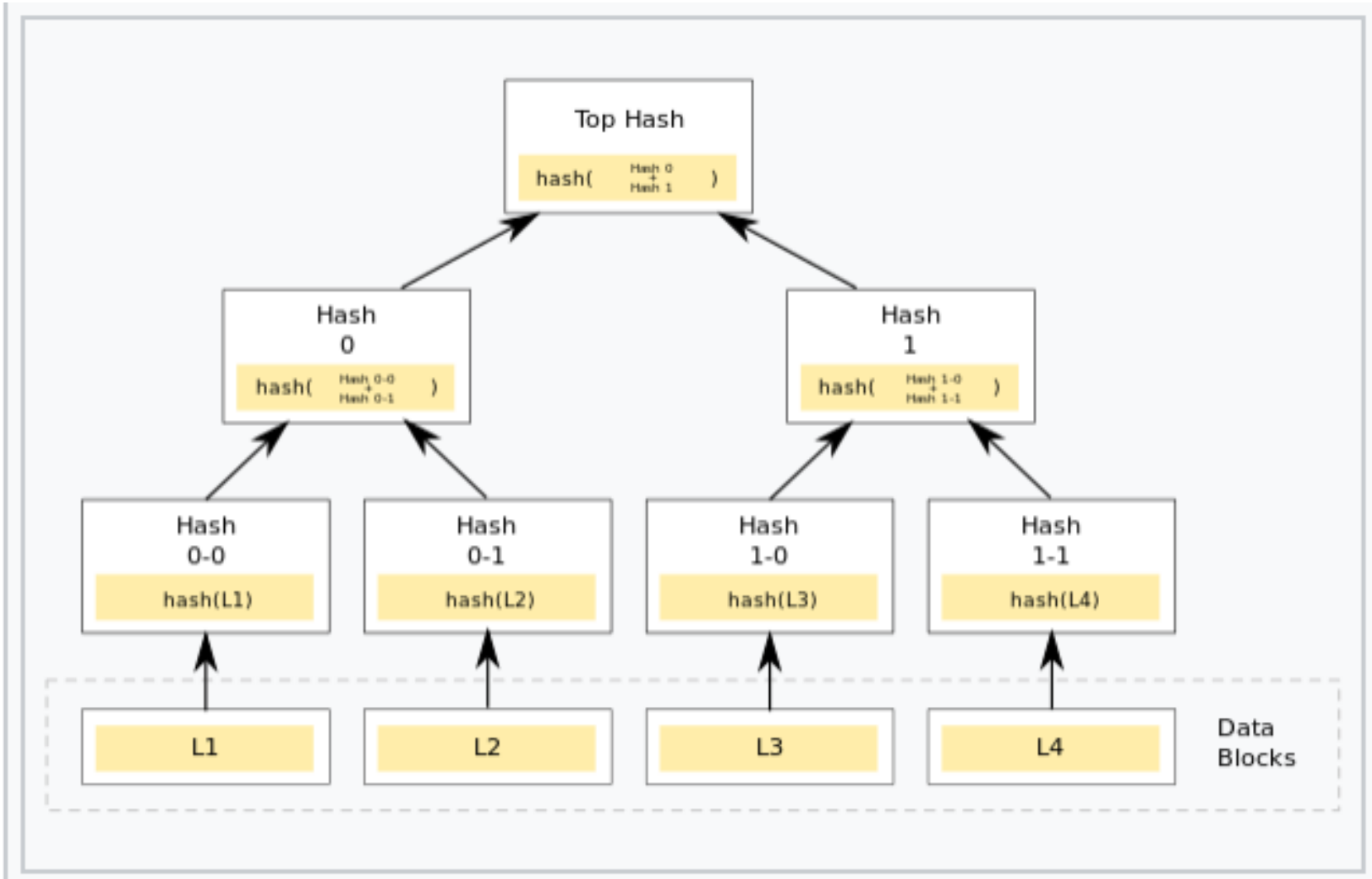
Delegating Streaming Computation with Sublinear Space

Tzu-Shen (Jason), Wang

Motivation

- A client with limit storage, delegates the computation to server
- Server with limit storage (larger than the client), computes the approximate result
- Client verify the result
  - Use its limit storage to store an authenticated data structure
- Lower the communication complexity
- Provide privacy over data stream

