NovaKey Crypto Audit Appendix

Scope: iOS Application + Desktop Daemon
Audience: External security reviewers, auditors, and App Store reviewers
Algorithms: ML-KEM-768 (Kyber768), HKDF-SHA256, XChaCha20-Poly1305

This document provides a complete, auditable description of NovaKey's cryptographic architecture and its concrete implementation across both the iOS application and the desktop daemon. All claims are grounded in source code and avoid overstatement.

1. System Overview

NovaKey is a local, device-to-device secure input system. The iOS application acts as a trusted input origin, while the desktop daemon performs session enforcement and OS-level input injection. All cryptographic operations use standard, peer-reviewed primitives.

2. Cryptographic Primitives

Post-quantum key establishment: ML-KEM-768 (Kyber768)
Key derivation: HKDF-SHA256 (domain separated)
Authenticated encryption: XChaCha20-Poly1305 (192-bit nonce)
Replay protection: Nonce cache scoped per device and session

3. iOS Application Responsibilities

The iOS application is responsible for user-visible trust decisions and message framing. It never implements cryptographic primitives directly. Instead, it invokes native crypto modules to perform ML-KEM and AEAD operations.

Key properties:

- Explicit user approval during pairing
- No plaintext keystrokes or clipboard contents persisted
- No cloud key escrow or analytics tied to cryptographic material

4. Desktop Daemon Responsibilities

The desktop daemon is the cryptographic authority for active sessions. It performs post-quantum decapsulation, session key derivation, authenticated decryption, replay detection, and policy gating prior to OS-level input injection.

Anchored implementation points:

- ML-KEM-768 decapsulation: pairing_proto.go:145
- HKDF-SHA256 AEAD key derivation: crypto.go:153–156
- XChaCha20-Poly1305 AEAD (NewX): crypto.go:254
- Nonce parsing + authentication: crypto.go:260–268
- Replay cache enforcement: crypto.go:56, 323–328

5. End-to-End Protocol Flow

1) Pairing: iOS scans a QR code containing server address and Kyber768 public key. User explicitly approves pairing.
2) KEM: The daemon decapsulates the ML-KEM ciphertext to obtain a shared secret.
3) KDF: Session keys are derived using HKDF-SHA256 with a protocol-specific context string.
4) Transport: Each message is encrypted with XChaCha20-Poly1305 using a fresh random nonce.
5) Verification: The daemon authenticates, checks replay cache, and applies policy gates.

6. Security Properties

• Confidentiality and integrity for all messages in transit
• Post-quantum resistance for session establishment
• Forward secrecy at the session level
• Strong replay resistance
• No reliance on cloud services or third-party trust

7. QR Pairing Sub-Diagram

QR Code Contents (conceptual):

```
{
  server_url,
  device_id,
  kyber768_public_key (base64)
}
```

Flow:

• User scans QR code on desktop daemon
• iOS app displays host identity and requires explicit approval
• QR data is used only for pairing bootstrap; no secrets are encoded
• ML-KEM handshake completes over the secure channel

8. Explicit Non-Goals

NovaKey does not attempt to defend against a fully compromised host OS, malicious kernel drivers, or physical device compromise. These are explicitly out of scope.