Social Networking with Frientegrity:

Privacy and Integrity with an Untrusted Provider

Ariel J. Feldman

Princeton ⟶ UPenn

Joint work with:

Aaron Blankstein, Michael J. Freedman, and Edward W. Felten

PRINCETON UNIVERSITY

Online social networks are centralized

Pro: Availability, reliability, global accessibility, convenience

Con: 3rd party involved in every social interaction

Must trust provider for
confidentiality & integrity

Threats to confidentiality

- Theft by attackers
- Accidental leaks
- Privacy policy changes
- Government pressure

Ars Technica. Mar. 11, 2011

EFF. Apr. 28, 2010

PC World. Dec. 6, 2011

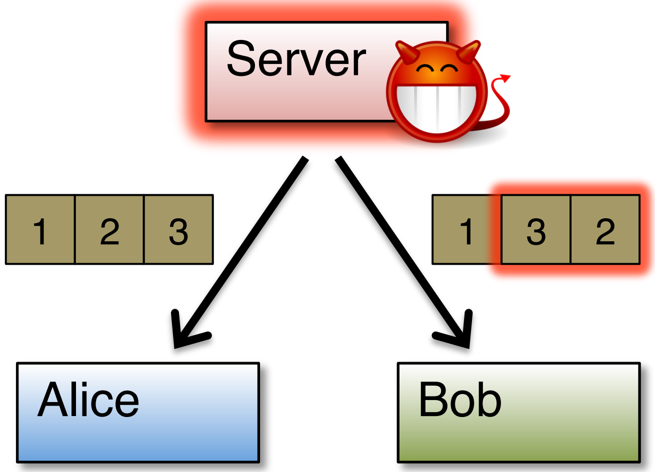
WSJ. Feb. 22, 2012

Google Transparency Report Jan. – Jun. 2011

Threats to integrity

Simple: Corrupting messages

Complex: Server equivocation



Equivocation in the wild:

(e.g to disguise censorship)


Location: Blog > Song Stone Boy " Why am I left Sina microblogging

Why do I leave Sina microblogging

Jul 14 2011, 22:29 Category: Social , Technology Tags: microblogging reading: 17,421 Comments: 82

http://songshinan.blog.caixin.com/archives/22322 (translated by Google)

Limits of prior work

1. Cryptographic

Don’t protect integrity

2. Decentralized

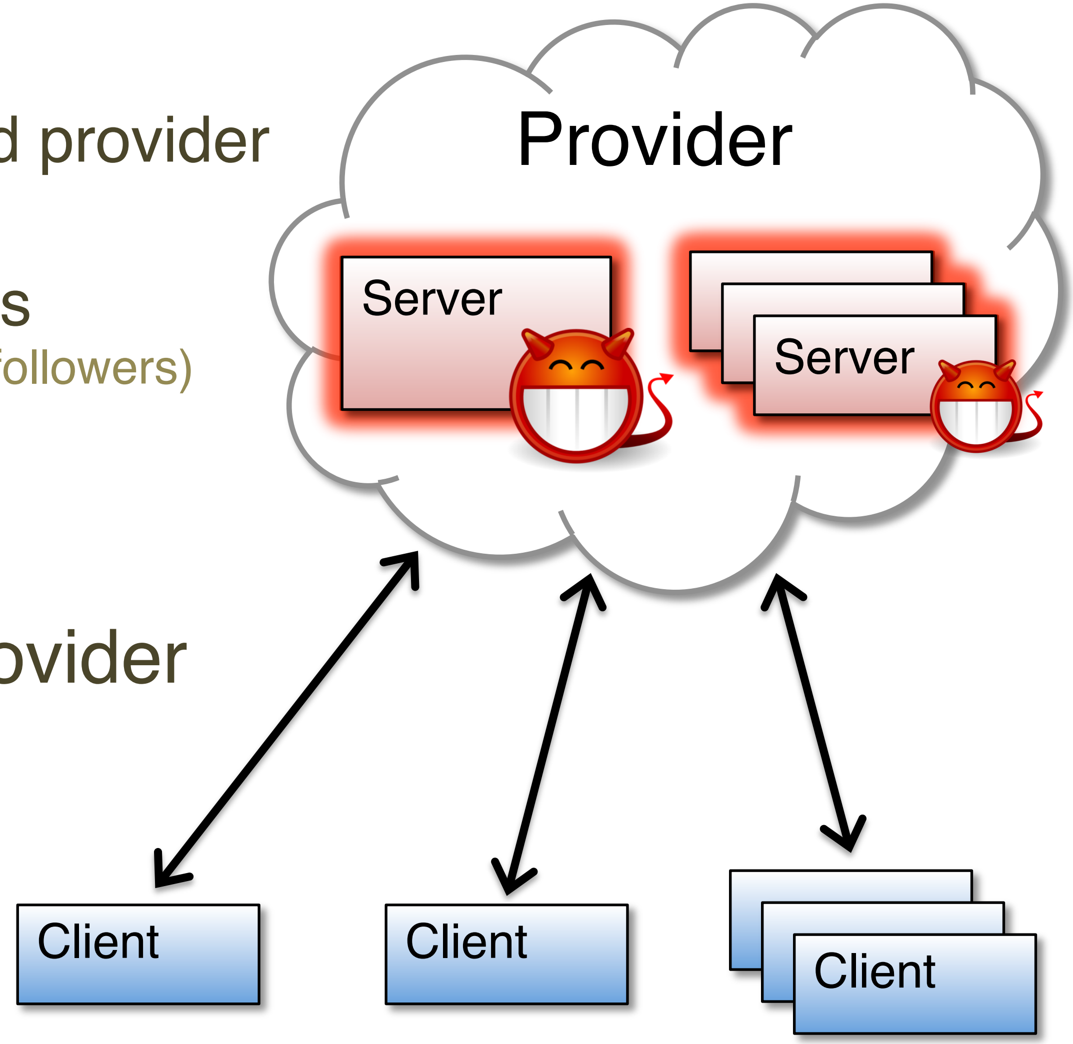
Run your own server	OR	Trust a provider
(sacrifice availability, convenience, etc.)		(who you may not know either)

