EchoPulse – Chronological Table of Contents (v1.0)

🔷 Foundation & Overview

🧠 KEM Core Design & Cryptographic Logic

→ Symbolic state graph `G(V,E)`, transition function δ

8. `2Pulse.Hash.Input.docx`

→ Hash input examples and full key derivation chains

9. `3.1Echo.Critical.Improvements.docx`

→ Security enhancements: entropy model, mutation locking, HMAC

10. `3Echo.Key.Derivation.docx`

→ Formal key generation model `K = SHA3(v_enc || r)`

🔄 Mutation, Symbolics & Structural Behavior

11. `3Pulse.Failure.CaseHandling.docx`

→ Failure recovery paths, deterministic fallback logic

12. `4.1Pulse.Key.Derivation.Path.docx`

→ Mermaid-based visual path diagram

13. `4.2Pulse.TestVecotr.ExportTable.docx`

→ Test vector export table with SK/r/K flows

14. `4.3Pulse.Golden.TestVector.docx`

→ Golden vector for full E2E reproduction

15. `4Echo.Path.Structure.docx`

→ Public/private key path mechanics in graph space

16. `4Pulse.Hash.Function.docx`

→ Details on SHA3 usage and constant-time hashing interface

🧪 Testing, Mutation, and Threat Modeling

17. `5Echo.Mutation.Framework.docx`

→ Graph mutation model `µ(G)`, three mutation types

18. `5Pulse.Struct.rs.docx`

→ Core Rust struct definition for EchoGraph, SK, PK

19. `6.1Echo.Critical.Enhancements.docx`

→ Patch 6.2: Forward secrecy, HMAC locking, mutation fallback

20. `6Echo.Formal.KEM.Definition.docx`

→ KEM formal operations: KeyGen, Encaps, Decaps

21. `6Pulse.Keygen.rs.docx`

→ Rust-based key generation implementation

22. `7Echo.Resource.Modeling.docx`

→ RAM/ROM profiling and device class recommendations

📊 Encapsulation, Benchmarks & Protocol Framing

23. `7Pulse.encaps.rs.docx`

→ Rust-based encapsulation logic

24. `8Echo.Symbol.Mapping.docx`

→ Formal symbolic alphabet Σ and transition behavior

25. `8Pulse.decaps.rs.docx`

→ Rust-based decapsulation code

26. `9.1Pulse.EchoTest.golde.rs.docx`

→ Golden test implementation in Rust

27. `9.2Echo.Protocool.Enhancements.docx`

→ Protocol patching: CBOR, session drift, resilience layers

28. `9Echo.Threat.Modelling.docx`

→ Symbolic threat model, attack surface analysis

🔒 Security, Comparison & Finalization

29. `10Echo.Security.Analysis.docx`

→ IND-CCA analysis (ROM), entropy, mutation unpredictability

30. `10Pulse.Benchmark.Profile.docx`

→ Full benchmark profile (Encaps, Decaps, Mutation)

31. `11Echo.Protocoll.ComparisonTable.docx`

→ Comparison vs Kyber512 and FrodoKEM640

32. `11Pulse.Benchmark.csv.docx`

→ CSV-based cycle-level benchmark data

⚙ Advanced Features, Strategic Optimizations

33. `12.4.Echo.Strategic.Enhancments.docx`

→ SPOR, MIV, constant-time transition planning

34. `12Echo.Implementation.Notes.docx`

→ Developer notes for practical integration

35. `12Pulse.Ram.Usage.docx`

→ RAM breakdown per function/module

36. `13Pulse.Mutation.Overhead.docx`

→ Timing & memory cost of mutations (µs per op)

37. `14.1Pulse.Benchmark.Enhancments.docx`

   → Performance distribution analysis (SHA3 vs δ vs µ)

38. `14Pulse.Benchmark.Script.py.docx`

   → Python benchmark simulation + plot generation

🧠 Strategic Evaluation, Compatibility & Gemini Enhancements

39. `15EchoPulse.vs.NIST.PQC.KEMS.Perfoamance.docx`

   → EchoPulse vs NIST-KEMs in size, speed, resistance

40. `16Echo.Protocoll.Structure.docx`

   → Formal protocol struct, ideal for external reviewers

41. `17EchoPulse.Hardware.Mapping.Concepts.docx`

   → Mapping to M0+/RISC-V/FPGAs, hardware paths

42. `18_EchoPulse_Security_Considerations_Verification.docx`

   → Open verification roadmap, review methodology

43. `19EchoPulse.TargetAssumption.docx`

   → SGPU security assumption + IND-CCA ROM sketch

44. `20EchoPulse.SHA3.Bottleneck.docx`

→ Bottleneck mitigation via BLAKE2s or prehashing

45. `21EchoPulseNIST.Compatible.docx`

→ Hybrid-KEM, TLS entropy integration, algebraic bridge concepts

---