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

Usenix ATC'11

Motivation

- Cloud storage provides extensive resources, scalability, and reliability

A main concern is security

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

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

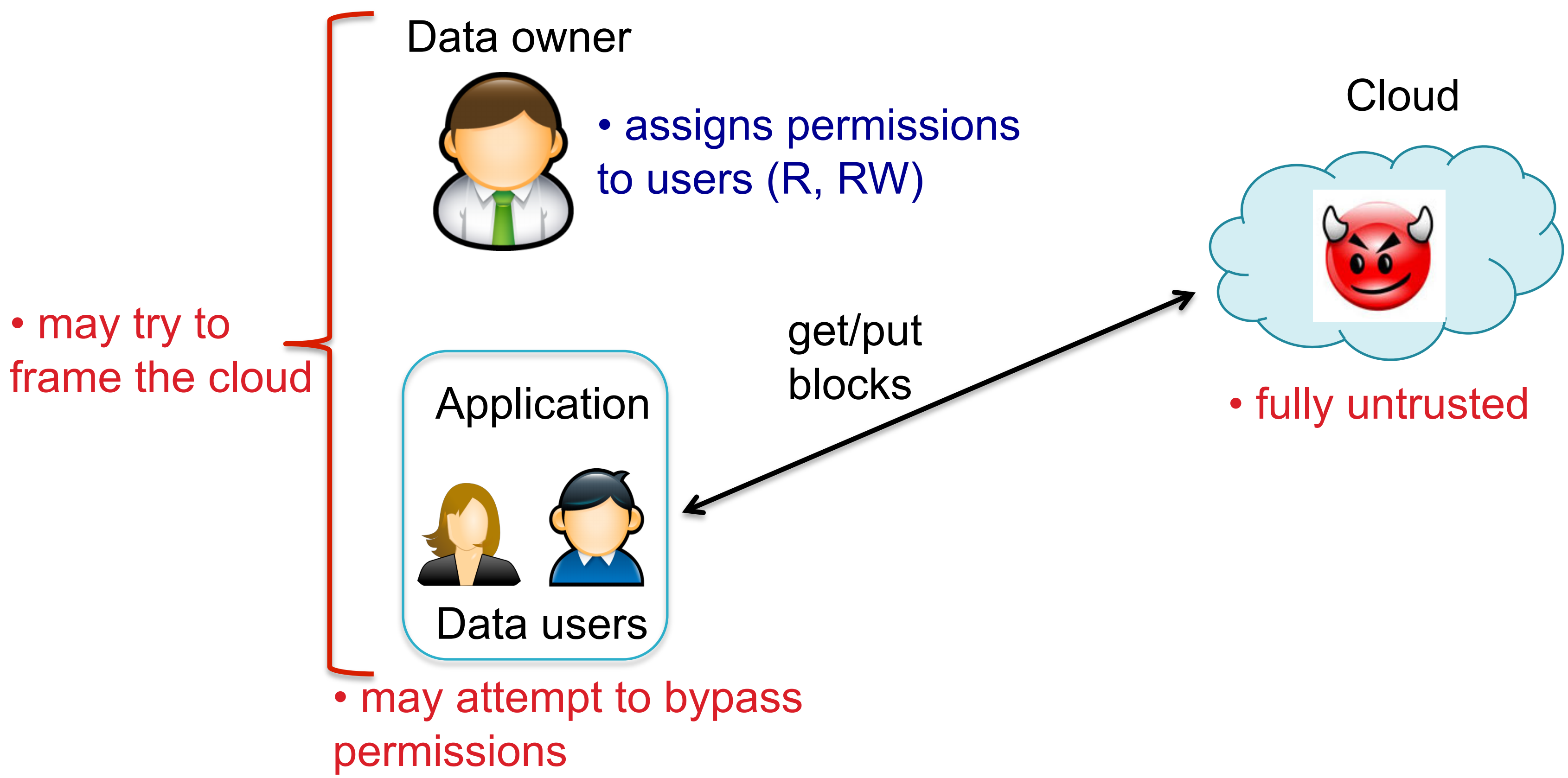
Problem: cloud services do not guarantee security in SLAs

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)
Cloud
• may try to frame the cloud
get/put blocks
• fully untrusted
Application
Data users
• may attempt to bypass permissions

Strawman

Block

Enc_{sk}[content]

Version no.