Frientegrity's approach

Benefit from a centralized provider

Support common features
(e.g. walls, feeds, friends, FoFs, followers)

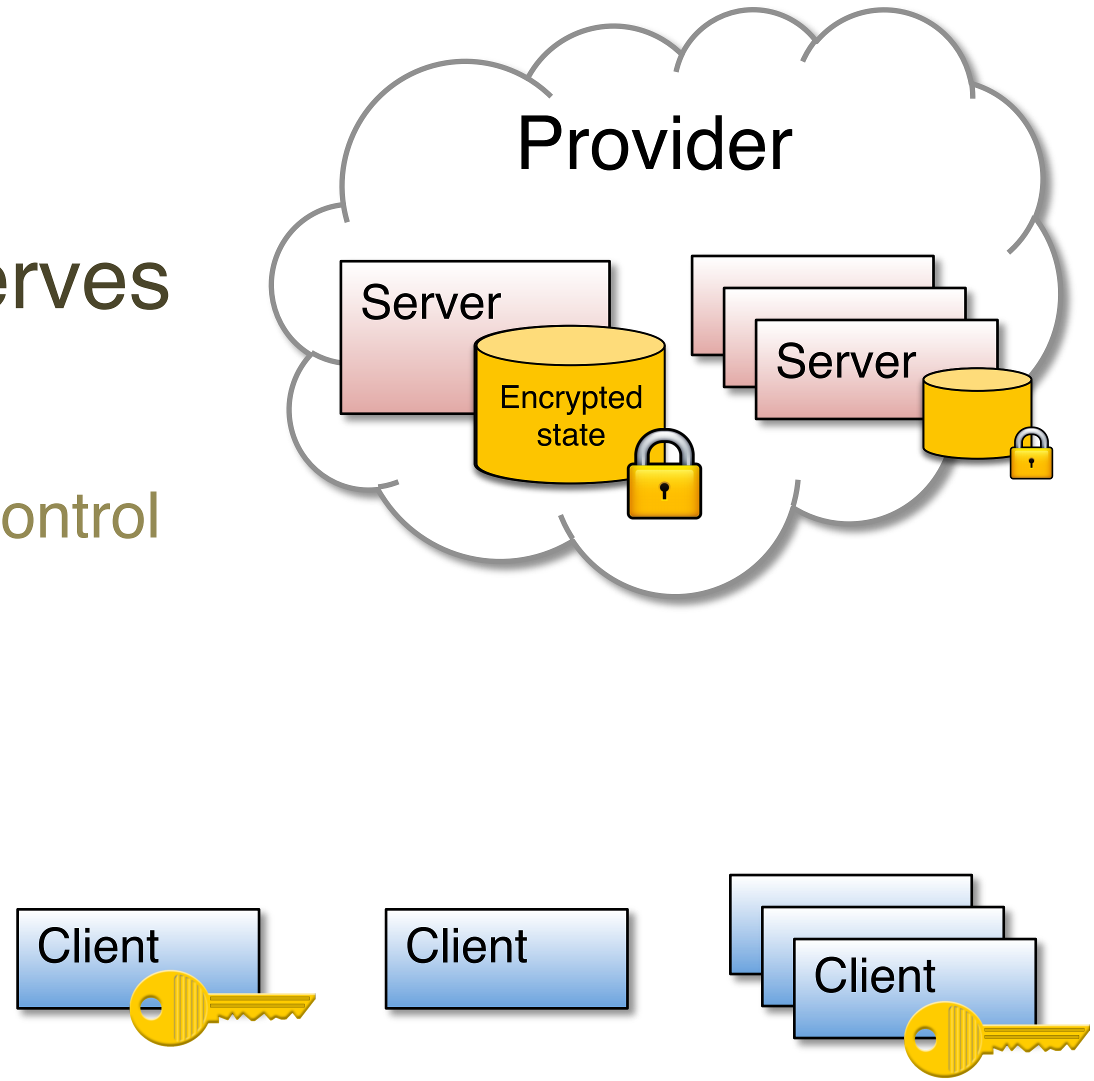
Assume untrusted provider



Enforce confidentiality

Provider only observes encrypted data

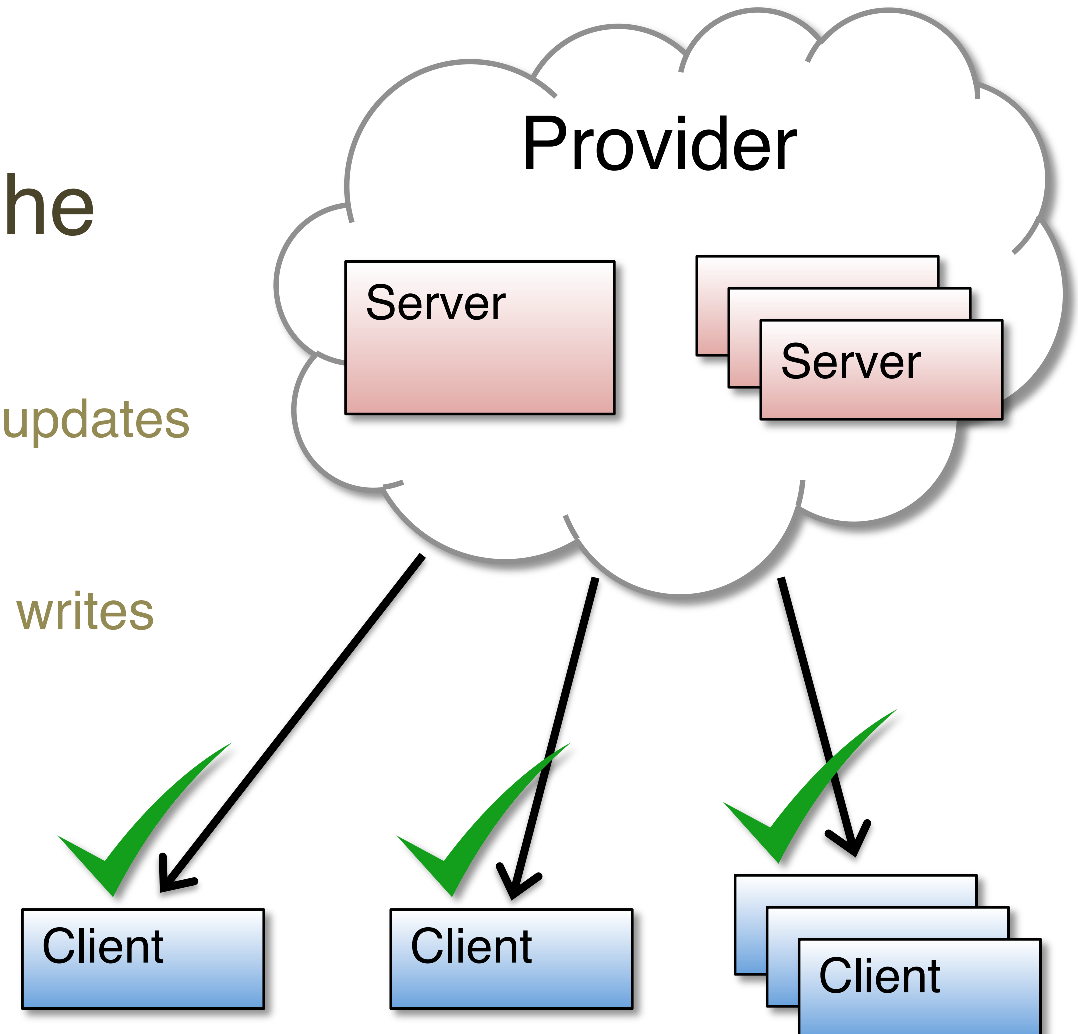
(Need dynamic access control and key distribution)



Verify integrity

