packages/ndk/versions/0.9.0

Scope & method

Artifacts reviewed

relaystr/ndk, the Dart package and its Rust FFI crate under packages/ndk/rust

ndk 0.8.3 (pub.dev), and prereleases through 0.8.4-dev.11

Method

Source read of the Rust crate, the FFI boundary and the Dart bindings. Dependency and licence analysis across the crate graph, including the transitive licences a static link pulls in. Rust test suite and clippy, flutter analyze, and sign and verify exercised end to end through the real FFI rather than against mocks. qScan v0.12.0 across the published pub.dev package. Release state checked against pub.dev's published versions.

A review of the SDK. It does not cover applications built on it, the relays they talk to, or the Nostr protocol itself. The post-quantum direct-message work that followed in a second pull request was reviewed as a change but is not part of the findings here.

Reviewers

León Acosta