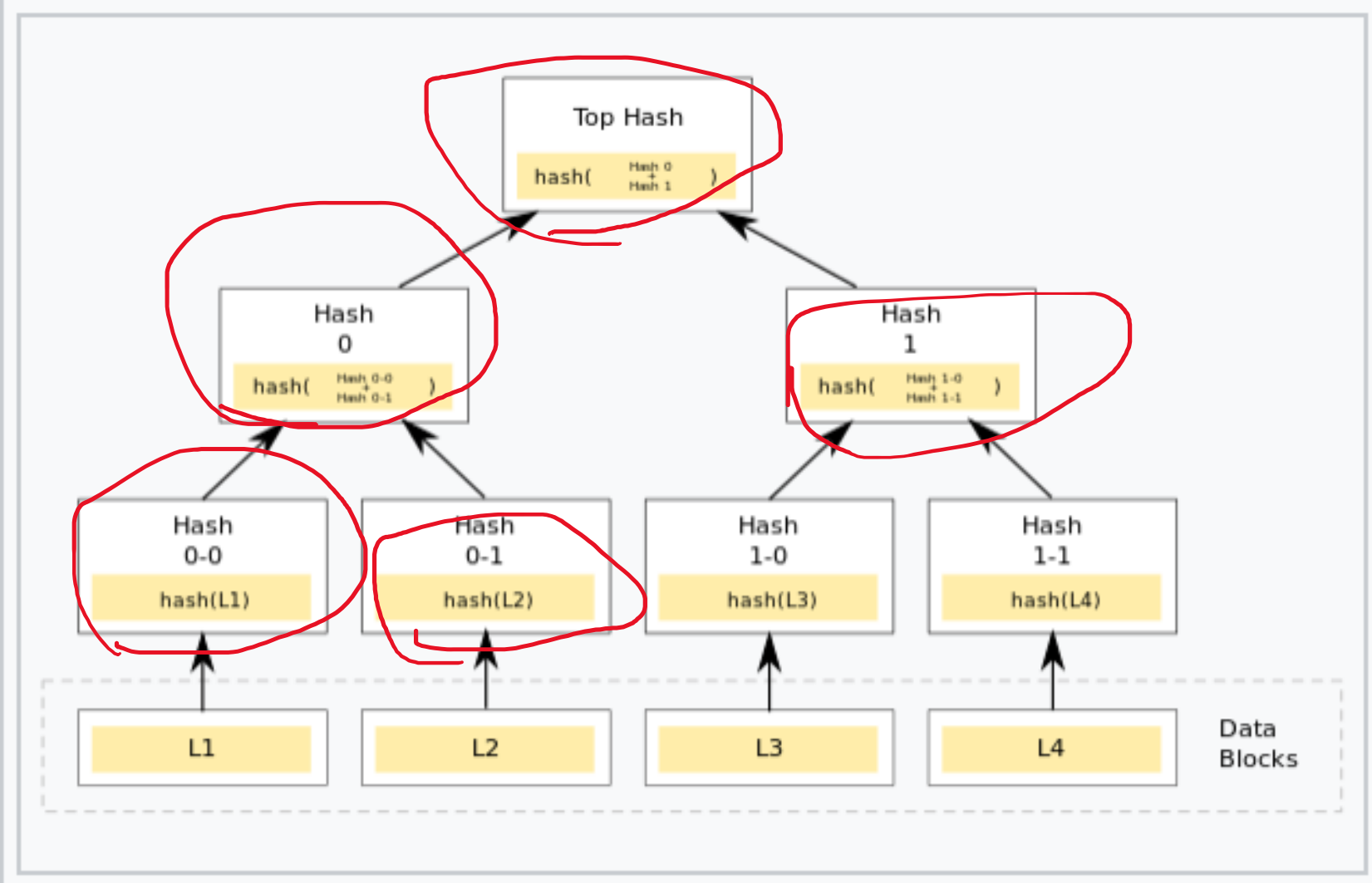
Previous Work --- Merkle tree [Merkle87, wiki]



Previous Work --- Merkle tree [Merkle87, wiki]

