Symbol Path Coverage Testing for EchoPulse: Experimental Validation of Graph Mutation Dynamics

1. Introduction & Purpose

This document specifies a formal test framework for empirically validating the symbolic path coverage and mutation dynamics of the EchoPulse Key Encapsulation Mechanism (KEM). Given EchoPulse's unique non-algebraic design, demonstrating the robust and unpredictable nature of its graph transitions and mutation function (μ) is crucial for its security and suitability as a cryptographic primitive. The primary purpose of this testing is to show that repeated key encapsulations, across various sessions and over time, lead to high diversity in traversed graph paths, unique state visits, and symbol reuse patterns, thereby supporting the Symbolic Graph Path Unpredictability (SGPU) assumption.

2. Test Harness Description (Symbolic Graph Testing Architect)

The test harness will simulate multiple EchoPulse encapsulation sessions, systematically recording relevant graph traversal data.

Session-Based Testing Setup:

For N total test sessions:

1. **Initialization:**
   - Load the initial public parameters: graph $G0(V,E)$, starting state $v0$, and the deterministic mutation schedule μ.
   - Define a fixed cryptographic seed for each test run to ensure repeatability. This seed will derive all subsequent random values (e.g., payloads r) within that run.
2. **Session Loop (for** $i=1$ **to** N**):**
   - **Time Step Determination:** Each session i will be associated with a specific logical time step ti. This can either increment sequentially ($ti=i-1$) or be uniformly sampled within a defined range $[0,Tmax]$.
   - **Graph Mutation Application:** Apply the mutation function: $Gti=\mu(G0,ti)$. The actual graph G for this session is Gti.
   - **Payload Generation:** Generate a cryptographically random payload $ri \in R$.

    The randomness for ri should stem from the fixed test run seed.

- **Path Traversal:** Execute the EchoPulse encapsulation process:
    - Start at initial state v0 in graph Gti.
    - For each symbol sj in the payload ri (parsed into appropriate symbol length):
        - Compute vj+1=δ(vj,sj) using the current graph Gti.
        - Record the traversed edge (vj,vj+1) and the symbol sj.
    - The final state venc is the last state reached after processing the full payload.
- **Data Recording:** For each session i, record the following:
    - Session ID, Time Step (ti)
    - Initial Graph State (conceptually Gti)
    - Full Sequence of Traversed States (v0,v1,...,vfinal)
    - Full Sequence of Applied Symbols (s1,...,sk)
    - List of Unique Edges Traversed within this session
    - List of Unique States Visited within this session

Test Harness Structure (High-Level Logic):

```
FUNCTION RunEchoPulsePathCoverageTest(N_sessions, MaxTimeSteps, InitialSeed):
 SET GlobalSeed = InitialSeed
 SET MasterGraph = InitializeEchoPulseGraph() // G_0
 SET RecordedSessionData = []

 FOR session_id FROM 1 TO N_sessions:
  SET CurrentTime = (session_id - 1) % (MaxTimeSteps + 1) // Example: simple
sequential time
  // Optionally: CurrentTime = SampleUniform(0, MaxTimeSteps) // More varied time
steps

  SET MutatedGraph = ApplyMutation(MasterGraph, CurrentTime) // G_t = μ(G_0, t)

  SET RandomPayload = GenerateRandomPayload(GlobalSeed) // Deterministic
random from seed
```

```
  SET CurrentState = MasterGraph.InitialState
  SET TraversedStates = [CurrentState]
  SET TraversedEdges = []
  SET UsedSymbols = []

  FOR EACH Symbol IN ParsePayloadToSymbols(RandomPayload):
    SET NextState = MutatedGraph.Transition(CurrentState, Symbol)
    TraversedStates.Add(NextState)
    TraversedEdges.Add((CurrentState, NextState, Symbol))
    UsedSymbols.Add(Symbol)
    CurrentState = NextState

  RECORD {
    "session_id": session_id,
    "time_step": CurrentTime,
    "payload": RandomPayload,
    "traversed_states": TraversedStates,
    "traversed_edges": TraversedEdges,
    "used_symbols": UsedSymbols
  } INTO RecordedSessionData

 RETURN RecordedSessionData
END FUNCTION
```

3. Metric Definitions (Metric Designer for Symbolic Path Evaluation)

These metrics will be computed across all recorded sessions to empirically demonstrate path unpredictability and graph dynamism.

