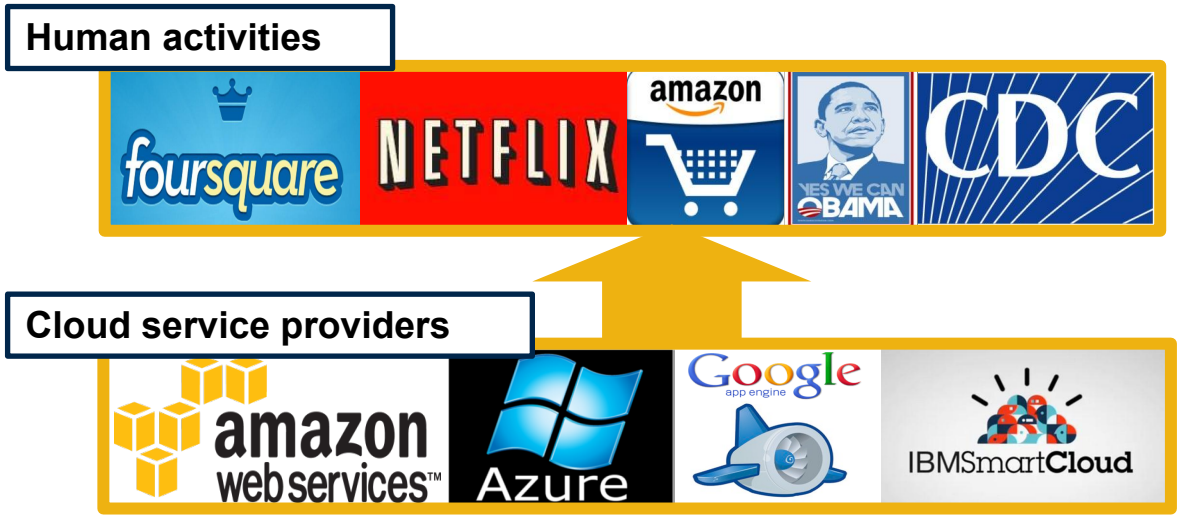
Lightweight Authentication of Freshness in Outsourced Key-Value Stores (ACSAC'14)

Yuzhe Tang, Ting Wang, Ling Liu, Xin Hu, Jiyong Jang

IBM Research

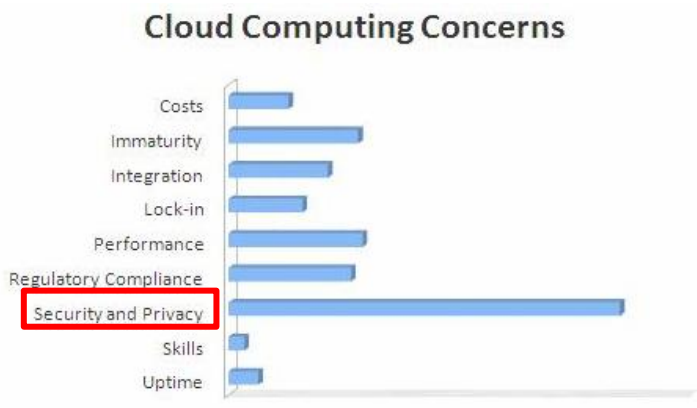
Cloud Computing

- Cloud computing has arrived:
  - Almost all human activities can be supported by cloud
  - Cloud service providers



Cloud Security/Lack of Trust

- Security becomes the biggest issue for adoption of cloud
- Lack of trust to public cloud
  - “Don’t be evil?”
  - Being caught evil all the time….
    - Intentionally evil
    - Accidentally



U.S. gov wants your online data (PRISM scandal!)

We'll be back shortly.

“Evil" cloud service provider

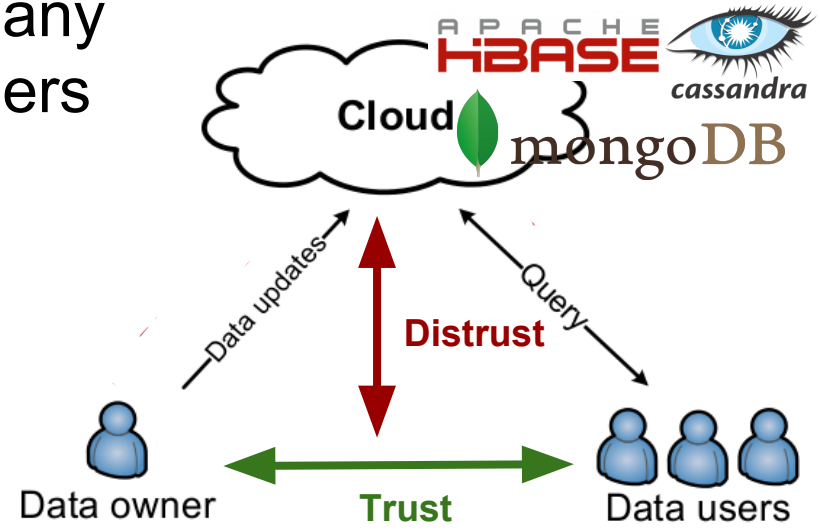
from Internet

Narrow Down the Problem: Securing Key-Value Stores in Untrusted Cloud

- Security issues of interest: authentication
  - “Does cloud ‘modify’ my data?” (Authenticity)
  - [Disclaimer] Not “Does cloud disclose my data?” (Confidentiality)
- System of interest: Key-value store (KV store)
  - Widely used for big-data storage
  - Simple yet powerful API: Key-value data, Put/Get
  - Easy to scale out