Sig_{SK}[encr. content]

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:

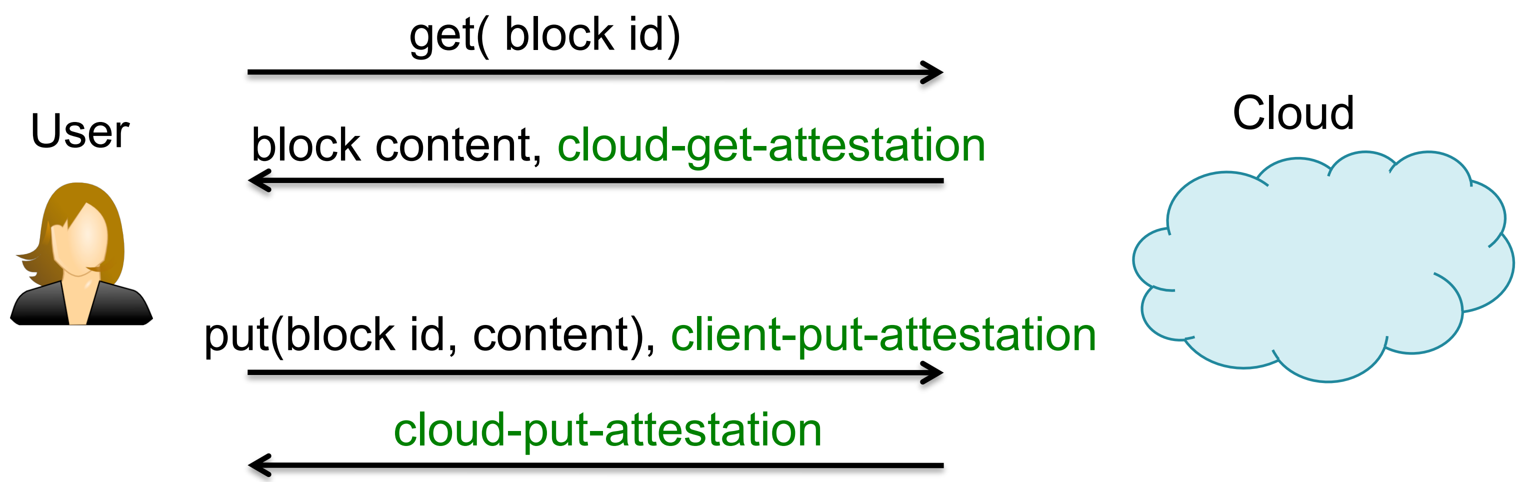
- No detection for write-serial., freshness
- No proofs of violation
- Access control/key distrib. not scalable

