```rust
// echo_keygen.rs (B2)

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

pub struct EchoKeyPair {
    pub sk: SymbolPath,
    pub pk: SymbolPath,
}

pub fn fingerprint(path: &SymbolPath) -> [u8; 32] {
    let mut hasher = Sha3_256::new();
    hasher.update(&path.symbols);
    hasher.finalize().into()
}

pub fn echo_keygen(graph: &EchoGraph) -> EchoKeyPair {
    let mut rng = rand::thread_rng();
    let sk_symbols: Vec<Symbol> = (0..28).map(|_| rng.next_u8()).collect();
    let sk = SymbolPath { symbols: sk_symbols };

    let v_priv = graph.resolve(0, &sk);

    let mut pk_symbols: Vec<Symbol> = Vec::with_capacity(28);
    let mut current_state = v_priv;
    for _ in 0..28 {
```

```
        let next_sym = rng.next_u8();
        let idx = (next_sym % 16) as usize;
        current_state = graph.transitions[current_state as usize][idx];
        pk_symbols.push(next_sym);
    }
    let pk = SymbolPath { symbols: pk_symbols };

    EchoKeyPair { sk, pk }
}
```