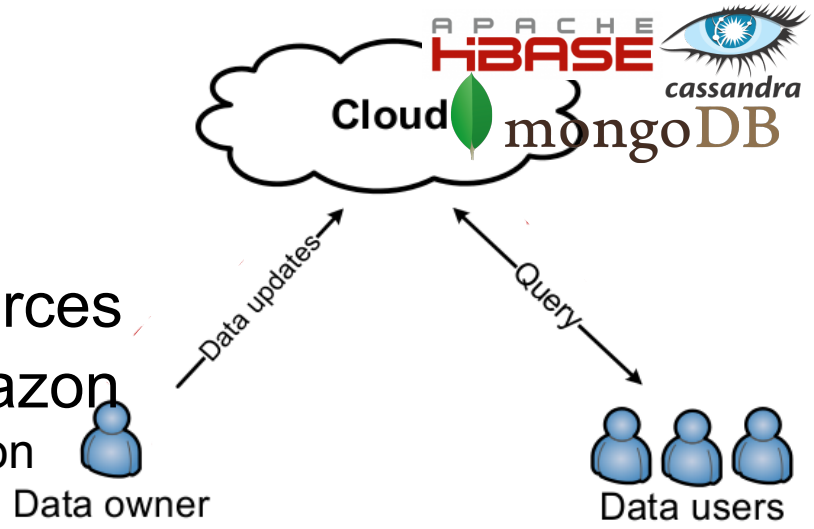
Scenario: Outsourced Key-Value Stores

- Small startup company outsources big-data (from its customers) to cloud:
  - Data owner: small company
  - Users: company customers
  - Public cloud: Amazon, w outsourced KV store



Scenario: Freshness

- Key-value data model:
  - Versioned data: $<k, v, t>$
  - $Put(k,v,t)$ / $Get(k,t_q) \rightarrow <v,t>$
  - e.g.: foursquare outsources its end users' data to Amazon
  - k: social user name; v: user location
- Freshness is important
  - Given $Get(k,t_q) \rightarrow <v,t>$, version $<v,t>$ is newest as of t_q
  - e.g. a foursquare user needs her friends' *current* location.



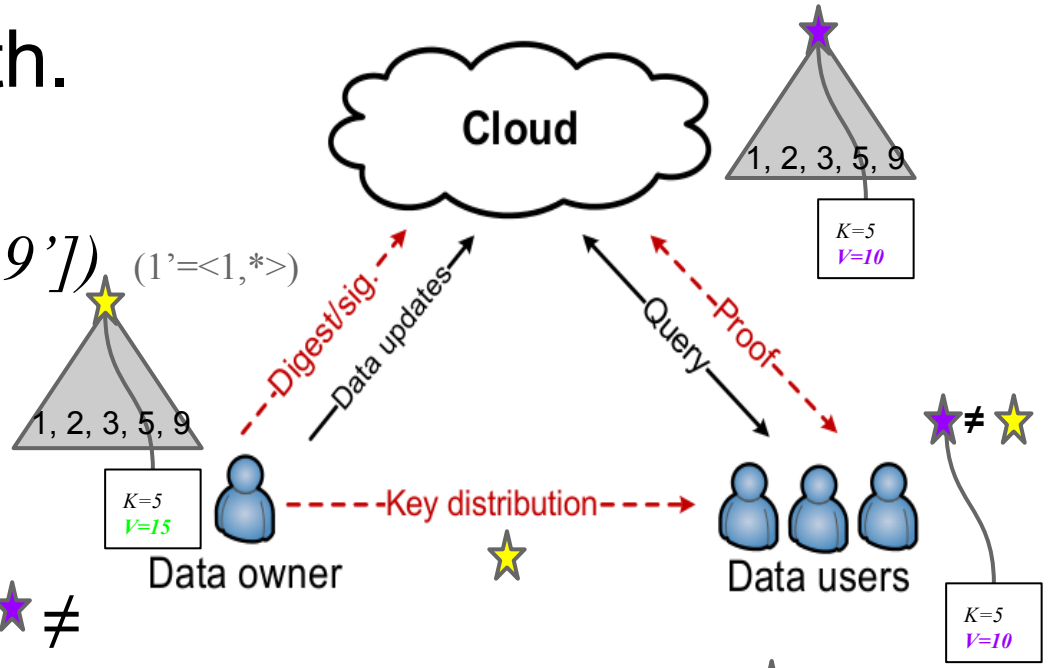
Scenario: Authentication of Freshness

With data returned from untrusted public cloud, how can data users be assured that the data is fresh?

Background: Merkle Tree Auth. Framework

Merkle tree based auth.

- Digest and sign
  - *RootHash([1',2',3',5',9'])*
- Proof construction
  - *AuthPath(h(<5,10>))*
- Proof verification
  - *AuthPath(h(<5,10>))* ★ ≠ *AuthPath(h(<5,15>))* = *RootHash([1',2',3',5',9'])* ★



Challenges: Freshness Auth. by Merkle Tree

- Freshness auth. requires both member/non-member- ship test
- “My friend’s *current* location is *A*”:
  - He did moved to location *A* one hour ago
  - He did not move during the last hour.

    (Non-membership of a “moving” event/update during the last hour)
- Non-membership test requires Merkle tree to build on *ordered data* (**KoMT: Key-ordered Merkle Tree**)
  - Challenge to auth. ordered big-data while handling updates
  - Ordered means to keep entire dataset local

Problem Formulation

Lightweight freshness auth. over intense data updates

1. Big-data with historical access
   - Versus stream-auth.[1] which only considers small windowed data.
2. Real-time verification
   - Versus audit-based auth. [2] which can't detect anomaly in real-time
   - Proof needs to accompany the query result
3. Lightweight signing
   - Versus traditional Auth-DS (e.g. MHT)[3] which maintain huge local states, not lightweight

