Code-Snippet

````
# EchoPulse Strategic Enhancements — Patch 12.4 **Version:** 12.4
**Date:** May 11, 2025 **Author:** EchoPulse Initiative This document
details formal and implementation-level enhancements to the EchoPulse
protocol, addressing advanced audit feedback concerning files 10–12 of
the EchoPulse dossier. ## 1. Security Model Declaration EchoPulse
security assumptions are grounded in the Random Oracle Model (ROM).
Specifically, the cryptographic hash function $H$ used in the shared
secret derivation, $K = H(v || r)$, must be modeled as a random oracle.
This means that $H$ is treated as an idealized function that produces
random outputs for each unique input. The formal IND-CCA security
target for EchoPulse applies to adversaries with the following
capabilities: * **Partial Graph Knowledge:** The adversary may possess
partial information about the initial state transition graph $G(V, E)$
and the mutation function $\mu$. * **Chosen Ciphertext Access:** The
adversary has access to a decapsulation oracle, allowing them to
attempt decapsulation of arbitrary ciphertexts (except for the
challenge ciphertext in the IND-CCA game). ## 2. Fallback Scenarios in
Graph Mutation Implementations must provide a fallback mutation path to
ensure continued operation in cases where the `session_index` is lost,
corrupted, or maliciously tampered with. * **Fallback Mechanism:** A
recommended approach is to maintain the last-known-good state of the
mutation process (i.e., the graph state corresponding to the last
successfully processed session) and recompute graph mutations forward
from that point using a securely stored, locked-down seed value. *
**Optional Salt-Anchored Retry Logic:** Implementations may also
introduce salt-anchored retry logic, where a new mutation salt is
derived from a long-term secret and used to re-synchronize the graph
evolution after a detected desynchronization. ## 3. Symbolic Start
Point Variation To further mitigate the risk of pattern anchor
formation under long-term adversarial exposure, an option is added to
randomize the initial node $v_0$ across a permitted subset $V' \
subseteq V$ of the graph's vertex set $V$. * **Randomized Initial
Node:** The initial node $v_0$ for key generation can be varied across
sessions. A recommended approach is to rotate $v_0$ periodically (e.g.,
per epoch) or derive it deterministically from a combination of a
device identifier and an epoch counter: $v_0 = H(\text{device\_id} || \
text{epoch\_counter})$. ## 4. AI-Resistance Metrics (Comparison Table)
To facilitate a more quantitative comparison of EchoPulse's AI-
resistance properties with other KEMs, the following metrics are
introduced: * **Symbol Path Overlap Rate (SPOR):** Defined as the ratio
of shared symbols between symbol paths across different sessions to the
total number of symbols in a path. A lower SPOR indicates greater
resistance to pattern learning. ``` SPOR = (Number of Shared Symbols) /
(Total Symbols in Path) ``` * **Mutation Interval Variance (MIV):**
Quantifies the degree of change between successive graph states. A
higher MIV suggests a more unpredictable graph evolution, hindering the
construction of accurate predictive models. ``` MIV =
Variance(Graph_State_{i+1} - Graph_State_i) ``` * **Session Symbol
Drift:** Measures the rate at which the mapping of symbols to state
transitions changes across sessions. A higher symbol drift indicates
increased difficulty for an adversary to track symbol semantics over
time. ``` Session Symbol Drift = Δσ / Session ``` EchoPulse is designed
to exhibit a very low SPOR and a high entropy in its mutation schedule,
contributing to its enhanced AI-resistance compared to static-graph
KEMs. ## 5. Minimum Platform Definition For practical implementation
guidance, the following minimum platform requirements are defined: *
**Target Architecture:** ARM Cortex-M0+ or equivalent microcontroller
architecture. * **Minimum RAM:** 64 KB of Random Access Memory (RAM). *
**Floating Point Unit:** No Floating Point Unit (FPU) is required. *
**Preferred Stack Depth:** 2.5 KB of stack space for a complete key
exchange cycle. * **Graph Mutation Cache:** 6–8 KB of temporary buffer
space for efficient graph mutation management (e.g., for storing graph
deltas or precomputed graph slices). ## 6. Constant-Time Transition
Technique The state transition function $\delta(v, \sigma)$ must be
implemented to resolve in a uniform amount of time across all possible
inputs (i.e., for any state $v \in V$ and symbol $\sigma \in \Sigma$).
* **Fixed-Length Jump Tables:** A recommended technique is to store the
transition function in fixed-length jump tables, where the next state
````