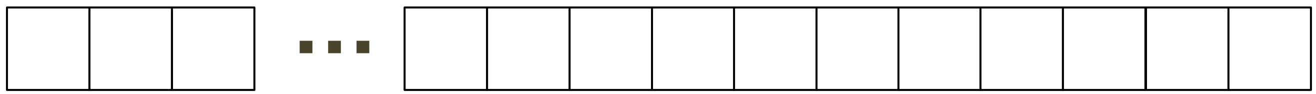
Clients verify that the provider:

- Hasn't corrupted individual updates
- Hasn't equivocated
- Enforced access control on writes



Scalability challenges

Long histories; only want tail



Don’t verify whole history each time

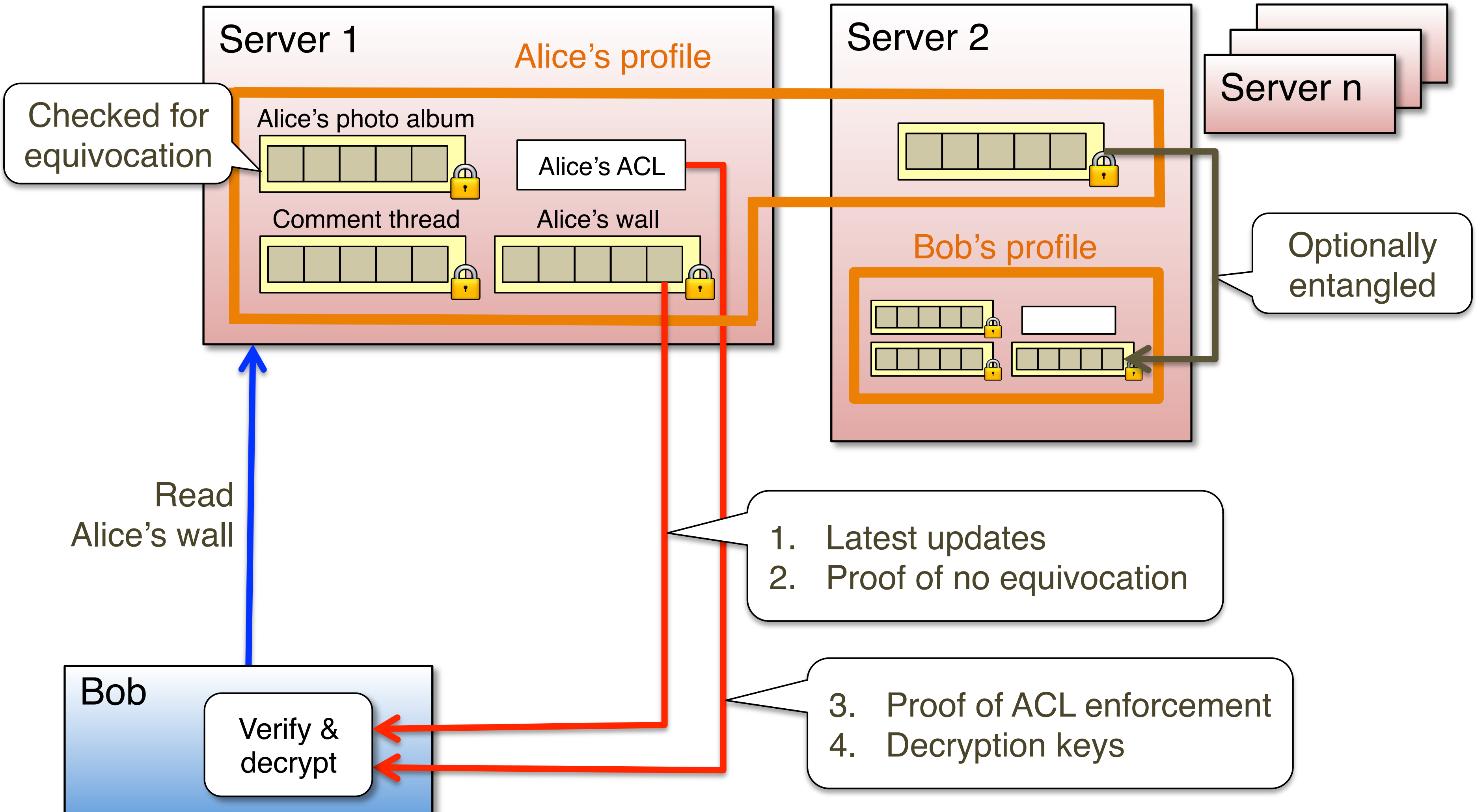
Many objects (walls, comment threads, photos, etc.)