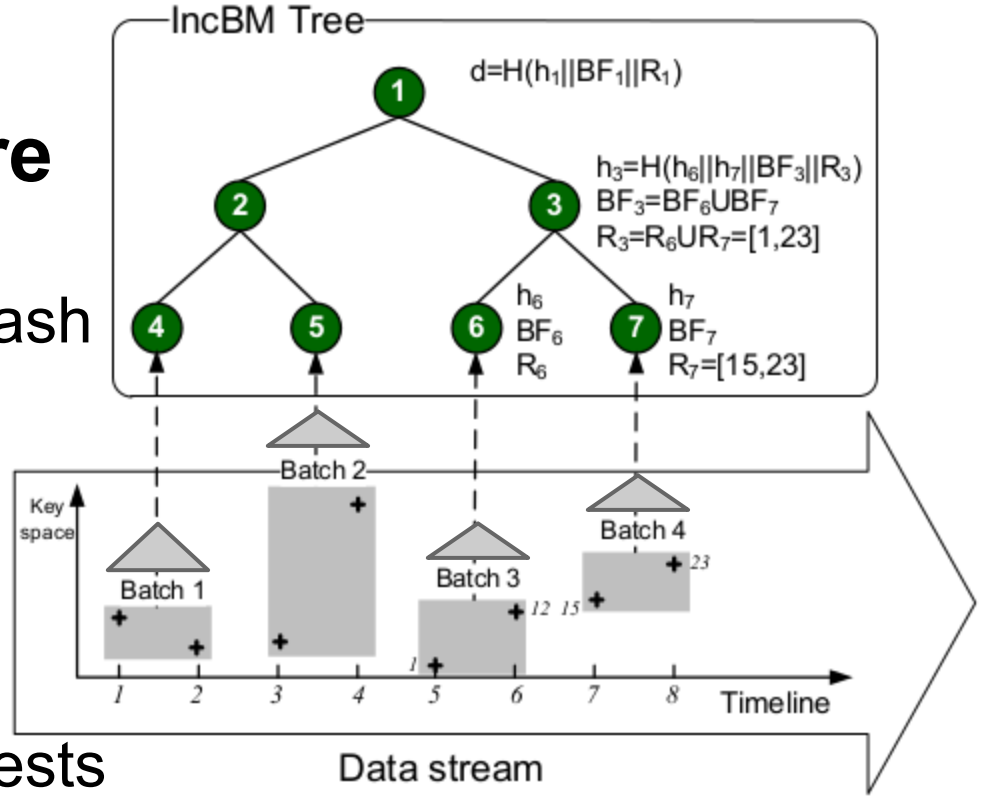
Proposal: Multi-Level Digest

- Design: keep KoMT local state small and in memory
  - When KoMT grows out of local “memory”, build a data summary **(Bloom-filter digest)** before dumping the local-state data.
  - Based on BFs, build another Merkle tree (**IncBM tree**) for better query performance.
- IncBM tree
  - Structure, maintenance op, and query op

IncBM Tree: Structure

Per-node data digest and hash

- Bloom-filter BF
- Range digest R



IncBM: Merkle tree with digests

$$
\begin{aligned}
R(\text{node}) &= R(\text{left\_child}) \cup R(\text{right\_child}) \\
BF(\text{node}) &= BF(\text{left\_child}) \cup BF(\text{right\_child}) \\
h(\text{node}) &= H(h(\text{left\_child}) \| h(\text{right\_child}) \| BF(\text{node}) \| R(\text{node}))
\end{aligned}
$$

IncBM Tree: Maintenance

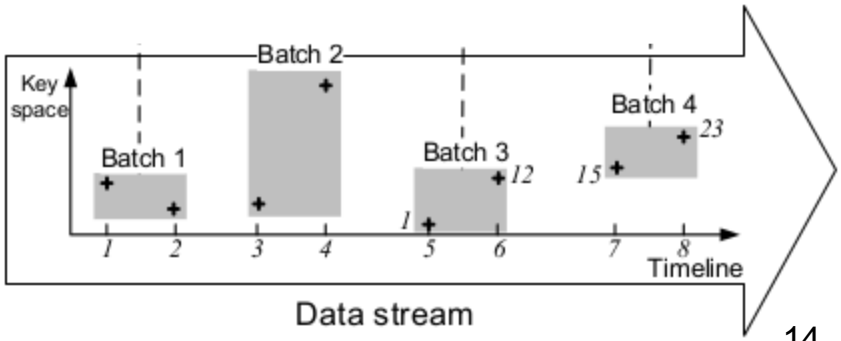
Signing a stream of data-updates

Workflow:

1. Batching data updates
2. Build MHT (upon data size reaches memory capacity)
3. Updating local partial IncBM tree
4. Sign the root of local IncBM tree, and upload it to the cloud

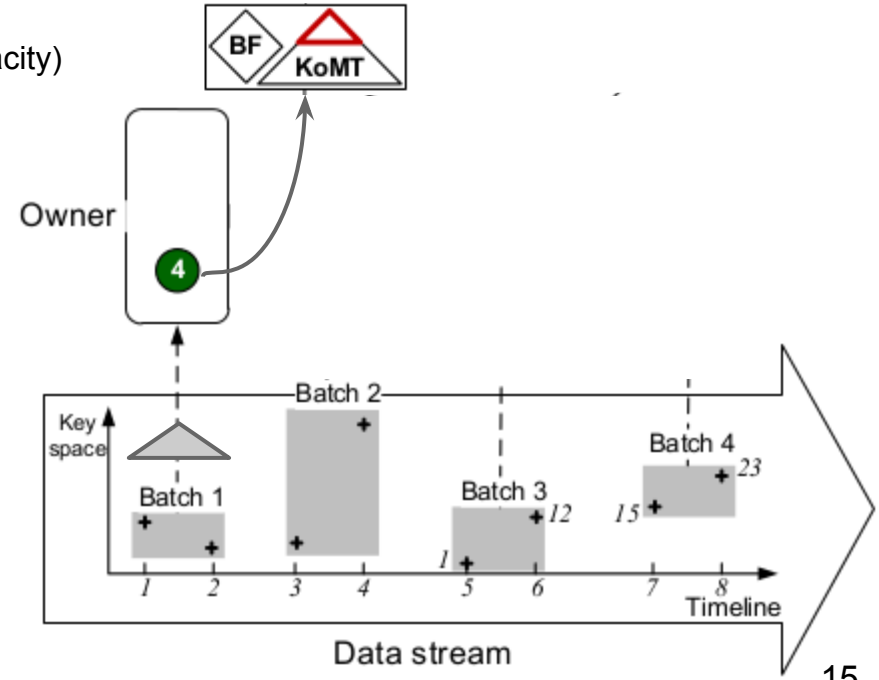
IncBM Tree: Maintenance Workflow (1)

