A Formally Verified, Crash-Recoverable State Machine for Zero-Downtime Post-Quantum Cryptographic Key Migration

Thomas W. Rodriguez Jr.
Protocol Architect
trodriguez@higgaion.io | higgaion.io | Source & Proofs

Abstract

The transition from classical elliptic-curve cryptography to Post-Quantum Cryptography (PQC) presents a critical operational vulnerability for distributed networks and institutional custody providers. Existing migration strategies often enforce conjunctive multi-signature requirements that risk network partitions. Alternatively, they rely on irreversible state transitions that lack crash-recovery assurances. We introduce a formally verified 4-state migration engine designed to execute zero-downtime, uncoordinated cryptographic upgrades. By implementing a Disjunctive (OR-mode) Hybrid Verification protocol combined with an inverted "Erasure-Before-WAL" write-ahead logging architecture, the engine guarantees deterministic crash recovery without risking the resurrection of classical key material. The safety, liveness, and memory-safety invariants of this engine have been mathematically verified using a quad-tier formal framework consisting of 101 Coq (Gallina) mechanized proofs, TLA+ model checking, Frama-C WP, and CBMC bounded model checking.

1. Introduction & The "Store Now, Decrypt Later" Threat

The enactment of the National Security Memorandum on Promoting United States Leadership in Quantum Computing (NSM-10) and the release of the Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) mandate that critical infrastructure migrate to PQC algorithms by 2030. For institutional asset custodians and high-value data repositories, the threat model of "Store Now, Decrypt Later" necessitates immediate active migration.

However, the migration of root key material in a distributed multi-node infrastructure introduces severe operational risks. Coordinated upgrades across heterogeneous shards often result in split-brain consensus failures. Furthermore, the intermediate state of key transition (where classical keys must be irreversibly

destroyed to prevent future extraction) is highly vulnerable to hardware faults or power interruptions.

This paper details a state machine architecture that solves these vulnerabilities, offering a mathematically proven path for safe PQC transition.

1.1 Threat Model

We assume a computationally bounded adversary capable of network partitioning, arbitrary packet delay, and inducing targeted hardware faults (e.g., power crashes) to manipulate state transitions. The adversary possesses the capability to record classical ciphertexts (evaluating the Store Now, Decrypt Later threat) and seeks to recover the classical private key through crash-induced memory extraction during an incomplete migration. We assume the system is free from underlying hardware side-channels that bypass explicit memory-zeroing (`OPENSSL_cleanse`), and that the OS-provided RNG is cryptographically secure.

2. Disjunctive Hybrid Verification (Zero-Downtime)

Traditional hybrid cryptography enforces a conjunctive (AND-mode) verification policy, requiring both the classical signature (e.g., secp256k1, Ed25519) and the PQC signature (e.g., ML-DSA-87) to be verified. In a distributed network, this requires all validating nodes to be upgraded simultaneously: a logistical impossibility for decentralized systems.

Our architecture introduces **Disjunctive (OR-mode) Hybrid Verification**. The engine generates a composite dual-signature containing both classical and PQC components. * **Legacy Nodes:** Continue to verify only the classical component of the signature. * **Upgraded Nodes:** Verify the ML-DSA-87 PQC signature.

This policy decouples the key-generation timeline from the network-upgrade timeline, permitting uncoordinated, rolling migrations with zero operational downtime.

OR-Mode State Transition Diagram

```
stateDiagram-v2
    direction LR
    CLASSICAL --> HYBRID : Initiate Transition
    HYBRID --> FINALIZING : Erase & Backup
    FINALIZING --> PQC_ONLY : Commit PQC Record
```

3. Erasure-Before-WAL: Deterministic Crash Recovery

During the final phase of migration, the classical private key must be destroyed to eliminate the SNDL liability. Standard database Write-Ahead Logging (WAL)

protocols dictate that the state transition be written to disk *before* executing the action.

If applied to key migration, standard WAL creates a critical vulnerability: if a crash occurs after the WAL commit but before classical key erasure, the recovery routine could unwittingly resurrect the classical key from system paging limits or WAL recovery payloads, thereby violating the primary security invariant.

To mathematically eliminate this risk, we invert the protocol: 1. **Backup** the classical public key (for ongoing verification needs). 2. **Execute** zero-pass overwrites (`OPENSSL_cleanse()`) on the classical private key memory pages. 3. **Destroy** the classical cryptographic handle context. 4. **Append** the `PQC_ONLY` transition record to the WAL.

If the node suffers a catastrophic power failure between step 2 and step 4, the WAL replays only up to the `FINALIZING` state during recovery. Because the PQC keypair has already been persisted to the dual-compromise keystore, the classical key remains definitively erased, and the node safely resumes the finalization routine.

Erasure-Before-WAL Flow

```
sequenceDiagram
    participant Mem as Memory
    participant WAL as Write-Ahead Log
    participant Disk as Storage
    Mem->>Disk: 1. Backup Classical PubKey
    Mem->>Mem: 2. OPENSSL_cleanse(PrivKey)
    Mem-->>WAL: 3. Append PQC_ONLY Record
    WAL->>Disk: 4. fsync()
```

4. Quad-Tier Formal Verification

In infrastructure managing institutional assets, empirical testing is insufficient. The migration invariants are proven using a quad-tier formal verification pipeline:

1. **Deductive Verification (Coq):** 89 Gallina lemmas and 12 core safety theorems (101 total proofs) rigorously prove the transition matrix invariants and keystore derivation formulas, compiling with zero admitted lemmas.
2. **Model Checking (TLA+):** Proves liveness and absence of deadlocks across heterogeneous shard states, ensuring no split-brain consensus derivations occur during the OR-mode propagation.
3. **Bounded Model Checking (CBMC):** Analyzes the C implementation loops and pointer arithmetic, mathematically proving the absence of memory leaks and undefined behavior up to an unwind depth of 25.
4. **Frama-C WP:** Ensures rigorous separation of classical and PQC memory domains.

