Interview Task: Implement a Merkle Tree Flag Extension for Private Leaves

Tags

Context:

At Seismic Systems, we modify Ethereum's Merkle Tree design for data integrity and verification. Ethereum's Merkle trees leverage **hex-prefix encoding** for representing nodes efficiently.

In Ethereum's encoding:

- The high nibble (first four bits) of the first byte contains two flags:
  - **Lowest bit**: Encodes whether the length of the nibbles is odd.
  - **Second-lowest bit**: Encodes whether the node is a leaf (`t` flag).
- The low nibble contains either zero (for an even-length path) or the first nibble (for an odd-length path).

Task Overview

Seismic Systems needs to extend this encoding format by **adding a third flag** in the high nibble of the first byte:

- **Third-lowest bit**: Indicates whether the node contains **private information**.
  - **Set to** `1` if the node is a **leaf node containing private data**.
  - **Set to** `0` for public leaves and all internal nodes (branches).

This ensures that when building a Merkle tree, leaf nodes can explicitly mark whether they hold private data without altering the existing tree structure for public nodes.

Requirements

1. **Modify the function signature**:
   - Update the `HashBuilder::add_leaf` method to include an additional parameter:

     ```
     fn add_leaf(&mut self, value: &[u8], is_private: bool)
     ```

   - The `is_private` flag should be passed by callers to specify whether a leaf contains private data.
2. **Update Merkle tree node encoding**:
   - When encoding leaf nodes, apply **hex-prefix encoding** as specified in the Ethereum Yellow Paper, with the following extension:
     - Use **three bits** in the high nibble of the first byte:
       - **Bit 0**: Oddness of the length (as in Ethereum).
       - **Bit 1**: Leaf indicator `t` (as in Ethereum).
       - **Bit 2 (NEW)**: Private indicator (`1` for private leaves, `0` otherwise).
   - Ensure **internal nodes** (branches) do not set the private flag.
3. **Update up the stack**
   - As you make these modification, other parts of the repo will begin to show compilation errors. Follow these updates up the stack, making compatible changes, until the exposed repo functions are also compatible.
   - We expect you to have to make changes to:
     - src/hash_builder
     - src/nodes
     - src/proof
     - src/root.rs
4. **Update all relevant tests**:
   - Modify existing tests or add new ones to cover:
     - Public leaf encoding (should behave like standard Ethereum).

    - Private leaf encoding (should have the private flag set).
    - Internal node encoding (should always have the private flag unset).
    - Etc.
5. **Create a Pull Request:**
    - Provide a clear commit message summarizing your changes.
    - Include documentation comments in the code explaining the purpose of the private flag.

Getting Started

Fork alloy-rs/trie and **CHECKOUT TO THE FOLLOWING COMMIT**: 4d91c0f. Name this branch “original” and push it. Then create a separate working branch that can later become a PR against “original.”

Evaluation

You will be evaluated on Correctness, Backward Compatibility, Code Quality, and Completion Speed. You may use any tooling you desire, such as ChatGPT or Cursor, for this technical. This exercise is designed to mirror real-world pull request workflows. Approach it as you would a production feature.

As a preference, keep your diffs from the original commit minimal while accomplishing the above goals. Focus on functional changes, but if adjustments are necessary to maintain correctness or clarity, they are acceptable.