1. Batching data updates
2.
3.
4.



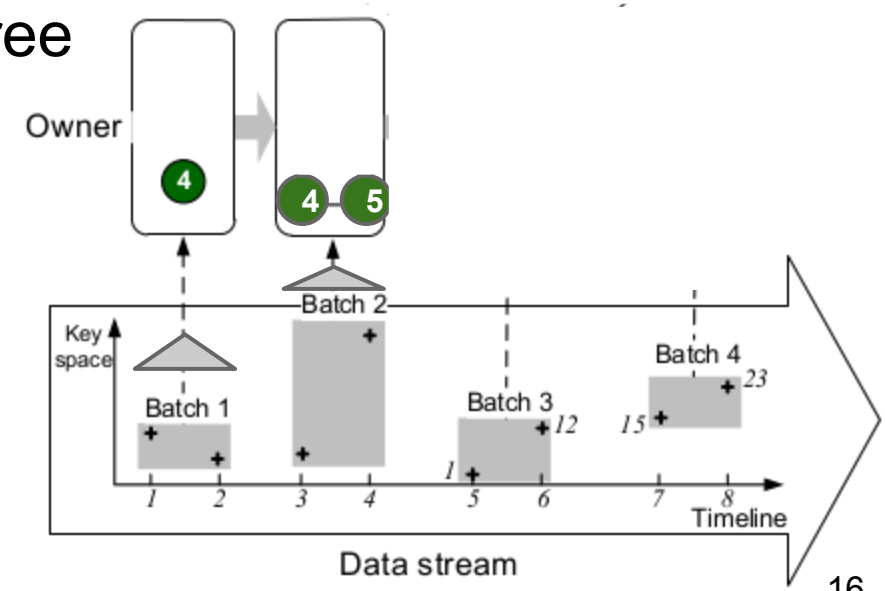
IncBM Tree: Maintenance Workflow (2)

1. Batching data updates
2. Build MHT (upon data size reaches memory capacity)
3.
4.



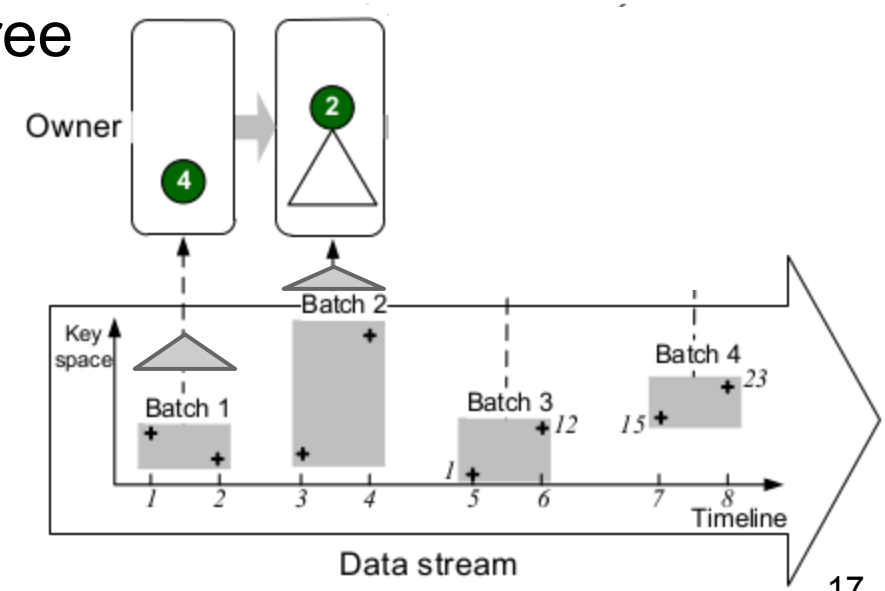
IncBM Tree: Maintenance Workflow (3)

1. Batching data updates
2. Build MHT (upon data size reaches memory capacity)
3. Updating local partial IncBM tree
4.



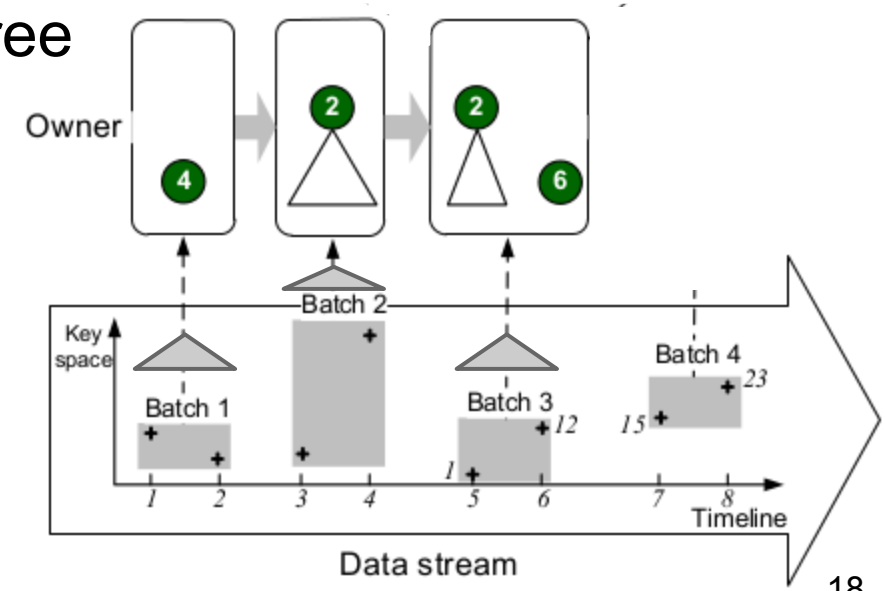
IncBM Tree: Maintenance Workflow (4)

1. Batching data updates
2. Build MHT (upon data size reaches memory capacity)
3. Updating local partial IncBM tree
4.



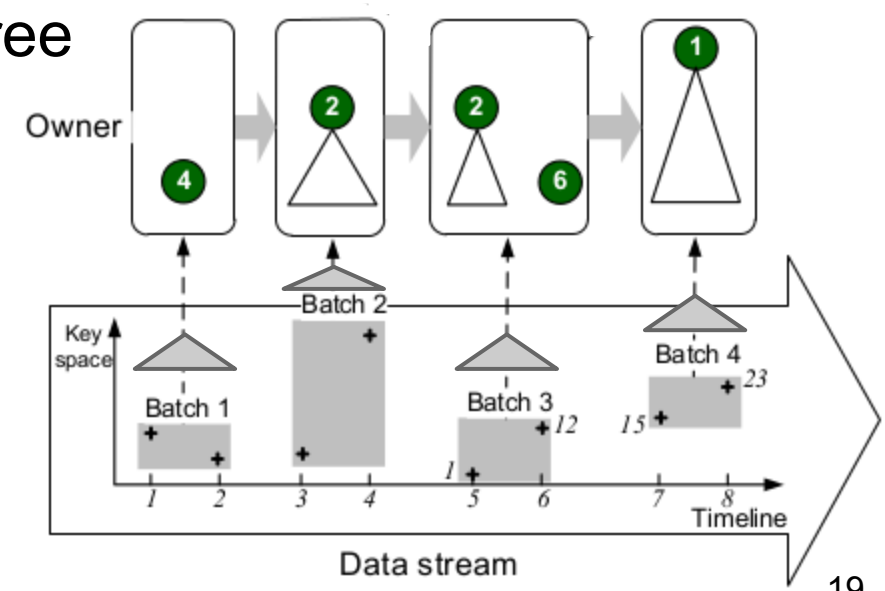
IncBM Tree: Maintenance Workflow (5)