Support sharding

Many friends and FoFs

O(log n) “(un)friending”

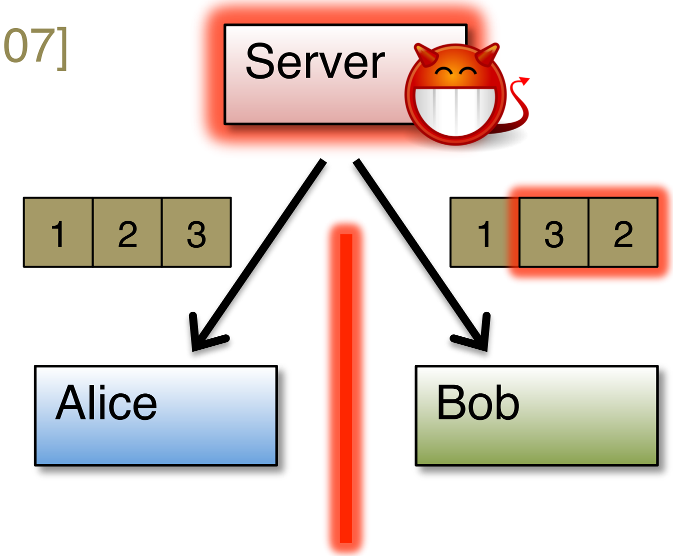
Frientegrity overview



Detecting equivocation

Enforce fork* consistency [LM07]

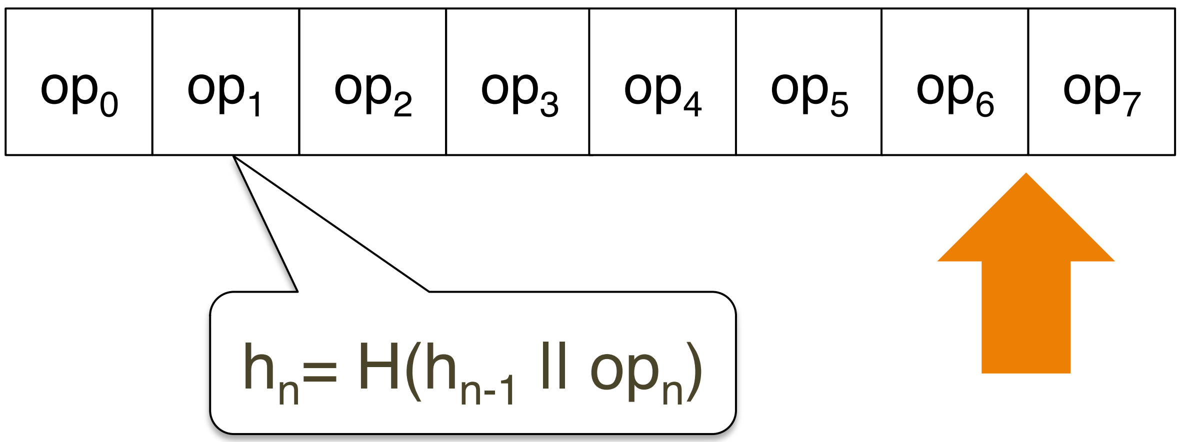
- Honest server: linearizability
- Malicious server: Alice and Bob detect equivocation after exchanging 2 messages
- Compare histories