in this talk

see paper

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

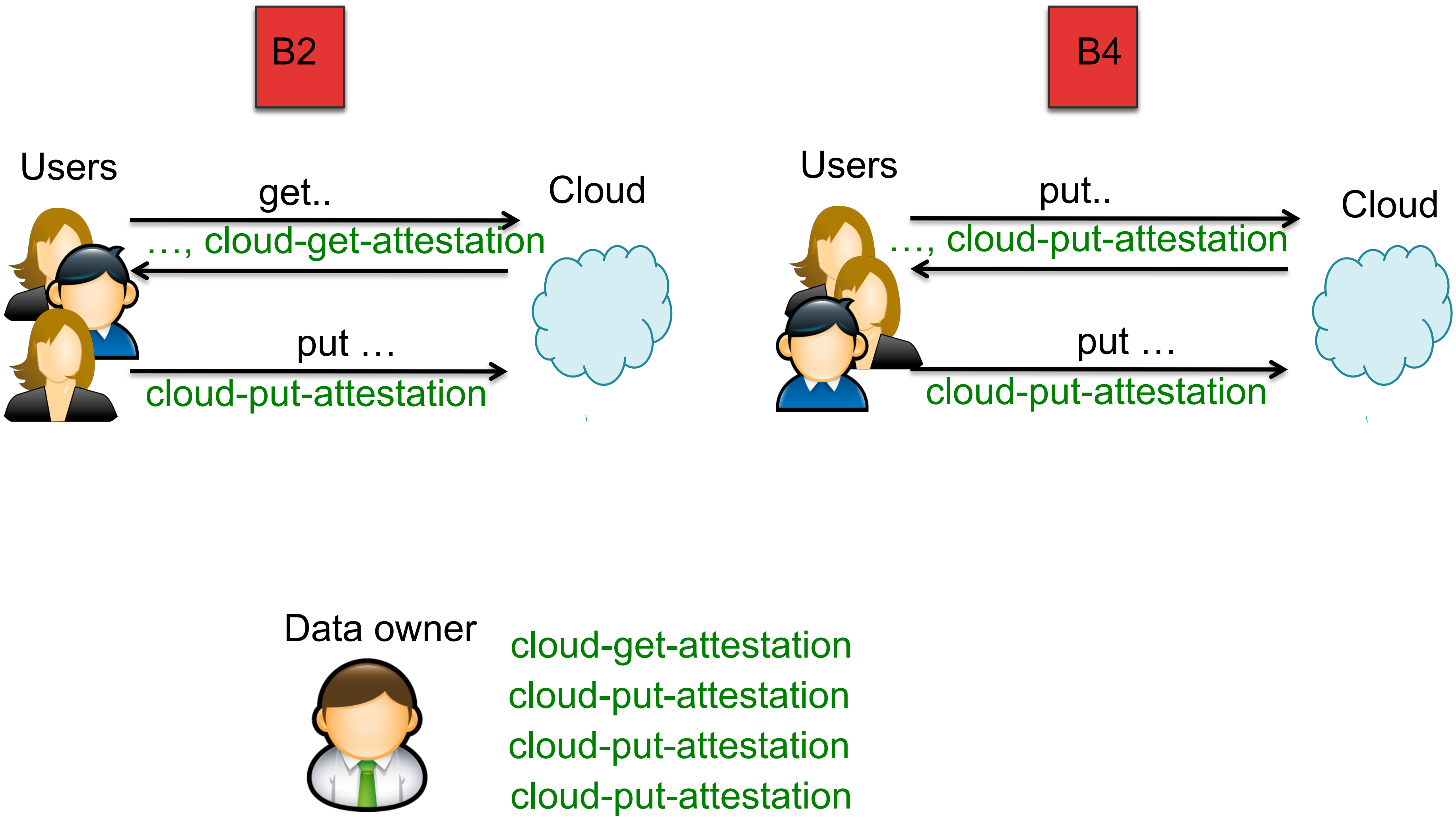
- Integrity: users check attestations from cloud

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

	B1	B2	B3	B4	B5
probability:	0.5	1	0.1	0.2	0.2

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

During the epoch



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.	. . .

Integrity

Hashed and signed by cloud

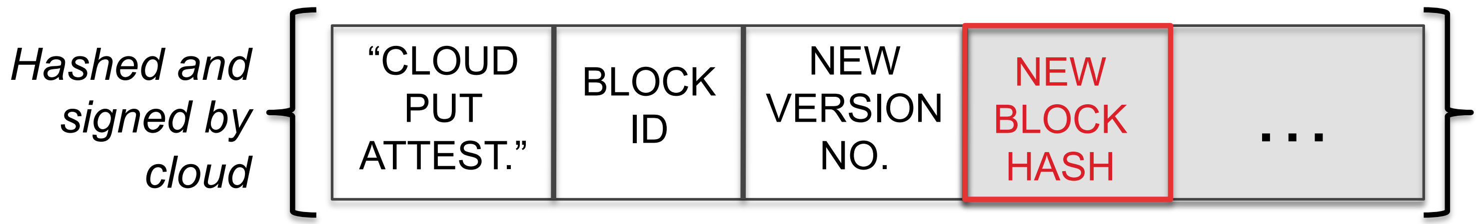
"CLOUD GET ATTEST."	BLOCK ID	BLOCK VERSION NO.	BLOCK HASH	. . .

Block

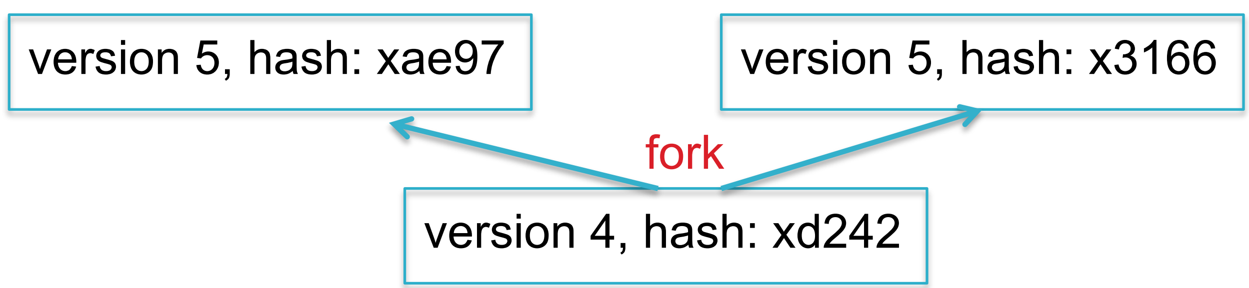
Enc_{sk}[content] version no.
Sig_{SK}[encr. content]

