EchoPulse Benchmark Profile (Document C1)

Version: 1.0

Date: May 11, 2025

Author: EchoPulse Initiative

This document outlines the methodology and metrics for profiling the resource utilization and performance characteristics of the EchoPulse Key Encapsulation Mechanism (KEM) system across a range of embedded and general-purpose computing platforms. The goal is to quantify the RAM footprint, CPU cycle consumption, stack usage, and timing behavior of the core EchoPulse operations: key generation, encapsulation, and decapsulation.

1. Purpose

The primary objectives of this benchmark profile are to:

* **Establish Test Metrics and Target Platforms:** Define the key performance indicators (KPIs) and identify the specific hardware platforms on which EchoPulse will be evaluated.

* **Quantify Resource Usage:** Accurately measure the RAM consumption, CPU cycle count, and stack depth required by the EchoPulse key encapsulation and decapsulation processes.

* **Characterize Timing Behavior:** Determine the execution time for each core operation to assess the performance envelope of the KEM on different hardware.

2. Target Platforms

The EchoPulse KEM will be profiled on the following target platforms to represent a spectrum of resource-constrained and more powerful environments:

* **Profile 1: Cortex-M0+:** Microcontroller with 48 KB of RAM and a clock speed of 16 MHz. This profile represents severely resource-constrained embedded systems.

* **Profile 2: Cortex-M4F:** Microcontroller with 96 KB of RAM and a clock speed of 80 MHz, including a Floating Point Unit (FPU). This profile represents more capable embedded devices.

* **Profile 3: RISC-V RV64:** Embedded processor with 256 KB of RAM and a clock speed of 120 MHz, utilizing a 64-bit RISC-V instruction set. This profile represents higher-performance embedded systems.

* **Optional: General-Purpose Laptop:** A standard x86_64 architecture laptop with a clock speed of approximately 2.0 GHz. This profile provides a baseline for performance in less constrained environments.

3. Core Metrics

The following core metrics will be measured and reported for each target platform and benchmark category:

* **Total RAM Usage:** The total amount of Random Access Memory (RAM) consumed by the EchoPulse implementation during each operation. This includes the memory occupied by the state transition graph $G(V, E)$, secret key (SK), random symbol sequence (r), input/output buffers, and any other runtime data structures. Measurement should be taken at the peak memory usage point during each operation.

* **Stack Depth per Function:** The maximum stack depth reached by each of the core functions: key generation (`echo_keygen`), encapsulation (`echo_encaps`), and decapsulation (`echo_decaps`). This metric is critical for ensuring stack overflow does not occur, especially on embedded systems with limited stack space.

* **CPU Cycles or Estimated Microseconds per Operation:** The number of CPU cycles consumed by each core operation. For platforms where direct cycle counting is feasible, this will be the primary metric. For others, the estimated execution time in microseconds will be derived using the platform's clock speed.

* **Mutation Cycle Overhead:** The additional RAM usage and CPU cycles/time required to perform a single graph mutation cycle ($\mu(G)$). This metric will assess the cost associated with the dynamic nature of the EchoPulse graph.

* **SHA3-256 Cost in µs:** The execution time specifically for the SHA3-256 hash operation, measured in microseconds. This isolates the performance of the cryptographic primitive.

4. Symbolic Input Parameters

To ensure consistent and comparable benchmark results, the following symbolic input parameters will be used across all target platforms:

* **Path Length (L):** The length of the secret key (SK) and the random symbol sequence (r) will be fixed at 28 symbols.

* **δ Transition Cost:** The cost (in CPU cycles or time) of a single state transition ($\delta(v, \sigma)$) will be measured by performing 100 sequential transitions and averaging the result. This helps characterize the performance of the graph traversal logic.

* **Hash Input Size:** The input to the SHA3-256 hash function will consistently be 30 bytes (2 bytes for v_{enc} and 28 bytes for r).

* **Mutation Type:** Benchmarks will be performed for two types of graph mutation:

   * **Full Row Rewrite:** Mutating a complete row of the transition table.

   * **Hash-Seeded Mutation:** Mutating a subset of the graph based on a hash of the current graph state and session index. The specific mutation algorithm will be documented alongside the benchmark results.

5. Benchmark Categories

The following core operational cycles of the EchoPulse KEM will be individually benchmarked:

* **Encapsulation Cycle:** This benchmark will measure the resources and time required for the complete encapsulation process, starting from the generation of a secret key (SK), deriving the public key (PK), generating the random symbol sequence (r), traversing the graph to the encapsulation state (v_{enc}), and finally computing the shared secret key (K).

* **Decapsulation Cycle:** This benchmark will measure the resources and time required for the complete decapsulation process, starting with the secret key (SK) and the received random symbol sequence (r), traversing the graph to the decapsulation state (v_{dec}), and computing the reconstructed shared secret key (K').

* **Mutation Event:** This benchmark will specifically measure the overhead associated with triggering and executing a single graph mutation event after a defined number of encapsulation cycles (e.g., 10).

6. Measurement Modes

Performance and resource usage will be assessed using the following measurement modes:

* **Simulated (Cycle Count Estimate via Model):** For embedded platforms, where direct cycle counting might be complex, cycle counts will be estimated using instruction set simulators or analytical models of the target processor.

* **Native Benchmark (Real Runtime using Rust `std::time`):** On platforms where it is feasible (including the optional general-purpose laptop), the `std::time` module in Rust will be used to measure the real-world execution time of each operation.

* **Symbolic Trace Profiler:** Optionally, a symbolic trace profiler can be used to analyze the number of symbols processed during graph traversals and the frequency of unique versus reused symbols. This can provide insights into potential areas for optimization.

7. Conclusion

This EchoPulse Benchmark Profile provides a structured framework for systematically evaluating the performance and resource consumption of the KEM across a range of target platforms. The defined metrics, input parameters, and benchmark categories will enable a comprehensive understanding of the system's operational characteristics in both resource-constrained and more scalable environments. The results of these benchmarks will be crucial for optimizing the EchoPulse protocol and its implementations for real-world deployment.

Document C1 — Execution Metrics Layer — EchoPulse Initiative