GEM^2-Tree: A Gas-Efficient Structure for Authenticated Range Queries in Blockchain

Ce Zhang*, Cheng Xu*, Jianliang Xu*, Yuzhe Tang‡, Byron Choi*

* *Department of Computer Science, Hong Kong Baptist University, Hong Kong*
‡ *Department of Electrical Engineering & Computer Science, Syracuse University, NY, USA*
{cezhang, chengxu, xujl, bchoi}@comp.hkbu.edu.hk, ytang100@syr.edu

Problem Statement

- **Authenticated Range Queries in Blockchain**
  - Four parties: data owner, blockchain with smart contract, service provider, and query client.
  - The blockchain itself and the cloud service provider are components of the hybrid-storage blockchain.
  - The data owner sends $o_i = \langle k_i, v_i \rangle$ to the service provider and sends $\langle k_i, \mathsf{h}(v_i) \rangle$ to the blockchain.
  - Objective: design an efficient index that supports range queries with integrity assurance in a hybrid-storage blockchain.
- **Threat Model**
  - The data owner, the blockchain, and the query client are assumed to be trusted.
  - The service provider is seen as an untrusted party since it may modify, add, or delete data.
    - ✓ The service provider returns query results with *verification object* (VO).
    - ✓ The user verifies the *soundness* and *completeness* of the results.
      - · Soundness: all of the answers in the result satisfy the query criteria and are originated from the data owner.
      - · Completeness: no valid answer is missing from the query result.

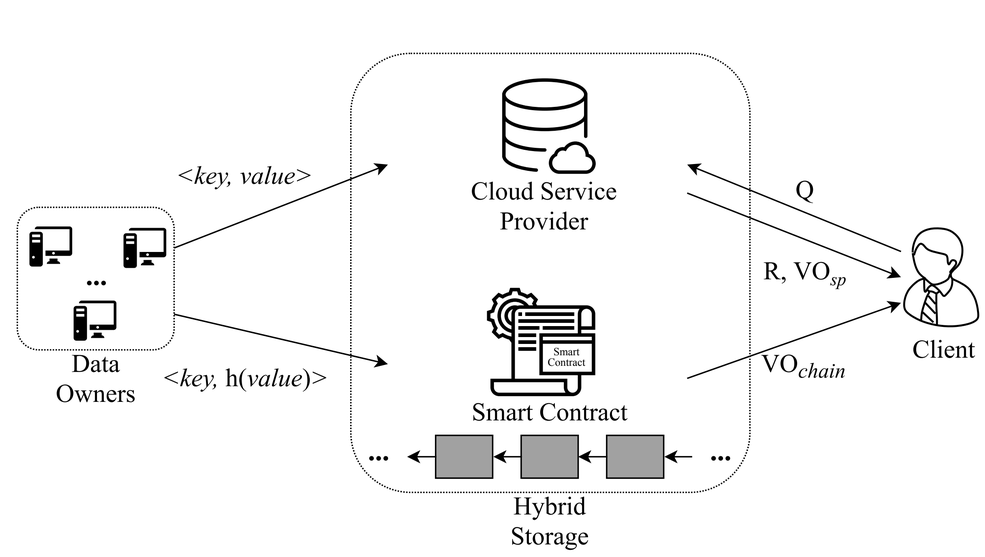


Fig. 1: Authenticated Query Framework in Hybrid-Storage Blockchain

Preliminaries

- **Merkle Hash Tree**

A MHT is a binary tree that can be used to authenticate a set of data objects with logarithmic time complexity.

  - Each leaf node contains the hashes of the indexed objects.
  - Each internal node contains a hash which is computed using its two child nodes.
  - The root hash is used to authenticate the data objects.

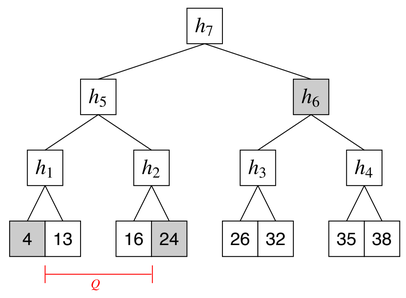


Fig. 2: Merkle Hash Tree

- **Smart Contract**
  - Trusted program that manages data in the blockchain.
  - Require transaction fee denominated in *gas* in Ethereum platform.
  - Target: minimize *gas* cost

Operation	Gas Used	Explanation
C_{sload}	200	load a word from storage
C_{sstore}	20,000	store a word to storage
$C_{supdate}$	5,000	update a word to storage
C_{mem}	3	access a word in memory
C_{hash}	$30 + 6 \cdot \lvert words \rvert$	hash an arbitrary-length data

Baseline Solutions

- **Merkle B-tree (MB-tree)**
  - Both the smart contract and the service provider maintain the MB-tree.
  - Maintenance cost

$$C^{\text{insert}}_{\text{MB-tree}} = \log_F N \Big(2C_{sstore} + 2C_{supdate} + (2F+1)C_{sload} + C_{hash} \Big) + C_{sstore}$$