- **Unique State Visit Ratio (**UVRS**):**
  - **Definition:** The ratio of the total number of unique states visited across all N sessions to the total size of the state space $|V|$.
  - **Computation:** $UVRS=|V||\bigcup i=1N\{states\ in\ sessioni\}|$
  - **Why it Supports SGPU:** A high UVRS indicates that EchoPulse explores a large portion of the state space, making it harder for an adversary to precompute or restrict possible path outcomes. If the ratio is low, it suggests a predictable subset of states is often reached.
- **Unique Edge Traversal Ratio (**UVRE**):**

  - **Definition:** The ratio of the total number of unique edges traversed across all N sessions to the total number of possible edges in the original graph ($|V|\times|\Sigma|$). This can also be defined as the ratio to the total edges generated by all mutations encountered ($|E_{total}|$). We use the latter for more precise coverage.
  - **Computation:** $UVRE=|\bigcup_{t_i}E_{G_{t_i}}||\bigcup_{i=1}^{N}\{\text{edges in session}_i\}|$ (where $E_{G_{t_i}}$ are edges present in G_{t_i})
  - **Why it Supports SGPU:** A high UVRE demonstrates that not only states but also the specific transitions between states are highly varied, directly supporting the unpredictability of the exact path.
- **Symbol Reuse Frequency (SRF):**
  - **Definition:** For each symbol $s\in\Sigma$, its frequency of leading to the *same* next state from *different* initial states within a mutated graph, or its frequency of leading to *different* next states from the *same* initial state across different mutated graphs. More practically, calculate the average (or variance of) how often each symbol appears in the used_symbols lists.
  - **Computation:** Compute the empirical probability distribution $P(s)$ for each $s\in\Sigma$ over all used_symbols lists. Calculate the entropy of this distribution. Additionally, track (v,s,v') triplets: how often does (v,s) map to v' vs. v'' across different G_t.
  - **Why it Supports SGPU:** High variance in SRF for specific (state, symbol) pairs across mutations suggests a dynamic graph. A low variance or skewed SRF could indicate preferred symbols/transitions, making prediction easier. High overall symbol usage diversity (high entropy of $P(s)$) is good.
- **Mutation Diffusion Factor (MDF):**
  - **Definition:** Measures how much the graph structure changes between adjacent time steps, and how widely these changes affect potential paths.
  - **Computation:** For G_t and G_{t+1}:
    - **Edge Divergence:** $MDFE(t)=|E_{G_t}\cup E_{G_{t+1}}||\text{Edges in } G_t \Delta \text{ Edges in } G_{t+1}|$ (Symmetric Difference Ratio)
    - **Path Divergence (Heuristic):** For a randomly sampled subset of (v,s) pairs, calculate $\delta_{G_t}(v,s)$ and $\delta_{G_{t+1}}(v,s)$. $MDFP(t)$ is the proportion of pairs where δ produces different next states.
  - **Why it Supports SGPU:** A consistently high MDFE and MDFP ensures that knowing the graph at time t gives negligible advantage in predicting paths at time $t+k$ without observing μ. This directly supports the unpredictability aspect of SGPU related to the time component.

4. Experimental Design (Symbolic Graph Testing Architect)

To ensure comprehensive testing, the experiment should be parameterized.

- **Parameters:**
  - N: Total number of sessions (e.g., 105 to 107).
  - Lr: Length of the random payload r (determines path length, e.g., 256 bits for 32 symbols if each symbol is 8 bits).
  - Tmax: Maximum time step to simulate for mutation (t∈[0,Tmax]).
  - SG: Seed for initial graph generation (G0).
  - SM: Seed for mutation function μ (if μ uses randomness in its deterministic generation of changes).
  - SP: Seed for random payload generation.
  - |V|: Size of the state space.
  - |Σ|: Size of the symbol alphabet.
- **Procedure:**
  1. Initialize EchoPulse with (SG,SM).
  2. Run the RunEchoPulsePathCoverageTest function with N, Tmax, and SP.
  3. Collect RecordedSessionData.
  4. Process RecordedSessionData to compute UVRS, UVRE, SRF, and MDF.
  5. Repeat the entire test run with different SP to ensure results are not sensitive to a particular payload sequence.
  6. Repeat the entire test run with different SG and SM to confirm robustness across different graph and mutation instantiations.

