Code-Snippet

```
# EchoPulse Protocol Enhancements — Patch 9.3 **Version:** 9.3
**Date:** May 11, 2025 **Author:** EchoPulse Initiative This document
details formal and implementation-level refinements to the EchoPulse
protocol, addressing issues identified during an expert review of files
7 through 9 within the EchoPulse dossier. ## 1. Formal Model
Declaration (KEM Definition) EchoPulse assumes the Random Oracle Model
(ROM) for security proofs involving the cryptographic hash function
$H(v || r)$. This means that $H$ is modeled as a function that, when
given an input, returns a random output from its range, and that all
parties, including the adversary, have access to this function. Key
correctness, where the derived shared secret $K$ equals $K'$, relies on
two crucial conditions: * **Mutation Synchronization:** The sender and
receiver must maintain perfectly synchronized state transition graphs
$G(V, E)$ through the deterministic mutation function $\mu$. *
**Uniformly Distributed r:** The random symbol sequence $r$ used during
encapsulation must be drawn from a uniform distribution over the symbol
set $\Sigma$. ## 2. Symbol Bias & Adversarial Reuse Protection The
random symbol sequence $r$ must be sampled using cryptographically
secure, high-entropy sources to ensure a near-uniform distribution
across $\Sigma$. * **Entropy Enforcement:** Optionally, implementations
may employ rejection sampling or other entropy enforcement mechanisms
during key generation and encapsulation to guarantee that the generated
symbol sequences meet a minimum entropy threshold. If $r$ exhibits any
statistical bias, it could expose frequency-based vulnerabilities,
particularly under adaptive chosen-ciphertext attacks where an
adversary can exploit the non-uniformity of symbol usage to gain
information about the graph structure or key paths. ## 3. Encapsulation
Failure Handling Implementations should include a robust fallback
mechanism for handling decapsulation failures, which might occur due to
transient graph desynchronization or subtle session mismatch. *
**Fallback Mechanism:** If decapsulation fails, the sender should
derive a fresh random symbol sequence $r'$ using a deterministic
function that incorporates the session index and a retry counter: ```
r' = H(seed || session_index || retry_count) ``` where `seed` is a
long-term secret shared between the sender and receiver, and
`retry_count` is incremented for each failed encapsulation attempt. To
prevent denial-of-service or misuse, encapsulation attempts should be
logged and rate-limited. ## 4. Stack & Interrupt Safety (Resource
Layer) For EchoPulse implementations on real-time operating systems
(RTOS) or interrupt-driven microcontrollers (MCUs), special care must
be taken to ensure stack and interrupt safety. * **Non-Preemptible
Memory Blocks:** The code implementing the state transition function $\
delta$ and related graph traversal logic should be allocated in non-
preemptible memory blocks or protected by appropriate synchronization
primitives (e.g., disabling interrupts) to prevent data corruption or
inconsistent graph states during concurrent operations. * **Sufficient
Stack Space:** Systems that might handle parallel $\delta$ operations
(e.g., due to concurrent connections or multi-threading) should
allocate a minimum of 2 KB of stack space per execution context to
accommodate the function call depth and local variables required for
graph traversal. ## 5. Memory Isolation Between PK and Graph To enhance
security and prevent potential attacks that might exploit memory
corruption or out-of-bounds access, strict memory isolation should be
enforced between the keying material and the graph representation. *
**Separate Memory Regions:** The secret key ($SK$), public key ($PK$),
and the state transition graph $G(V, E)$ must reside in distinct, non-
overlapping memory regions. * **Memory Protection Unit (Optional):** If
the target system provides a Memory Protection Unit (MPU) or Memory
Management Unit (MMU), it is strongly recommended to utilize these
hardware features to enforce memory access control and prevent
unauthorized access between the key storage and the graph data. ## 6.
CBOR Encoding Format for EchoPulse Keys For interoperability and
efficient serialization, we define a recommended CBOR (Concise Binary
Object Representation) encoding format for EchoPulse keys:
```

EchoKey ::= {

0: byte_string, ; PK or SK symbol sequence

1: uint, ; path length