1. Batching data updates
2. Build MHT (upon data size reaches memory capacity)
3. Updating local partial IncBM tree
4.



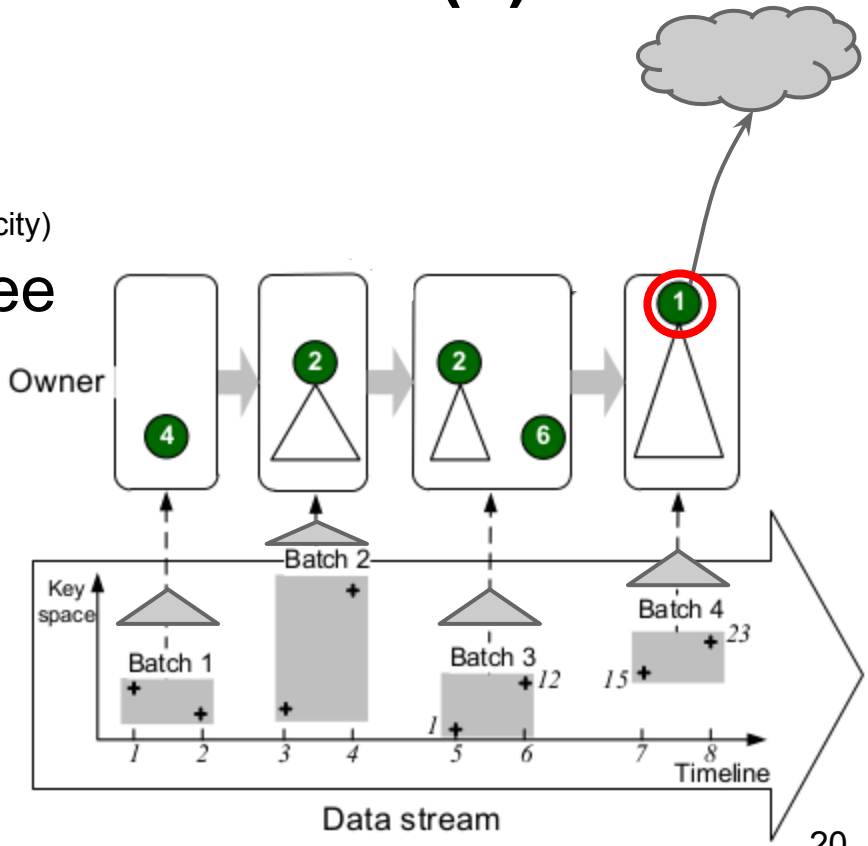
IncBM Tree: Maintenance Workflow (6)

1. Batching data updates
2. Build MHT (upon data size reaches memory capacity)
3. Updating local partial IncBM tree
4.



IncBM Tree: Maintenance Workflow (7)

1. Batching data updates
2. Build MHT (upon data size reaches memory capacity)
3. Updating local partial IncBM tree
4. Sign the root of local IncBM tree, and upload it to the cloud



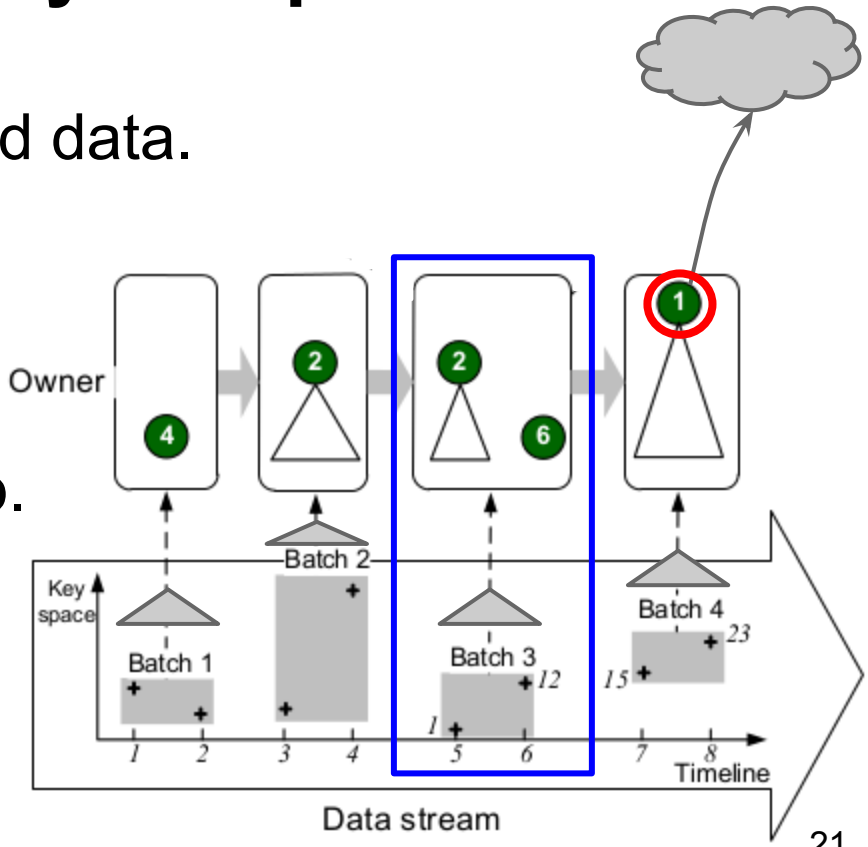
IncBM Tree: Small Memory Footprint

Node ① signs all **4** batches of sorted data.

