EchoPulse Resource Modeling

1. Target Platforms

This document analyzes the estimated computational and memory resources required to implement the EchoPulse Key Encapsulation Mechanism (KEM) on several representative target platforms with varying resource constraints:

1. **RISC-V Microcontroller (MCU):** Characterized by limited Random Access Memory (RAM) in the range of 64 KB and moderate clock speeds (e.g., tens of MHz).
2. **ARM Cortex-M0+ Microcontroller:** Similar to the RISC-V MCU in terms of resource constraints, typically featuring RAM sizes from a few KB up to 32 KB and low power consumption.
3. **Embedded Linux System:** Representing more resource-rich embedded devices with RAM sizes of 512 KB or greater and significantly higher clock speeds (e.g., hundreds of MHz).

2. Memory Requirements

The memory footprint of EchoPulse is primarily determined by the storage requirements of the state transition graph G(V,E), the private and public keys, and the temporary buffers needed during operation.

4. **Compressed Graph Storage:** Assuming 1024 vertices and an average out-degree of 24, storing each transition (σ,v) using a compressed representation (e.g., 1 byte for the symbol index and 2 bytes for the vertex index) would require approximately $1024\times24\times3=73728$ bytes (72 KB). More aggressive compression techniques, as discussed in Section 4, could potentially reduce this.
5. **Secret Key (SK) Storage:** With a private key length of $l=28$ symbols, and each symbol represented by 1 byte, the SK requires 28 bytes.
6. **Public Key (PK) Storage:** Assuming a public key length of $l'=28$ symbols (in the uncompressed case), the PK also requires 28 bytes. If a seed-based derivation is used, the storage could be smaller (e.g., size of the seed + derivation algorithm description).
7. **Payload Buffer (r):** The random symbol sequence r of length $m=28$ symbols requires a buffer of 28 bytes during encapsulation and decapsulation.
8. **Optional Temporary Graph State:** If the mutation is precomputed for a window of k sessions, additional memory might be needed to store these mutated graph states. This depends on the value of k and the size of the graph representation. For immediate mutation, this might be negligible.

Estimated Total Memory: Depending on the graph compression and PK representation, the total memory requirement could range from approximately 72 KB (with minimal compression and direct PK storage) down to potentially 30-40 KB with aggressive optimization. Precomputing multiple graph states for mutation would increase this further.

3. Computation Requirements

The primary computational costs in EchoPulse involve the evaluation of the transition function δ and the execution of the cryptographic hash function H.

- **Number of δ Evaluations:**
  - **Key Generation:**l evaluations to reach vpriv and l′ evaluations to reach vpub. Assuming $l\approx l'\approx28$, this is around 56 evaluations.
  - **Encapsulation:**l′ evaluations to reach vpub and m evaluations to reach venc. With $l'\approx m\approx28$, this is around 56 evaluations.
  - **Decapsulation:**l evaluations to reach vpriv and m evaluations to reach vdec. With $l\approx m\approx28$, this is around 56 evaluations.
- **Hash Function Calls:** One call to H (e.g., SHA-3) during both encapsulation and decapsulation. Key generation might involve hashing for seed expansion or key derivation, adding to the computational cost.
- **Estimated Cycles on Embedded Devices:** The number of clock cycles required for a single δ evaluation depends heavily on how the transition function is stored and accessed (e.g., lookup table vs. more complex logic). Assuming a relatively efficient lookup, each evaluation might take a few tens to hundreds of cycles on a low-power MCU. A SHA-3 hash operation can take millions of cycles on such devices.
  - **Encapsulation/Decapsulation (δ evaluations):**56×(tens to hundreds of cycles).
  - **Encapsulation/Decapsulation (SHA-3):** Millions of cycles.

  Therefore, a single encapsulation or decapsulation operation on a resource-constrained MCU could take tens to hundreds of millions of clock cycles.

4. Memory Optimization Options

Several techniques can be employed to reduce the memory footprint of the state transition graph:

9. **Symbol Mapping:** If the effective out-degree is significantly lower than $|\Sigma|=256$, a local mapping can be used to represent only the relevant symbols for each vertex, reducing the storage per transition.
10. **Delta Encoding:** If the target vertex indices for transitions from a given state exhibit patterns, delta encoding can compress the storage.
11. **Transition Matrix Pruning:** If the transition matrix (implicitly defined by δ) is sparse, storing only the non-default transitions can save significant memory.
12. **Key/Payload Length Tuning:** Reducing the lengths l,l′,m of the symbol sequences directly reduces the memory required for storing the keys and the payload, at a potential trade-off in security margin.
13. **Preloading Graph Slices:** For systems with slightly more RAM, preloading the relevant "slices" of the graph for the next k sessions based on the mutation schedule could improve performance by reducing access times.

5. Energy and Throughput

Estimating energy consumption is highly platform-dependent. However, the dominant factors will be the number of CPU cycles spent on δ evaluations and the hash function. SHA-3 operations are generally energy-intensive.