5. Security & Persistence Model

The keystore persistence mechanism utilizes a PQC-hybrid encryption scheme to prevent offline decryption attacks. Stored key material is encrypted utilizing a derived AES-256-GCM key from the concatenation of a `PBKDF2-HMAC-SHA3-256` password derivative and a post-quantum `ML-KEM-1024` shared secret.

This **Dual-Compromise Resistance** requires an adversary to simultaneously break a high-entropy password dictionary attack and the Module-Lattice-Based Key Encapsulation mechanism.

5.1 Performance & Security Analysis

Benchmarks indicate that OR-mode verification adds negligible overhead during the `HYBRID` phase. Validating an ML-DSA-87 signature requires approximately ~140 microseconds on modern x86 hardware, which is roughly comparable to Ed25519 verification limits under load. The primary bottleneck is disk I/O during the Erasure-Before-WAL sequence. This inverted flow appends a single synchronous `fsync` latency (typically 2-5ms) to the migration's critical path. However, because this occurs on a per-node asynchronous basis without coordinated network locks, overall system throughput remains entirely unaffected. From a security standpoint, uncoordinated rolling upgrades reduce the attack surface for distributed denial-of-service (DDoS) during the migration window to zero.

5.2 Evaluation

The standalone cryptographic abstractions and test harnesses are open-sourced for peer evaluation. Native execution of the 26 integration roundtrip tests natively verifies the Disjunctive framework and hardware limits.

Metric	Measurement / Assertions
Total Migration Asserts	26 / 26 Hardware Pass
Test Suite Execution	397 ms
Codecov Function Coverage	100.0%
CBMC Pointer Checks	Passed (Unwind 25)

6. Conclusion

The described PQC Migration Engine provides a verifiable, deterministic path for critical infrastructure to comply with NSA CNSA 2.0 timelines without inducing operational instability. By combining disjunctive verification for network availability with inverted WAL mechanics for absolute key destruction guarantees (and backing these algorithms with mechanized Coq proofs), we establish a robust standard for institutional cryptographic upgrades.

Legal Notice & Intellectual Property The overarching PQC crash-recoverable migration state machine, the Erasure-Before-WAL crash recovery architecture, and the Disjunctive (OR-Mode) Hybrid Verification protocols described herein are protected under **U.S. Patent Application No. 64/000,480**. The cryptographic primitives and mechanised proofs are open-sourced under the MIT License to encourage peer review and academic evaluation. For commercial implementation and Enterprise SDK licensing, please contact the authors.

7. References & Related Work

1. National Security Agency (NSA). "Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) Cybersecurity Advisory." (2022).
2. NIST. "FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard." (2024).
3. NIST. "FIPS 204: Module-Lattice-Based Digital Signature Standard." (2024).
4. Shor, P. W. "Algorithms for quantum computation: discrete logarithms and factoring." Proceedings 35th Annual Symposium on Foundations of Computer Science (1994).
5. Mosca, M. "Cybersecurity in an era with quantum computers: will we be ready?" IEEE Security & Privacy (2018).
6. Bindel, N., et al. "Transitioning to a quantum-resistant public key infrastructure." Post-Quantum Cryptography (2017).
7. Bos, J., Costello, C., Naehrig, M., and Stebila, D. "Post-quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem." IEEE Symposium on Security and Privacy (2015).
8. Crockett, E., et al. "Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH" Cryptology ePrint Archive (2019).
9. Kampanakis, P., et al. "The Viability of Post-Quantum X.509 Certificates." IEEE TrustCom (2018).
10. Campagna, M., et al. "Quantum Safe Cryptography and Security; An introduction, benefits, enablers and challenges." ETSI White Paper (2015).
11. Mohan, C., et al. "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging." ACM TODS (1992).
12. Chlipala, A. "Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant." MIT Press (2013).
13. Lamport, L. "Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers." Addison-Wesley (2002).
14. Cuoq, P., et al. "Frama-C: A Software Analysis Perspective." Software Engineering and Formal Methods (2012).
15. Clarke, E., et al. "A tool for checking ANSI-C programs." TACAS (2004).
16. Boneh, D., and Zhandry, M. "Secure Signatures and Chosen Ciphertext Security in a Quantum Computing World." CRYPTO (2013).
17. Koblitz, N., and Menezes, A. J. "A survey of public-key cryptosystems." ACM Computing Surveys (1997).

18. Bernstein, D. J., et al. “Post-quantum RSA.” Post-Quantum Cryptography (2017).
19. Bos, J., et al. “CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM.” IEEE European Symposium on Security and Privacy (2018).
20. Ducas, L., et al. “CRYSTALS - Dilithium: A Lattice-Based Digital Signature Scheme.” IACR Transactions on Cryptographic Hardware and Embedded Systems (2018).