- Detection: signature does not verify
- Proof of violation: attestation

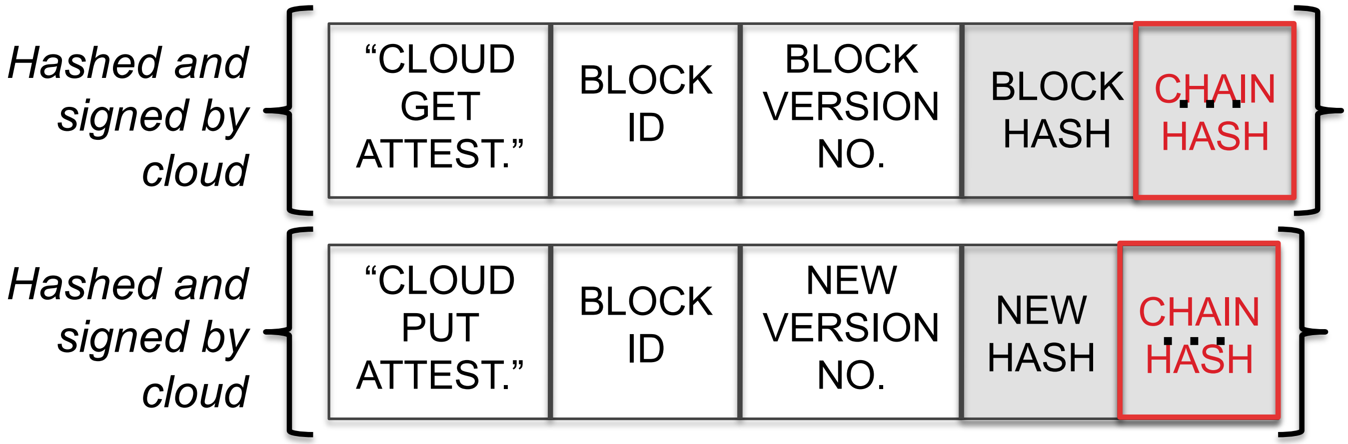
Write-serializability



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



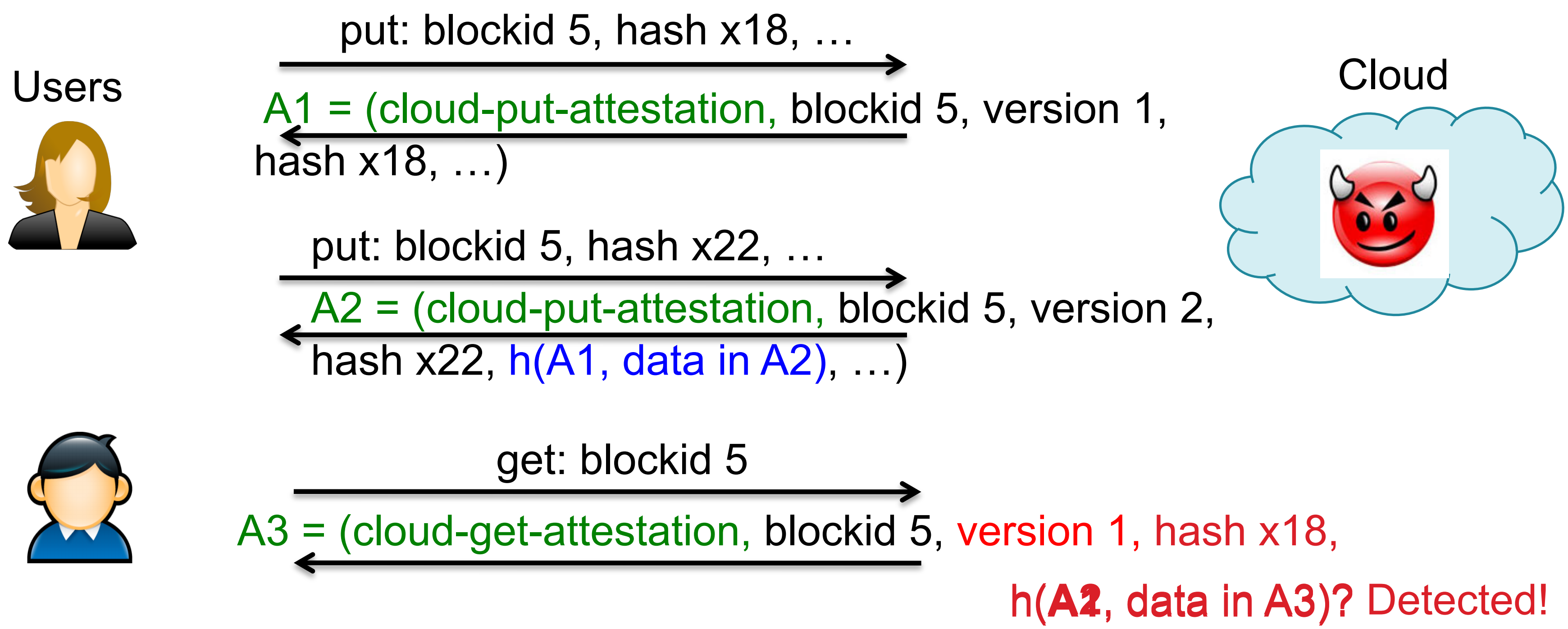
Freshness



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

Freshness (cont'd)

- Detection: attestations do not chain correctly



- Proof of violation: broken chain of attestations

Implementation

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

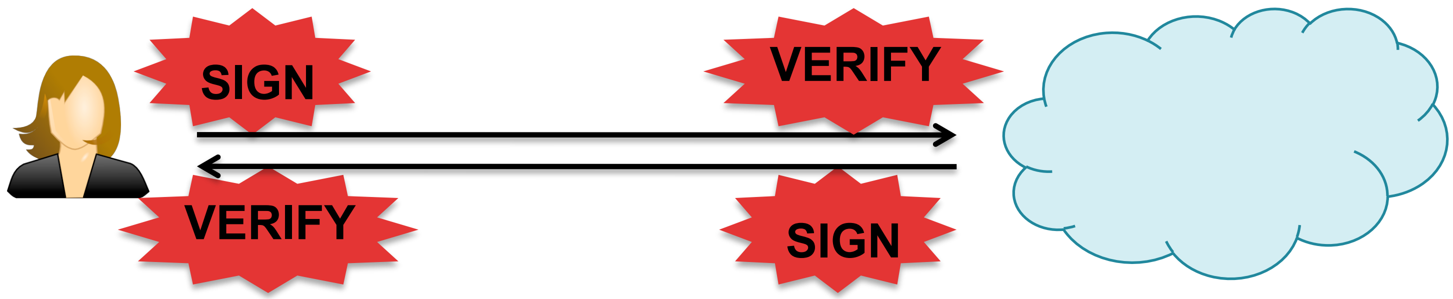