Provider can still fork the clients, but can't unfork

Comparing histories

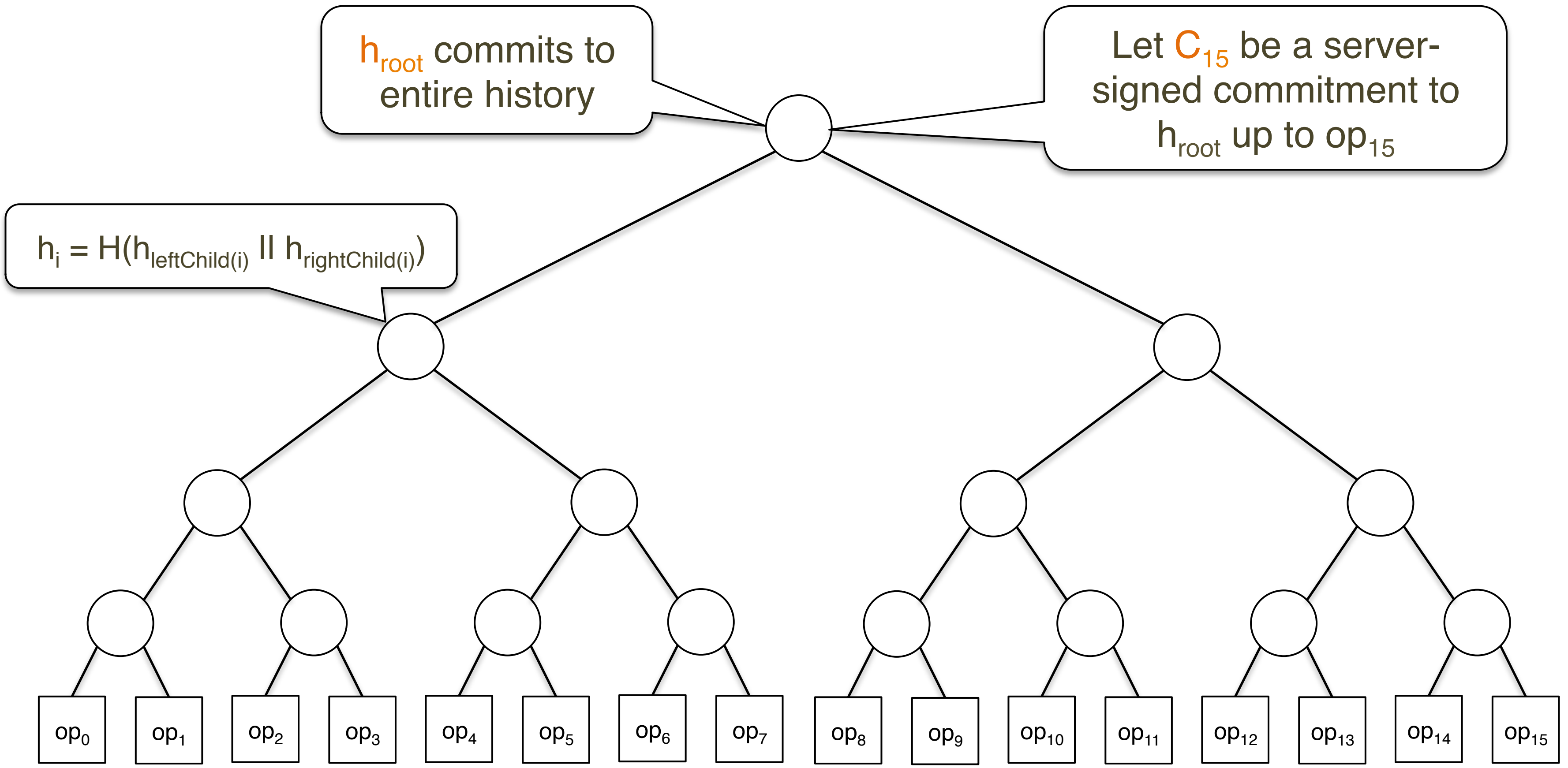
Previously: use a hash chain



Hash chains are O(n)

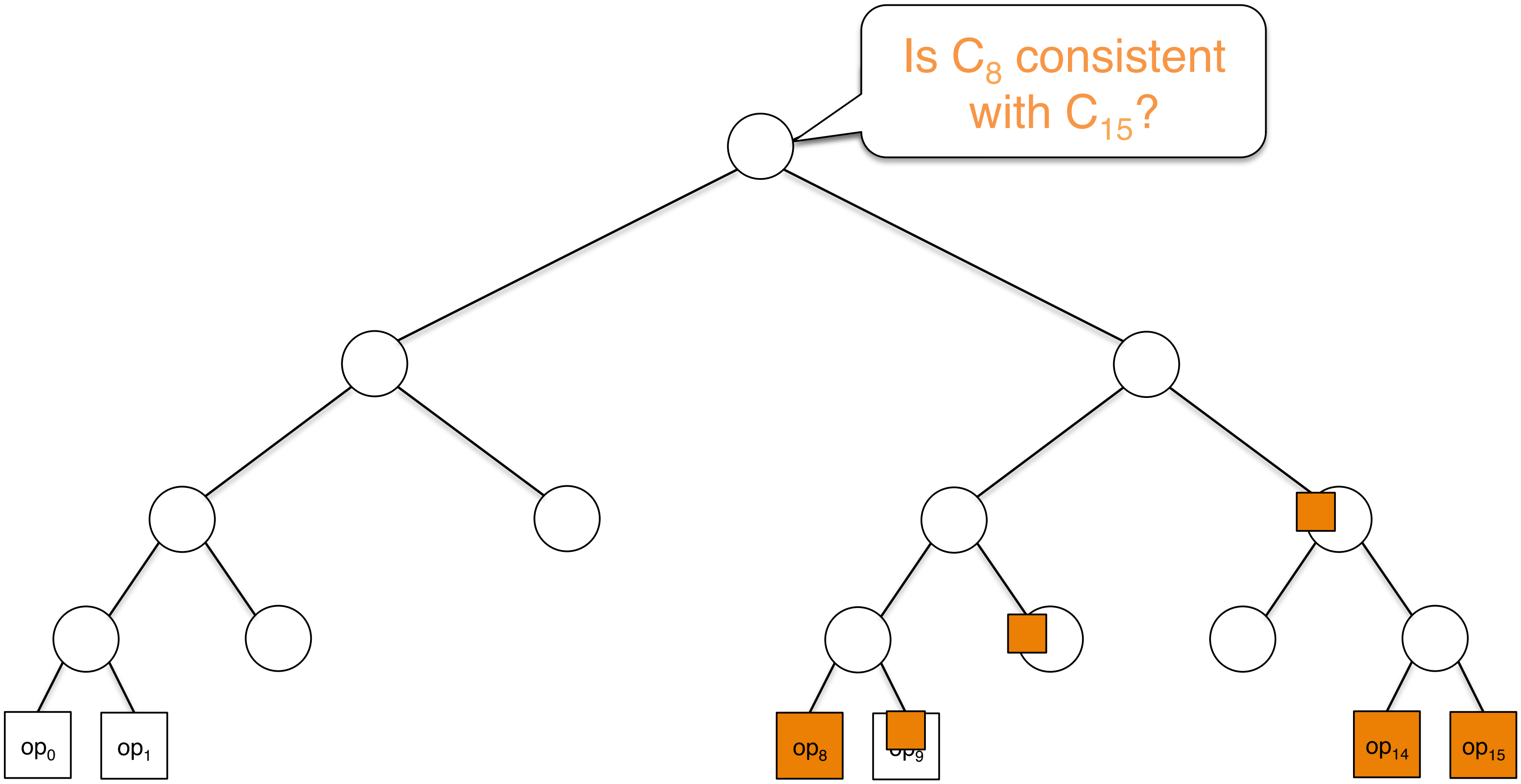
(and must download the whole history)

