```rust
// echo_encaps.rs

use rand::RngCore;
use sha3::{Digest, Sha3_256};
use crate::echo_struct::{Symbol, State, SymbolPath, EchoGraph};

pub struct EchoEncapsulation {
    pub r: SymbolPath,
    pub k: [u8; 32],
}

pub fn echo_encaps(graph: &EchoGraph, pk: &SymbolPath) -> EchoEncapsulation {
    let v_pub = graph.resolve(0, pk);
    let mut rng = rand::thread_rng();
    let r_symbols: Vec<Symbol> = (0..28).map(|_| rng.next_u8()).collect();
    let r = SymbolPath { symbols: r_symbols };

    let v_enc = graph.resolve(v_pub, &r);

    let v_enc_bytes = [(v_enc & 0xFF) as u8, (v_enc >> 8) as u8];

    let mut hasher = Sha3_256::new();
    hasher.update(&v_enc_bytes);
    hasher.update(&r.symbols);
    let k: [u8; 32] = hasher.finalize().into();

    EchoEncapsulation { r, k }
}
```