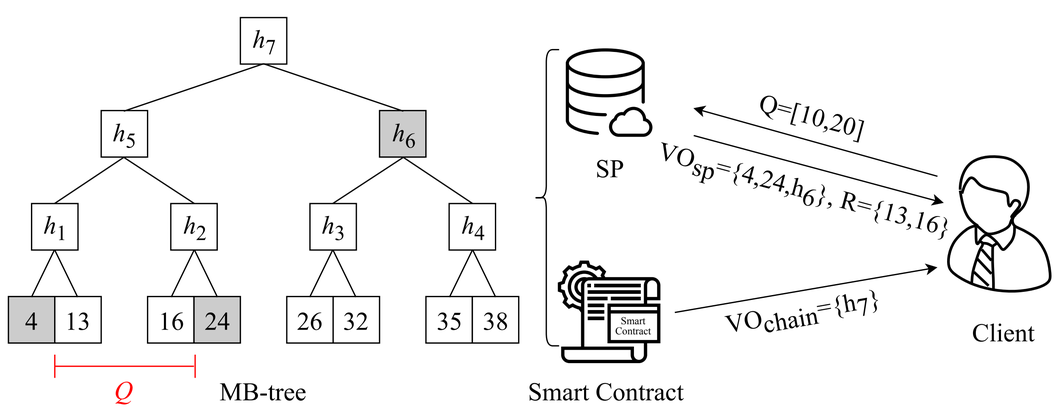


Fig. 3: MB-tree Example in Hybrid Storage Blockchain

- **Suppressed Merkle B-tree (SMB-tree)**
  - Initial principal: only the root hash is used during the client verification.
  - Suppress all nodes of the MB-tree and only materialize the root node.
  - Smart contract trades write operations with more reads and computations.
  - Maintenance cost

$$C^{\text{insert}}_{\text{SMB-tree}} = N \Big(C_{sload} + \log N \cdot C_{mem} + \frac{1}{F} C_{hash} \Big) + C_{sstore} + C_{supdate}$$

Gas Efficient Merkle Merge Tree (GEM^2-tree)

- **Structure**
  - Maintain multiple separate structures: a large fully-structured MB-tree as a major index and a series of small structured SMB-trees to index newly inserted objects.
    - ✓ A new object is always inserted into the smaller SMB-trees (more gas-efficient).
    - ✓ The objects indexed by the SMB-trees can be merged to the MB-tree *in batch* (optimize the update cost).
  - Organize the storage space into a set of exponentially-sized partitions.
    - ✓ Use *logical* partitions which change dynamically along with merges.
    - ✓ Each partition contains up to two SMB-trees that are merged with more insertions.

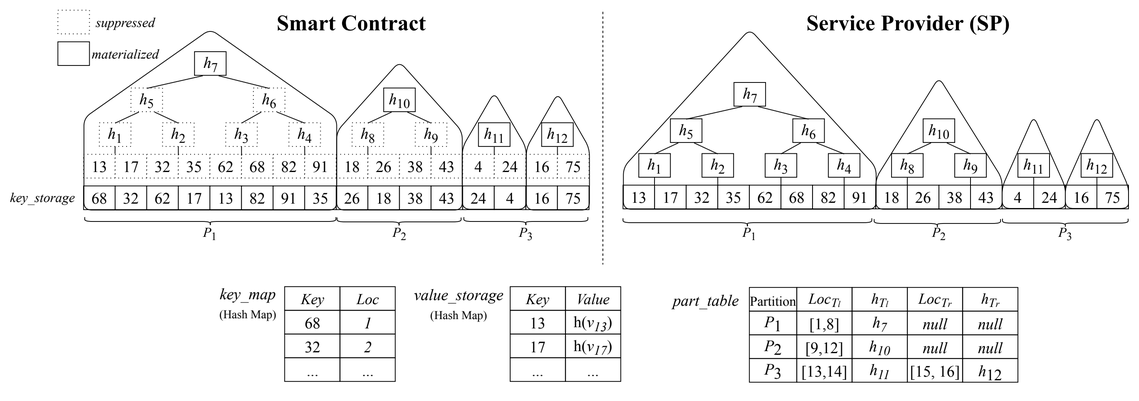


Fig. 4: Overall Structure of GEM^2-tree with Hybrid Storage

- **Authenticated Query Processing**
  - The service provider traverses the sub-trees to compute the results and VO_{sp}.
  - The root hashes of each tree are retrieved from the blockchain as VO_{chain}.
  - The result, VO_{sp}, VO_{chain} are used as the input of the client verification.
- **Optimization index: GEM^{2*}-tree**
  - Two level index: split search key domain into several regions in upper level; a GEM^2-tree is built for each region in lower level.
    - ✓ Use more SMB-trees to gain more gas reduction.
    - ✓ Use space splitting in upper level to reduce the traversal of the small SMB-trees.

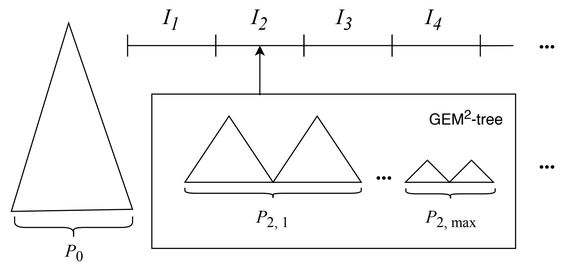
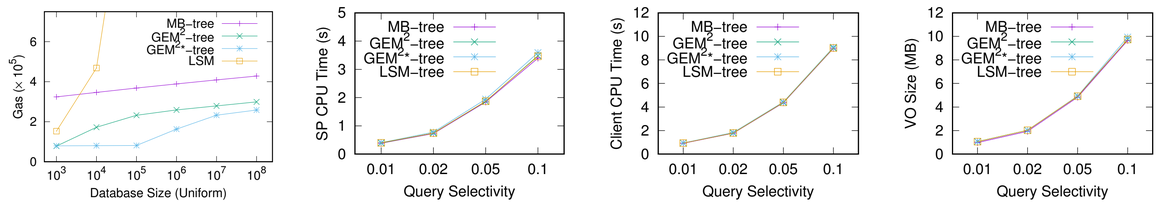


Fig. 5: GEM^{2*}-tree

Performance Evaluation



a Insertion Cost b SP CPU Time c Client CPU Time d VO Size