EchoPulse Threat Modeling

1. Assumptions & Attack Surface

This document outlines potential attack vectors against the EchoPulse Key Encapsulation Mechanism (KEM) and analyzes their feasibility based on the protocol's design. We assume a standard cryptographic adversary model where the attacker has the following capabilities:

1. **Passive Observation:** The attacker can intercept and record public keys (PK) transmitted by the receiver and ciphertexts (r) transmitted by the sender during encapsulation.
2. **Active Participation:** The attacker can act as a sender, initiating encapsulation processes with legitimate receivers using their observed public keys.
3. **Reverse Engineering Attempts:** The attacker may attempt to reverse-engineer the state transition function $\delta: V \times \Sigma \to V$ and the underlying structure of the state transition graph G(V,E).
4. **Local Access & Side Channels:** In certain deployment scenarios, the attacker may gain local access to devices implementing EchoPulse, enabling them to attempt timing attacks and power analysis side-channel attacks.

The primary attack surface consists of the transmitted public keys, the ciphertexts, and the internal operations of devices implementing the protocol.

2. Symbol Path Inference

A potential attack vector involves attempting to infer the secret key SK from the publicly known PK by analyzing the symbolic paths within the graph originating from the initial state v0. This could involve searching for overlaps in the symbol sequences or applying statistical analysis to observed PK values over multiple sessions, assuming a static or slowly evolving graph.

However, the EchoPulse design incorporates several countermeasures:

5. **Independent or One-Way PK Derivation:** If PK is derived from SK, the derivation process must be a computationally irreversible one-way function, preventing the direct computation of SK from PK. If PK is generated independently, there is no direct mathematical relationship to exploit.
6. **Graph Mutation:** The dynamic mutation of the graph G(V,E) across sessions, driven by the shared salt and session index, significantly limits the value of statistical analysis over time. The relationship between symbol sequences and resulting states changes with each mutation.
7. **Sufficient Path Entropy:** The length of the symbol sequences in SK and PK, coupled with the branching factor of the graph, is designed to provide sufficient entropy, making brute-force searching of the key space computationally infeasible.

3. Graph Structure Analysis

An attacker might attempt to reconstruct the structure of the state transition graph G(V,E) by observing the outcomes of encapsulation processes with a known PK and self-chosen random symbol sequences r. By analyzing the resulting shared secrets, the attacker might try to deduce the connectivity and transition rules of the graph.

The mutation framework is the primary defense against this attack:

8. **Dynamic Graph:** The continuous evolution of the graph structure through the deterministic mutation function μ ensures that any static model of G(V,E) built by the attacker will become invalid in subsequent sessions.
9. **Complexity of Mutation:** A well-designed mutation function will introduce sufficient complexity to prevent an attacker from easily predicting the graph transformations based on observed sessions.

4. Replay & Session Reuse Attacks

An attacker could intercept a valid ciphertext r from a previous session and attempt to use it to derive a shared secret with the receiver in a later session. This would be a replay attack. Similarly, an attacker might try to reuse a captured ciphertext within the same session multiple times.

EchoPulse incorporates the following mitigation:

10. **Session-Indexed Mutation:** The state transition graph G(V,E) is unique for each session due to the deterministic mutation based on the session index. A ciphertext r captured in one session will be processed on a different graph in a subsequent session, leading to a different final state vdec and thus a different (incorrect) derived key K′.
11. **Implicit Nonce in Ciphertext:** The fresh randomness of the symbol sequence r generated for each encapsulation acts as a nonce, preventing the reuse of the same ciphertext within a single session from yielding the same shared secret if the underlying graph traversal leads to different states due to internal dynamics or implementation details.

5. Chosen-Ciphertext Attack (CCA)

In a CCA scenario, an attacker might try to choose specific ciphertext symbol sequences r to feed into the decapsulation process (if they had access to a decapsulation oracle) to gain information about the secret key SK or the graph structure. The goal could be to force the decapsulation process into degenerate states or to reveal information about the transition function δ.

EchoPulse's design aims to resist this through:

12. **Complexity of δ Structure:** A well-designed transition function δ will map inputs (state and symbol) to seemingly pseudorandom next states, making it difficult to predict the outcome of chosen symbol sequences or to force specific state evolutions.
13. **Hashing of Final State and Ciphertext:** The shared secret K′ is derived by hashing the final state vdec *and* the ciphertext r. This means that even if an attacker could manipulate the final state, they would also need to control the hash of that state combined with their chosen r to gain useful information about a target key from a different encapsulation.
14. **Lack of Invertible δ:** The design principle of a transition function that is hard to reverse (finding u given $\delta(u,\sigma)=v$) makes it difficult for an attacker to work backward from a chosen final state to deduce