Yet, at any time, it only stores at most **1** batch with extra digest.

- Data batch as large as mem-cap.

Using **1** memory space it signs **4** mem-cap. worth of data.



IncBM Tree: Query Proof Construction

Prove:

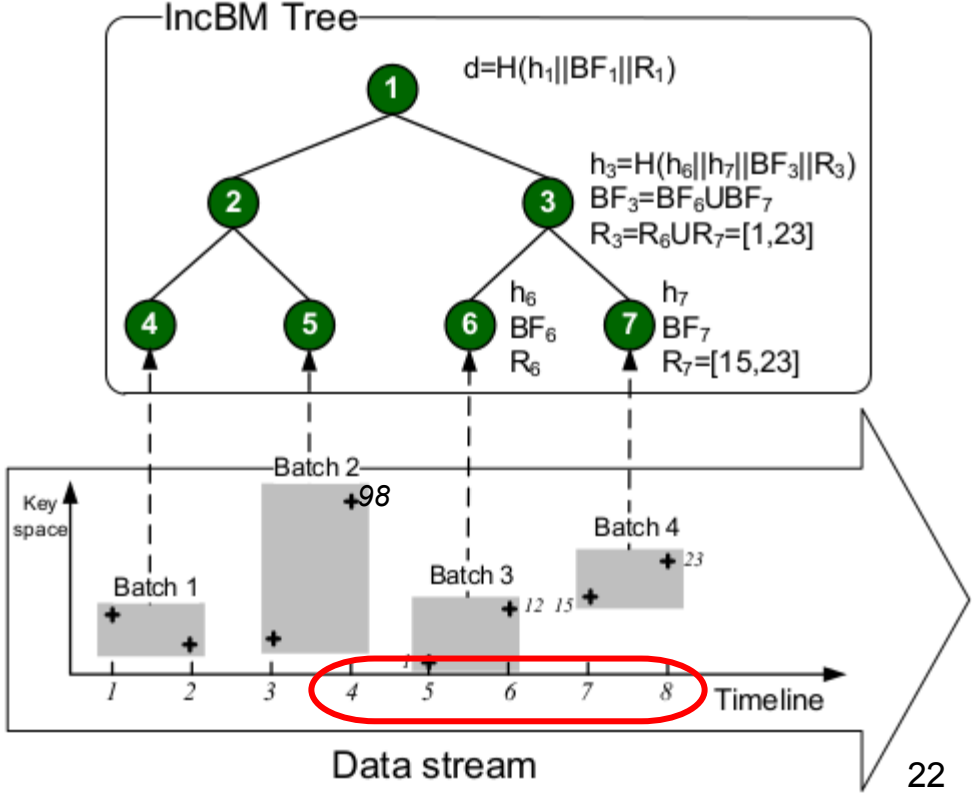
"Key 98 is fresh as of time 8"

Equivalent to say:

1. "Key 98 is there at time 4"

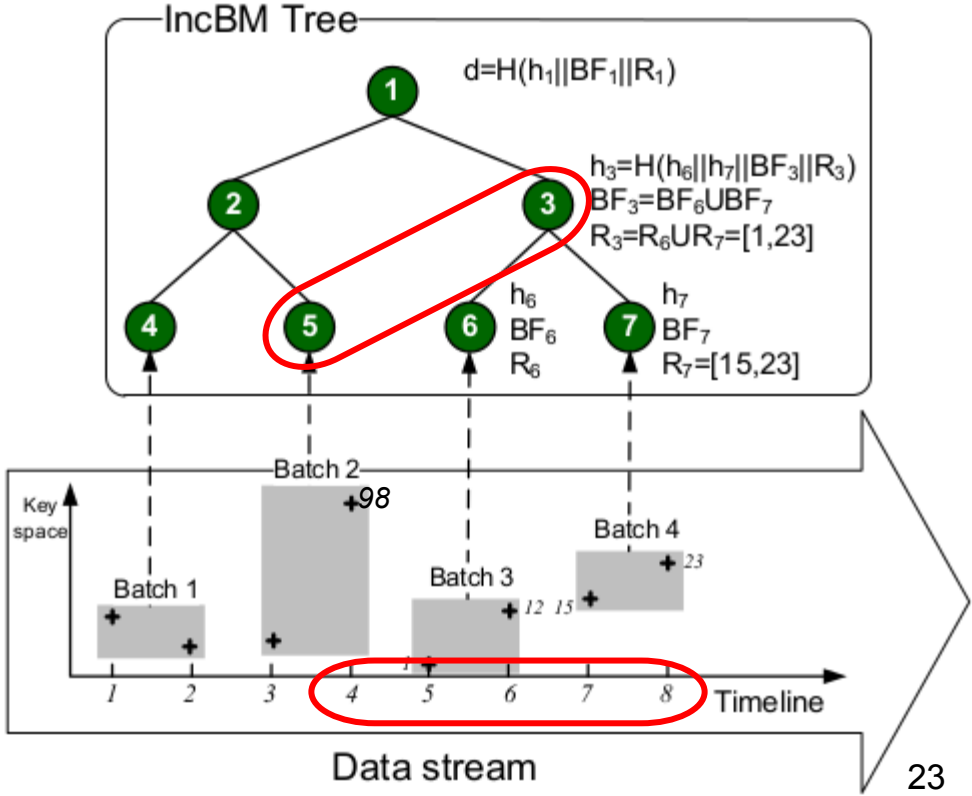
AND

2. "Key 98 is not in [5,8]"



IncBM Tree: Query Proof Construction

1. "Key 98 is there at time 4"
   - can be proved by node 5 using Merkle root
2. "Key 98 is not in [5,8]"
   - maybe proved by BF by node 3
   - if not, go down to node 6 AND 7