Objects in Frientegrity

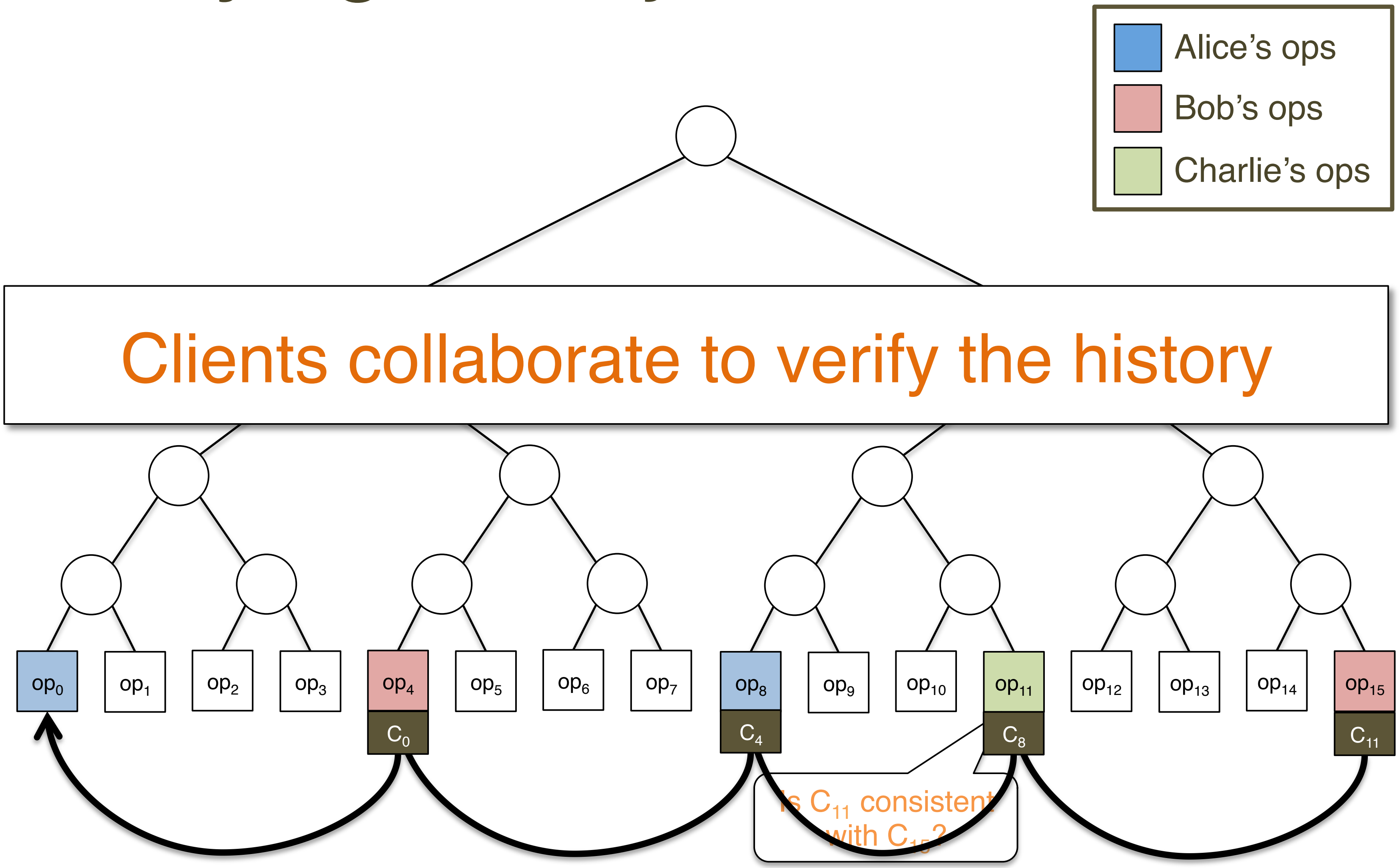


History tree [CW09]

Objects (cont.)

