Enabling Security in Cloud Storage SLAs with CloudProof

Raluca Ada Popa, MIT; Jacob R. Lorch, David Molnar, Helen J. Wang, and Li Zhuang, Microsoft Research
A main concern is security

- Data leakage/corruption due to bugs, hackers, employees
- Many customers perceive security as main concern

Cloud storage provides extensive resources, scalability, and reliability
Security properties

- Confidentiality (C): only authorized users can read data
- Integrity (I):
  - Each get returns the content put by an authorized user
- Write-serializability (W):
  - Each user committing an update is aware of the latest update to the same block
- Freshness (F):
  - Each get returns the data from the latest committed put

Problem: cloud services do not guarantee security in SLAs

Need proofs of misbehavior
CloudProof

A secure storage system for the cloud:

1. Security mechanisms needed for SLAs with security:
   - Detection of violations for integrity, write-serial., and freshness (IWF)
   - Publicly-verifiable proofs of violation for IWF
     - Any external party can be convinced of cloud misbehavior
     - Users cannot falsely accuse cloud

2. Scalable design of security mechanisms
   - Scalable access control using modern cryptographic tools
Model

- Data owner
  - assigns permissions to users (R, RW)
  - may try to frame the cloud
- Application
- Data users
  - may attempt to bypass permissions
- Cloud
  - fully untrusted

get/put blocks
For each block:

- Confidentiality: owner gives a secret key for encryption, $sk$, to allowed readers
- Integrity: owner gives public key pair for signing, $SK$, $PK$ to allowed writers

Problems:

- No detection for write-serial., freshness
- No proofs of violation
- Access control/key distrib. not scalable
Detection and proofs of violation for IWF

- **Attestations**

  - Proofs verifiable by any outside party
  - Non-repudiable signature scheme [Micali et. al.,’99]
  - Each party verifies attestation signatures
Auditing

- **Integrity**: users check attestations from cloud

- **W and F**: Owner does probabilistic auditing
  - Time divided in epochs (e.g., day)

  
<table>
<thead>
<tr>
<th></th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5</td>
<td>1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

  - Only owner and authorized users know in which epochs a block is audited
During the epoch

Data owner

Cloud

Users

B2

get...

..., cloud-get-attestation

put ...

cloud-put-attestation

Users

Cloud

B4

put...

..., cloud-put-attestation

put ...

cloud-put-attestation

Users

Data owner

cloud-get-attestation
cloud-put-attestation
cloud-put-attestation
cloud-put-attestation
At the end of epoch

- For the blocks to audit:
  - Owner requests all cloud-attestations from the cloud
  - Audits attestations from clients and from cloud
  - Audit guarantees write-serial. and freshness for entire epoch
Attestation Structure

- Hashed and signed by cloud:
  - “CLOUD GET ATTEST.”
  - BLOCK ID
  - VERSION NO.
  - ...

- Hashed and signed by user with SK:
  - “USER PUT ATTEST.”
  - BLOCK ID
  - NEW VERSION NO.
  - ...

- Hashed and signed by cloud:
  - “CLOUD PUT ATTEST.”
  - BLOCK ID
  - NEW VERSION NO.
  - ...

Detection: signature does not verify

Proof of violation: attestation
Detection: Fork in sequence of put attestations
Proof of violation: the forked sequence of attestations

Hashed and signed by cloud

version 4, hash: xd242
version 5, hash: xae97
version 5, hash: x3166
fork
Freshness

- chain hash = hash (data in current attestation, previous attestation)
- Detection: attestations do not chain correctly
Detection: attestations do not chain correctly

Cloud

Users

put: blockid 5, hash x18, ...

A1 = (cloud-put-attestation, blockid 5, version 1, hash x18, ...)

put: blockid 5, hash x22, ...

A2 = (cloud-put-attestation, blockid 5, version 2, hash x22, h(A1, data in A2), ...)

get: blockid 5

A3 = (cloud-get-attestation, blockid 5, version 1, hash x18, h(A2, data in A3)? Detected!

Proof of violation: broken chain of attestations
Implementation

- C#, Windows Azure:
  - Storage component: blobs and queues
  - Compute component: web and worker roles

- Four modules: owner, user, cloud, auditor
- .NET crypto tools: AES, SHA-1, RSA
Evaluation

- What is the overhead at users/cloud?
  - Latency/throughput
- What is the workload of the owner?
  - Access control/auditing
User/server overhead

- Mostly from sign-verify of attestations

Delay added per request: 30 ms at server, 40 ms at user
- Can optimize: e.g., batch many attestations in one signature using a Merkle hash

Throughput scales roughly linearly at server
Owner work

- Two offline tasks:
  - **Key distrib.**: for a widely-used software with > 5000 developers, membership changes take <1.6 sec/month
  - **Auditing** cost is modest and parallelizable

- Detection probability increases exponentially in no. of epochs of violation

- 4 min for $10^8$ attestations
Related work

- **Secure file/storage systems** (e.g., SiRiUS, SUNDR, Plutus):
  - No proofs of violation
  - No W and F detection due to different model
  - Access control not as scalable

- **Proofs of retrievability/possession** (e.g., POR, HAIL)

- **Byzantine fault tolerance** (e.g., BFT)
Conclusions

- **CloudProof** is a secure storage system for the cloud:
  - Detection of WF via auditing
  - Proofs of violation for IWF via attestations
  - Scalable access control using broadcast encryption

Thanks!