- Four modules: owner, user, cloud, auditor
- .NET crypto tools: AES, SHA-1, RSA

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

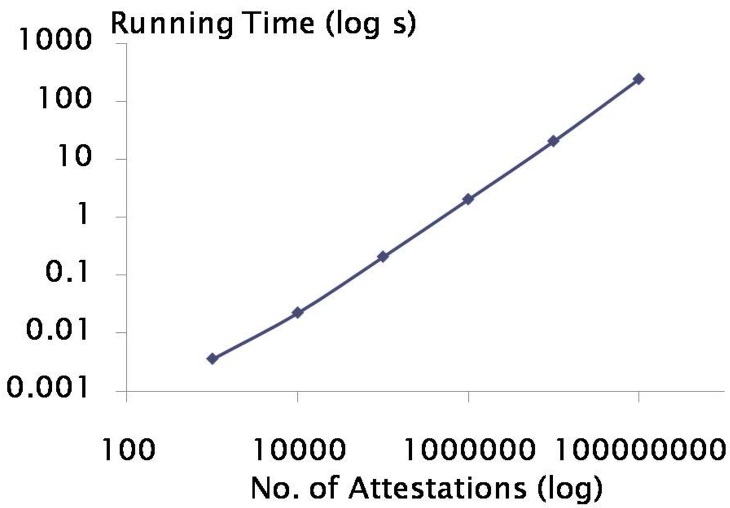


- Delay added per request: 30 ms at server, 40 ms at user
- Can optimize: e.g., batch many attestations in one signature using a Merkle hash

- 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



• 4 min for 10^8 attestations

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

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!