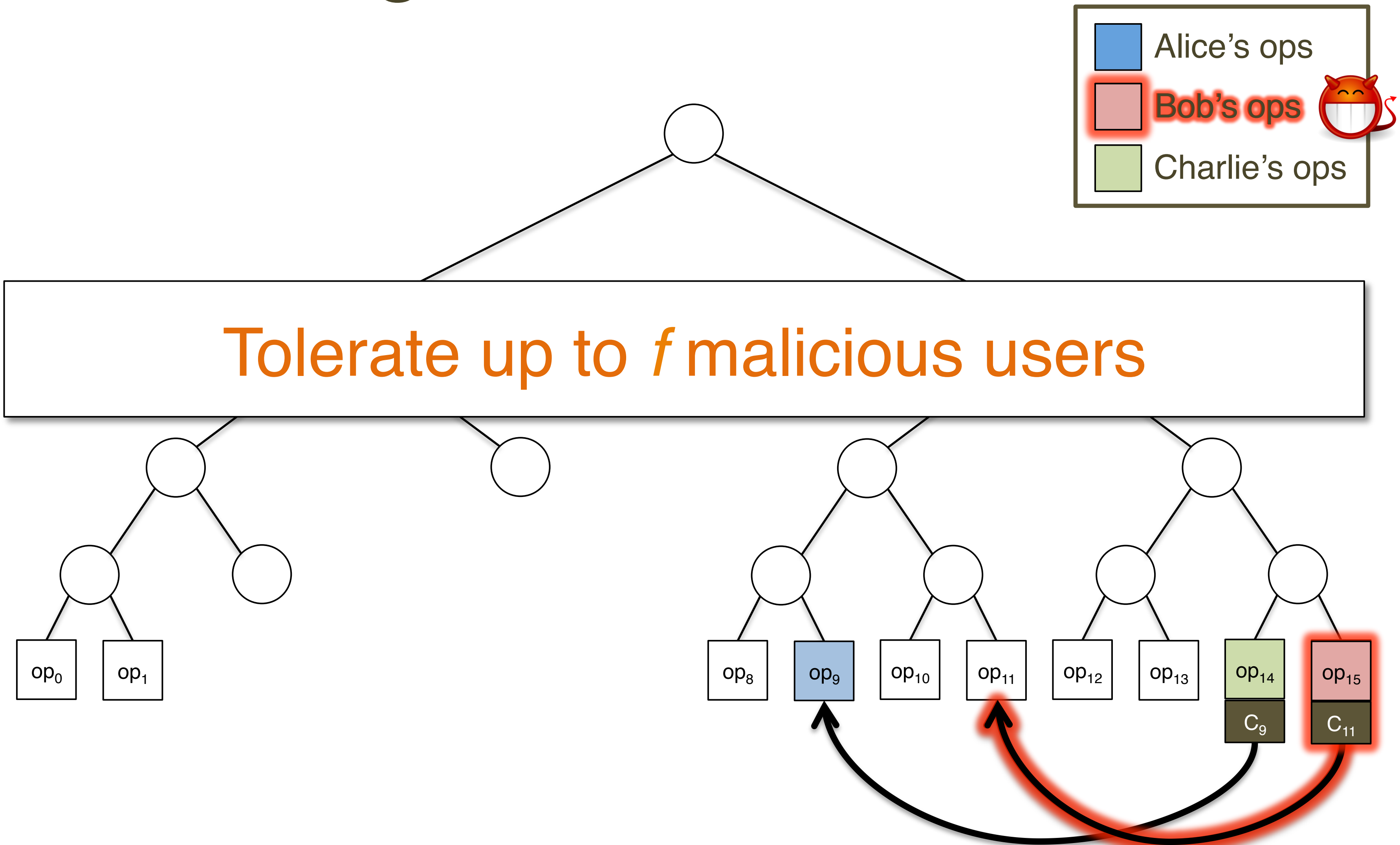


Verifying an object

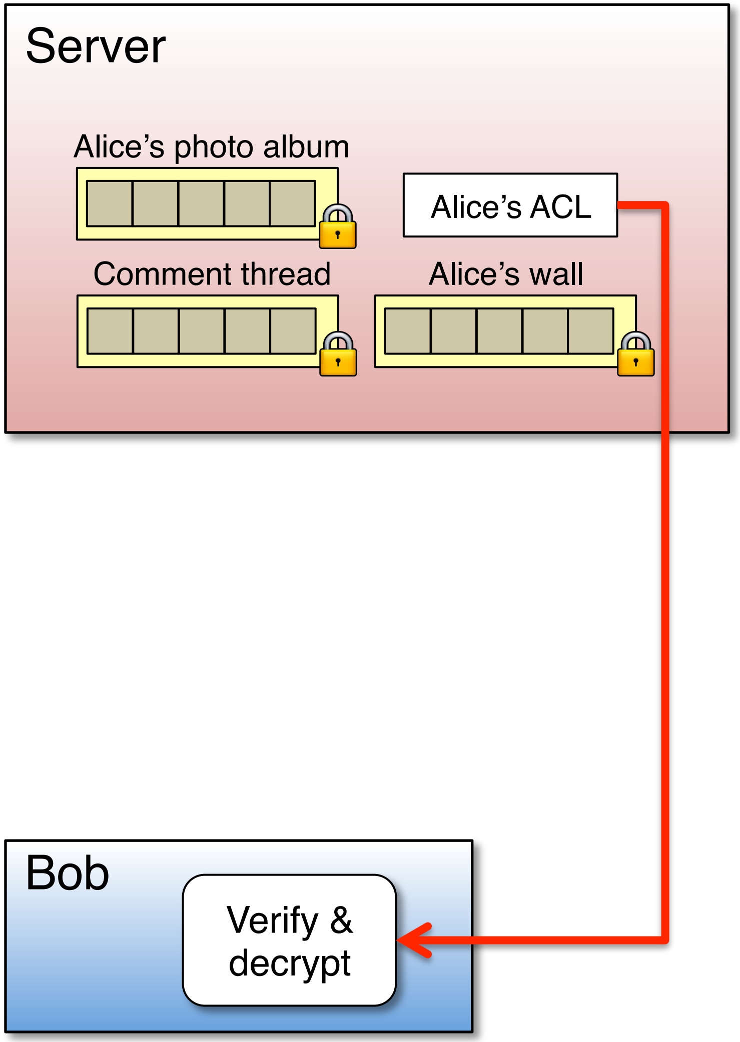


Clients collaborate to verify the history

Tolerating malicious users



Access control

