EchoPulse State Graph Design

1. Graph Parameters

The EchoPulse Key Encapsulation Mechanism (KEM) utilizes a finite, directed graph G(V,E) as its core state transition structure. The graph consists of $|V|=1024$ distinct vertices, each representing a unique state within the system. The set of directed edges E connects these vertices, with each edge $e=(u,v)\in E$ originating from a state $u\in V$ and terminating at a state $v\in V$. Each edge is uniquely labeled with a symbol σ drawn from the symbol set Σ, where $|\Sigma|=256$. The out-degree of each vertex, defined as the number of outgoing edges, is constrained to be within the range of 16 to 32. This range aims to balance the graph's connectivity for efficient traversal and mixing of states while maintaining a manageable storage footprint.

2. Graph Construction Methods

The state transition graph G(V,E) and the associated transition function $\delta:V\times\Sigma\to V$ are constructed deterministically. The following methods can be employed for generating the initial graph:

(a) Pseudorandom Generation via Master Seed: A cryptographically secure pseudorandom number generator (PRNG) is seeded with a master secret key. The PRNG is then used to determine the outgoing edges for each of the 1024 vertices. For each vertex $u\in V$, the PRNG generates a sequence of target vertices $v\in V$ for each symbol $\sigma\in\Sigma$. The number of outgoing edges from each node is constrained to the defined range (16-32). To ensure that each symbol in Σ labels at least one outgoing edge from each node (within the degree constraint), a specific assignment strategy is employed. For instance, the first 16 (or up to 32) symbols could be deterministically assigned to distinct target vertices generated by the PRNG for each source vertex.

(b) Deterministic Hash-Based Adjacency Creation: For each vertex $u\in V$ and each symbol $\sigma\in\Sigma$, the target vertex $v=\delta(u,\sigma)$ can be determined by a cryptographic hash function H′. The input to H′ could be a combination of the vertex index of u, the symbol σ, and a global initialization vector. The output of H′ is then mapped to a vertex index in the range [0,1023] using modulo operation or a similar deterministic mapping function. This method ensures that the graph structure is entirely determined by the initial parameters and the hash function.

(c) Optional Noise Injection for Entropy: To enhance the initial entropy of the graph structure, a limited amount of deterministic "noise" can be injected during the construction process. This could involve a small number of controlled swaps of edge targets or relabeling of edges based on a specific deterministic algorithm and the master seed. This aims to make the initial graph less predictable without compromising the deterministic nature required for consistent operation between parties.

The transition function $\delta:V\times\Sigma\to V$ is effectively represented by the set of labeled edges in G(V,E). This function can be stored in several ways, including:

1. **Transition Table:** A 2D array of size 1024×256, where the entry at index (u,σ) stores the target vertex $v=\delta(u,\sigma)$. This offers fast lookups but requires significant storage.
2. **Compressed Adjacency List:** For each vertex $u\in V$, a list of pairs (σ,v) is stored, representing the outgoing edges. This can be more memory-efficient, especially if the out-degree is significantly less than |Σ|. Given the defined out-degree constraint, this approach is likely more suitable for resource-constrained environments. The list can be sorted by symbol σ for efficient lookup.

3. Mutation Design (μ)

The mutation mechanism $\mu(G,salt,session\_index)\to G'$ deterministically alters the state transition graph G based on a shared secret salt and the current session index. The mutation process is designed to be reversible only by knowing the salt and the session index.

(a) Mutating Components: The mutation function can affect several components of the graph:

3. **Edge Relabeling:** The symbols labeling the existing edges can be permuted according to a deterministic function of the salt and session index.
4. **Edge Redirection:** The target vertices of existing edges can be changed to other vertices based on a deterministic algorithm driven by the salt and session index. This could involve a pseudorandom permutation of the vertices applied to the target of specific edges.
5. **Local Subgraph Rearrangement:** Small, localized subgraphs can be deterministically rearranged by altering the connections between a subset of vertices based on the salt and session index.

(b) Deterministic Mutation Triggering: Mutations are triggered based on the session index. For example, a mutation might occur every k sessions, where k is a predefined parameter. The specific parameters of the mutation (e.g., which edges are relabeled, which targets are changed) are determined by a deterministic function that takes the shared salt and the current session index as input. This ensures that both communicating parties apply the exact same sequence of mutations to their local copies of the graph.

(c) Graph Consistency: Consistency between the sender and receiver's graphs is maintained by the shared secret salt and the synchronized session index. Both parties start with the same initial graph (derived from the initial construction method). For each session, they independently apply the same deterministic mutation function based on the agreed-upon salt and the current session number. This ensures that at the beginning of each key exchange, both parties are operating on an identical version of the state transition graph.

4. Security Properties

The security of the EchoPulse KEM relies on several properties of the state transition graph:

6. **Collision Resistance in Path Space:** It should be computationally infeasible for an attacker to find two distinct sequences of symbols that, when applied to the public state vpub, lead to the same final state venc, unless they follow the intended path defined by the secret key. The graph structure should be designed to promote mixing and diffusion of states, making unintended collisions unlikely.
7. **Hardness of Reverse Traversal (No Efficient $\delta-1$):** Given a state v and a symbol σ, it should be computationally hard to determine the preceding state u such that $\delta(u,\sigma)=v$. The transition function should ideally not be easily invertible. The graph structure should avoid simple patterns that allow for efficient backtracking.