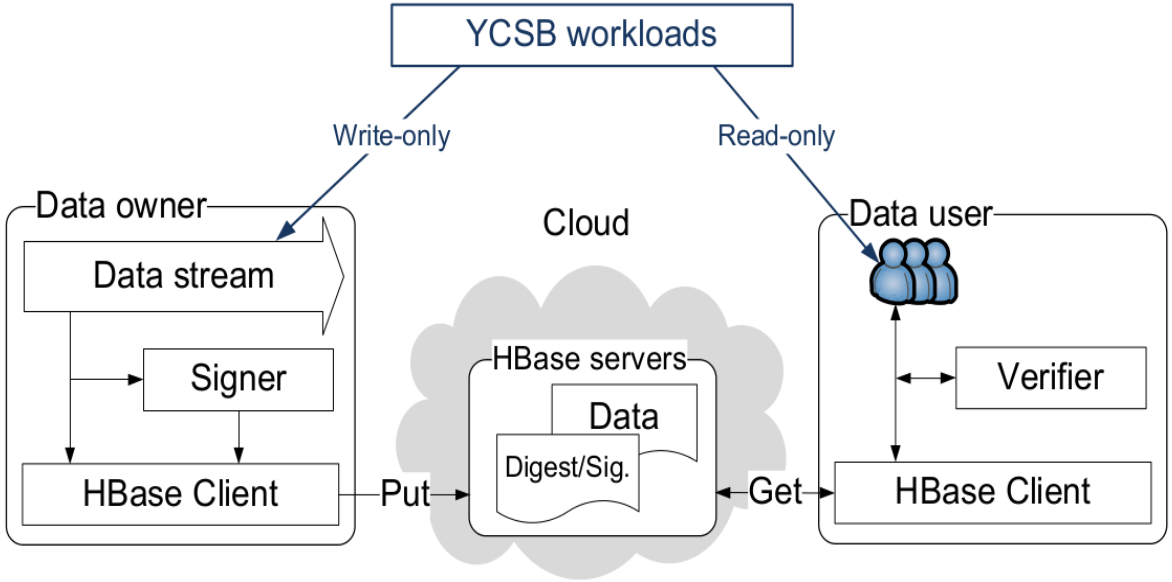
System prototyping: HBase

Digest and sign

- Signer produces signed root
- submit to cloud thru *Put* call

HBase in cloud:

- Key-value stores:
  - Write-optimized
  - Strong consistency



- Tables: Meta-data sharded by time (digest, signatures), base data by key

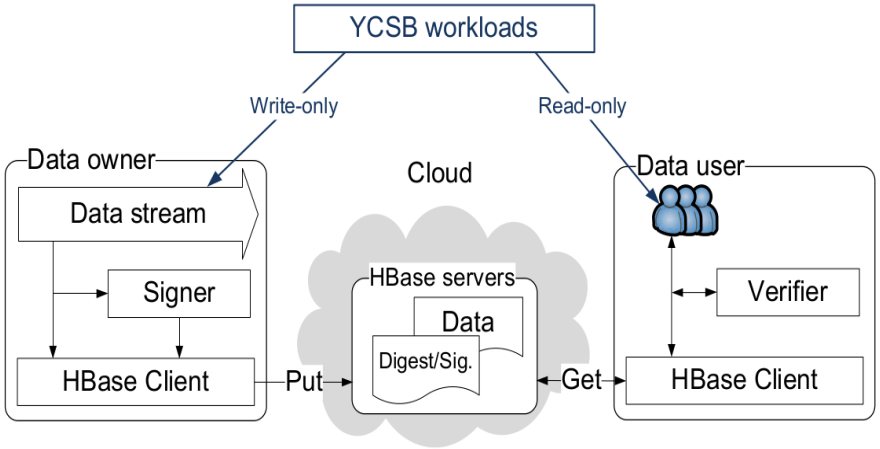
Proof verification by users

- Proof construction in cloud, send back along w *Get* call
- *Get* result verified by user

Performance study: Setup

Data generated by YCSB

- *Get*s thru. read-only workload
- *Put*s thru. write-only workload
- 500 GB data poured into a 10-node HBase cluster



Evaluated in Emulab

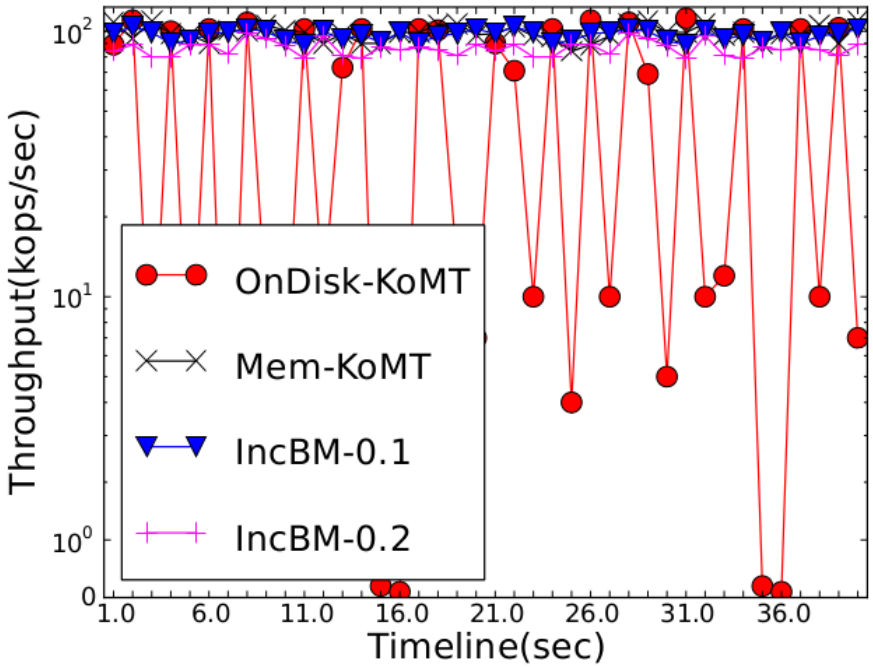
- Owner, HBase cluster, users are on separate machines
- Commodity machines: 2GB memory

Performance study: Write and sign

- KoMT-onDisk: MHT size triples mem. cap.
- KoMT-mem: MHT size equals mem. cap.
- IncBM-0.1: 10% mem for local partial IncBM
- IncBM-0.2: 20% mem for local partial IncBM

