Proof of IND-CCA2 Security in the Random Oracle Model (ROM) under the Symbolic Graph Path Unpredictability (SGPU) Assumption

1. Introduction

This document provides a formal security proof sketch for the EchoPulse Key Encapsulation Mechanism (KEM) under the Indistinguishability under Chosen Ciphertext Attack (IND-CCA2) security notion. Given EchoPulse's non-algebraic, symbolic structure based on deterministic graph transitions and mutating path-based key generation, standard hardness assumptions like Learning With Errors (LWE) or Short Integer Solution (SIS) are not directly applicable. Instead, we propose a novel assumption: the Symbolic Graph Path Unpredictability (SGPU) assumption. The proof will be constructed within the widely accepted Random Oracle Model (ROM).

2. Threat Model

We consider a standard adversarial model where an adversary (denoted as A) is computationally bounded (probabilistic polynomial-time, PPT). The adversary's goal is to distinguish between two keys, one of which was genuinely encapsulated by the KEM, and the other being a randomly chosen key.

The adversary has access to specific oracles:

Encapsulation Oracle (OEncaps): Allows the adversary to request the encapsulation of a random or chosen message, receiving a valid ciphertext and the corresponding encapsulated key.
Decapsulation Oracle (ODecaps): Allows the adversary to query any ciphertext (except the challenge ciphertext) and receive the encapsulated key if the decapsulation is successful, or an error otherwise.
Hash Oracle (OH): Allows the adversary to query the hash function, modeled as a Random Oracle, and receive a random output for any input not previously queried, or the consistent output for repeated queries.
The adversary is considered successful if its advantage in distinguishing the challenge key from a random key is non-negligible.

3. Game Definition (Formal Cryptographer)

We define the IND-CCA2 security of EchoPulse via the following game GameEchoPulseIND-CCA2(A) played between a Challenger C and an Adversary A.

Setup:

C generates the public parameters: a finite symbol graph $G(V,E)$, a starting state $v0 \in V$, and a deterministic mutation schedule μ. These parameters are derived during a trusted setup phase.
C initializes a list Hlist to record queries to the Random Oracle H. Initially, Hlist is empty.
C sends $(G,v0,\mu)$ to A.
Phase 1 (Learning Phase):
A may make the following queries:
Hash Query OH(x):
If $(x,y) \in$ Hlist for some y, C returns y.
e, C chooses a random $y \leftarrow \{0,1\}256$, adds (x,y) to Hlist, and returns y.