5. Output Formats (Report Synthesizer)

The collected data and computed metrics should be output in formats suitable for analysis and visualization.

- **Raw Session Logs (JSON/CSV):**
  - Each row/entry represents a session.
  - Fields: session_id, time_step, payload_hash (to save space), traversed_states (list of integers), traversed_edges (list of tuples (v_start, v_end, symbol)).
  - Example CSV structure:

    Code-Snippet
    ```
    session_id,time_step,payload_hash,traversed_states,traversed_edges
    1,0,abc123def456,"[0,123,456,...]","[(0,123,s1),(123,456,s2),...]"
    2,1,ghi789jkl012,"[0,789,121,...]","[(0,789,s3),(789,121,s4),...]"
    ```

- **Aggregated Metrics (JSON/CSV):**

    - Summary of computed metrics.
    - Fields: N_sessions, MaxTimeSteps, UniqueStateVisitRatio, UniqueEdgeTraversalRatio, SymbolReuseFrequency_Distribution (e.g., as a dictionary or flattened list), MutationDiffusionFactor_Edge_Mean, MutationDiffusionFactor_Path_Mean, TotalUniqueStatesVisited, TotalUniqueEdgesTraversed.
    - Example JSON:

```json
{
  "N_sessions": 1000000,
  "MaxTimeSteps": 1000,
  "UniqueStateVisitRatio": 0.985,
  "UniqueEdgeTraversalRatio": 0.75,
  "SymbolReuseFrequency_Distribution": {"s0": 0.003, "s1": 0.004, ...},
  "MutationDiffusionFactor_Edge_Mean": 0.65,
  "MutationDiffusionFactor_Path_Mean": 0.70,
  "TotalUniqueStatesVisited": 1008,
  "TotalUniqueEdgesTraversed": 192000
}
```

- **Plot-Ready Logs:**
    - Time-series data for MDF or other metrics across different time steps for visualization.
    - Example: time_step,mdf_edge,mdf_path.

Export Format Recommendations:

- **Python:** Ideal for scripting the test harness, metric computation, and data analysis. Libraries like networkx for graph manipulation, numpy for numerical operations, and pandas for data handling are well-suited. Outputs can be easily generated in CSV or JSON.
- **Rust:** For high-performance simulation or production-grade test harnesses. Offers strong type safety and performance guarantees. Can easily output to CSV or JSON using crates like csv and serde_json.

6. Interpretation Guidelines (Report Synthesizer)

Interpreting the results requires careful consideration of the expected security properties of EchoPulse.

- **High Unique State/Edge Visit Ratios:** UVRS and UVRE values close to 1 (or 100%) indicate that the random payload sequences and graph mutations effectively explore the vast majority of the graph's states and transitions. This supports the notion that an adversary cannot easily predict or limit the potential paths.
- **Homogeneous Symbol Reuse Frequency:** A symbol reuse frequency distribution that is relatively flat (high entropy) indicates that symbols are used roughly equally often, preventing an adversary from focusing on high-frequency symbols to predict paths.
- **High Mutation Diffusion Factor (MDF):** Consistently high MDF values (e.g., above 0.5 for edge/path divergence) suggest that the graph structure changes significantly and unpredictably between time steps. This implies that pre-computing or memorizing previous graph states for long-term prediction is infeasible, directly bolstering the SGPU assumption. Low MDF values would indicate a predictable or stagnant mutation.
- **Correlation Analysis:** Investigate potential correlations between path length, payload content, and the type of states or edges visited. Ideally, there should be no easily discernible patterns that an adversary could exploit.
- **Reproducibility:** The ability to reproduce all test results using the specified seeds and parameters is paramount for validating the test framework and the observed dynamics.

This test specification provides a robust empirical framework to complement theoretical security proofs, offering valuable insights into the practical unpredictability and dynamism of EchoPulse's symbolic graph.