- With the collision resistant property of hash, if one of the leaf's value is changed, the root will also change
- The server stores all the leaf nodes
- The client only stores the root
- Server wants to proof value of a leaf node:
  - Server sends a leaf node along with the path (and its siblings) from the leaf node to the root
  - Client verify that the path and reconstruct the root, check if equal to the root it stores
  - With the collision resistant property of the hash the server can not fake a proof

An Example of Merkle tree [Merkle87, wiki]
Proof --- proving L1



Improvement of Merkle Tree[PST+13]

- Contribution of [PST+13]: Construct the collision resistant hash function based on SIS (shortest integer solution)
  - Stateless update --- client does not need to hold the leaf nodes to update the root, can directly update the tree root when each element comes in
- Problems:
  - The storage of the Server: linear to universe size
  - Communication complexity, sublinear of universe size (O(path) = log (#leaf-node))

Our Contribution

- Reduce server's storage --- reduce #leaf nodes
- Reduce communication complexity
  - Comes with the reduction of the number of leaf nodes (the path is O(log #leaf-node))
- Generalize to general linear sketches
- Model for privacy

Reduce Storage

- Utilize the technique similar in count sketch to reduce #leaf node
- Instead of mapping each element into a leaf, map multiple elements into a bucket
  - Update(x, freq(x)): B(h(x)) += S(x)*freq(x), S(x) ∈ {-1, 1}
  - Retrieve(x): S(x)B(h(x))
- Two hash function:
  - The first one decide which bucket to add
  - The second one based on SIS, is collision resistant, construct the Merkle-like tree that gives us stateless update
- Reduce the prover's storage from O(m) to O($\frac{1}{\varepsilon}\log \mathrm{m}^2$)

Reduce Storage (2) --- Security Proof

- Proof in reduction that if the prover is able to generate a fake proof in our protocol, then it is able to generate a fake proof in [PST+13] (where each element is stored on a leaf node) with self select data stream
  - Selective adversary: can select the data stream
  - When our protocol updates leaf node y with frequency z, we also want to update [PST+13]'s leaf node y with frequency z
  - Observation: for every incoming element (x, freq(x)) for our protocol, we update node $h_1(x)$, with frequency S(x)freq(x)
  - We give element ($h_1(x)$, S(x)freq(x)) to [PST+13]
  - If we can break our protocol for data stream (x, freq(x)), we can break [PST+13] with data stream ($h_1(x)$, S(x)freq(x))
- We also do a simulation proof
  - Property based proof: define property, and check if the property holds --- might miss property
  - Simulation proof: Can do whatever the adversary can do without the exchanging message
    - Can generate the message itself
    - Exchanging messages is what a party learns in the protocol

Generalize to General Linear Sketches

- Extend the data structure to have negative values for leaf nodes
  - With one bit sacrification
  - Also reduce from the data structure of [PST+13]
- If there exists a linear sketch that uses S words of space and solves a problem P with probability at least $\frac{2}{3}$. Then there exists a protocol π that solves P with probability at least $\frac{2}{3}$ in which a prover stores and communicates S words, and a verifier stores O (log n) bits

Different Model

- Privacy
  - To hide the data stream from client, the server utilize homomorphic encryption (he) and sends the encryptions to the client
  - Homomorphic property: Compute on the encrypted data, then decrypt, equals to the computation on plaintext
    - Dec(C(Enc(x))) = C(x)
  - With the homomorphic property, the client and computes on the encrypted inputs
  - Curator model --- honest curator generates and sends all the data
    - Can send permutation of data stream to the server if we also want to hide the exact data stream from the server
- Dealing with the error of lattice based encryption
  - Open every c (constant) encryptions

Reference

- Wiki https://en.wikipedia.org/wiki/Merkle_tree
- [Merkle 87] Merkle, R. C.. "A Digital Signature Based on a Conventional Encryption Function". *Advances in Cryptology – CRYPTO '87*. Lecture Notes in Computer Science. Vol. 293. pp. 369–378
- *[PST+13] Charalampos Papamanthou,* Elaine Shi, Roberto Tamassia, Ke Yi:
  Streaming Authenticated Data Structures. EUROCRYPT 2013: 353-370