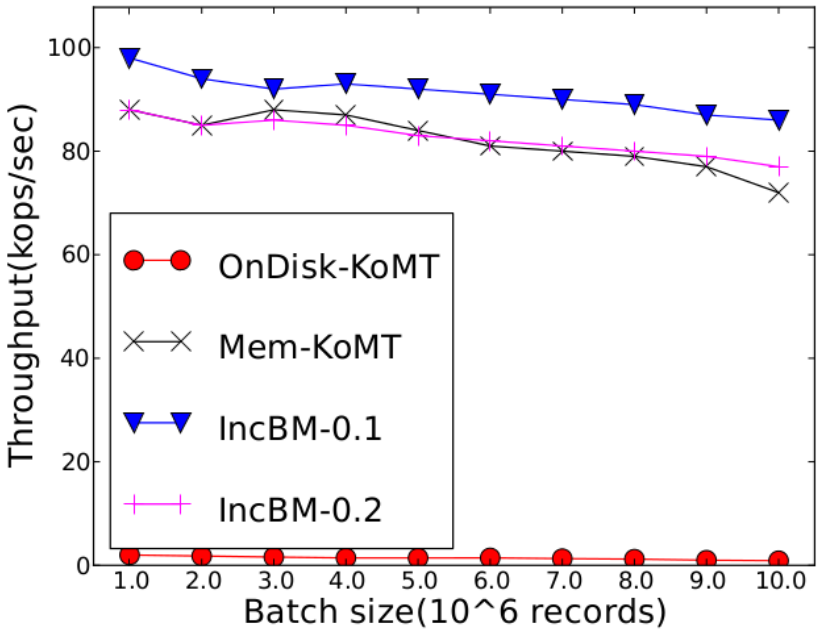
Spikes in KoMT-onDisk due to flushing data onto disk.

Others' write-performance is similar.

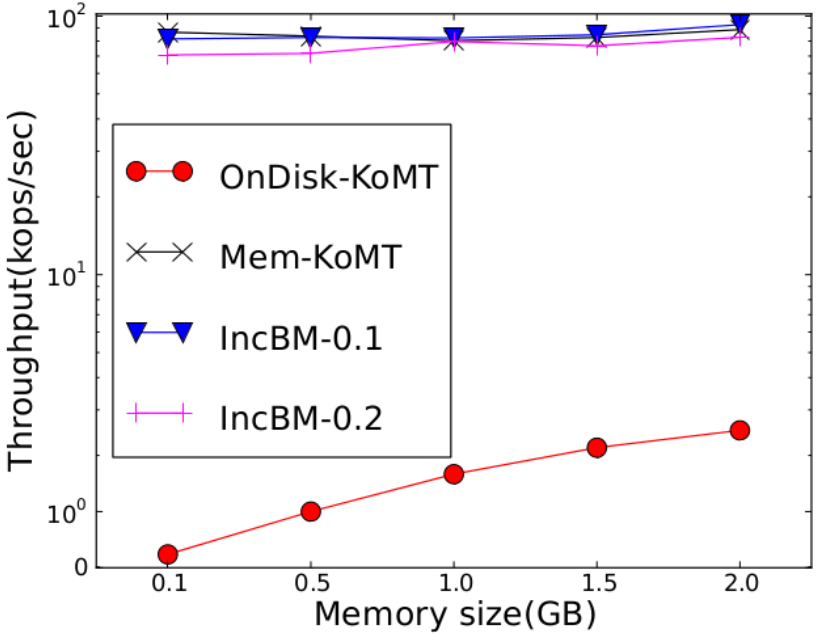


(a) Time series

Performance study: Write and sign



(b) Varying batch sizes

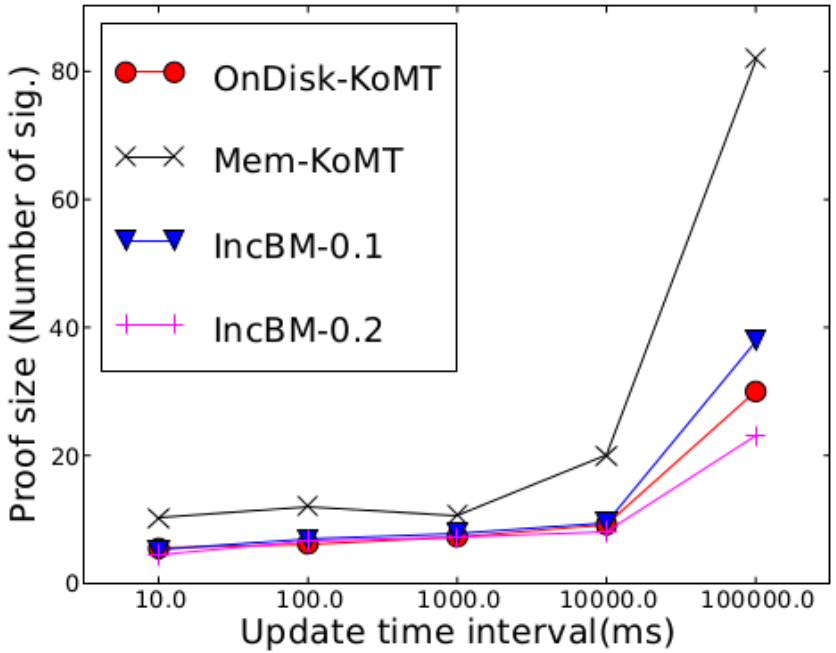


(c) Varying memory sizes

Performance study: Query and verify

Mem-KoMT is slowest because small batch size (Verifying the same time interval requires more digital signature verifications)

IncBM is the same efficient to OnDisk-KoMT on query performance



(b) Proof size

Summary

Articulated the problem of providing data freshness assurance for outsourced multi-version key-value stores.

Proposed INCBM-TREE:

1. lightweight for both data owners and end users,
2. optimized for intensive data update streams,
3. immediate authentication of data freshness in the presence of real-time and historical data accesses.

Referenced Work

- [1] Feifei Li, et al, Proof-infused streams: enabling authentication of sliding window queries on streams, VLDB'07
- [2] Raluca Ada Popa, et al, Enabling security in cloud storage SLAs with CloudProof, USENIX-ATC'11
- [3] Emil Stefanov, et al, Iris: a scalable cloud file system with efficient integrity checks, ACSAC'12

Questions?

Thank you

Contact:
Yuzhe (Richard) Tang
Syracuse University
ytang100@syr.edu