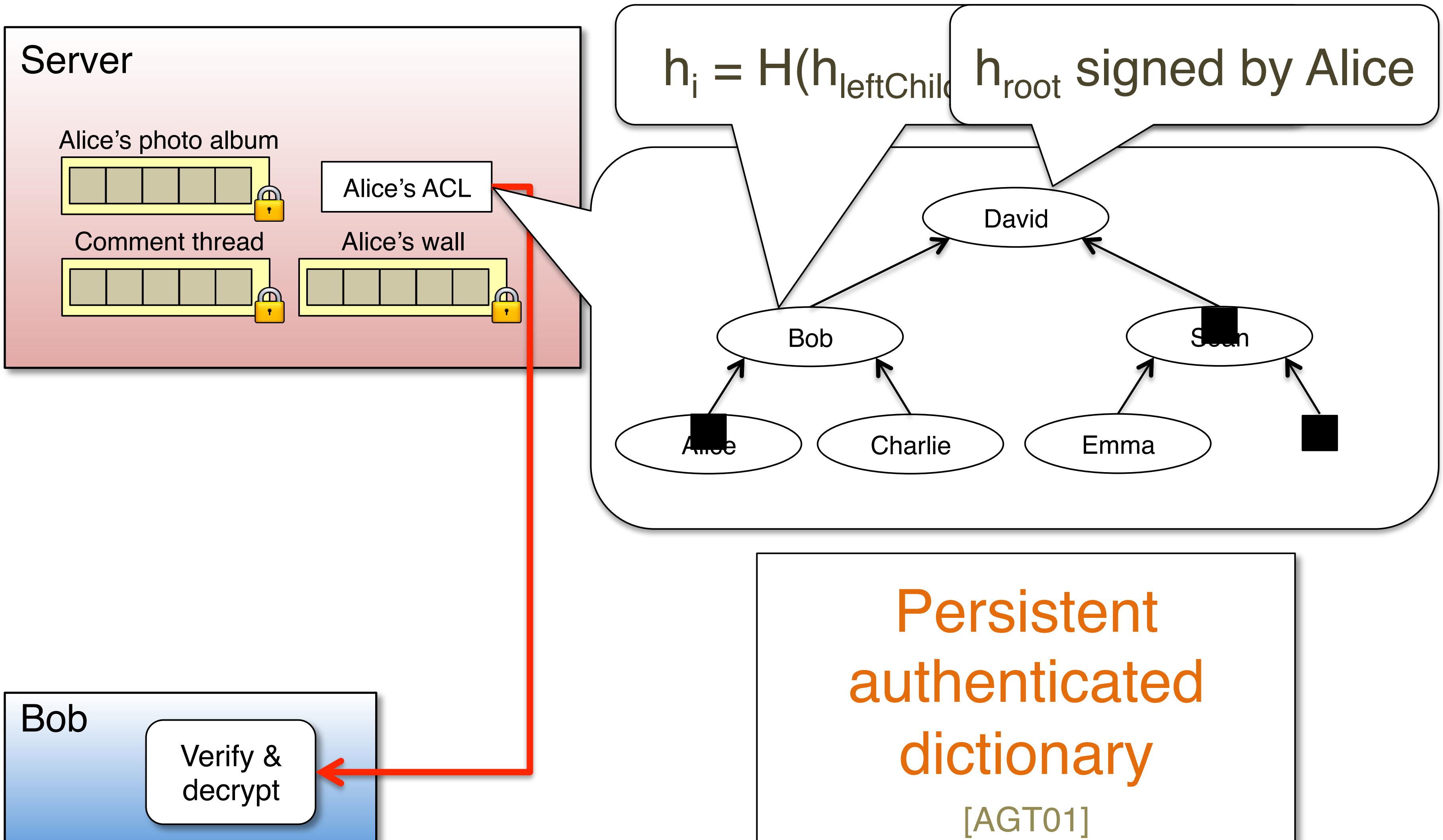


Prove ACL enforcement

Efficient key distribution

O(log n) "(un)friending"

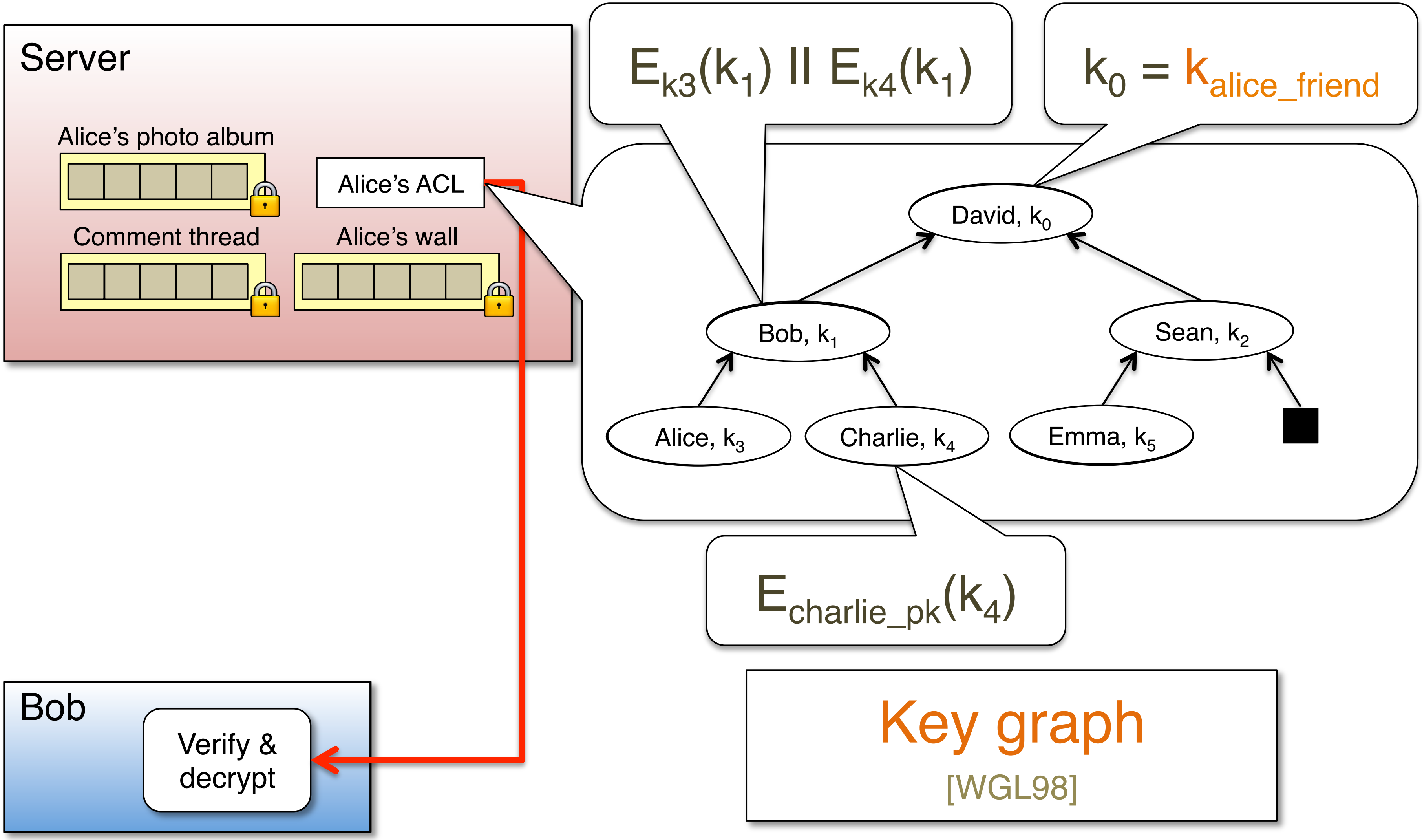
Proving ACL enforcement



Persistent authenticated dictionary

[AGT01]

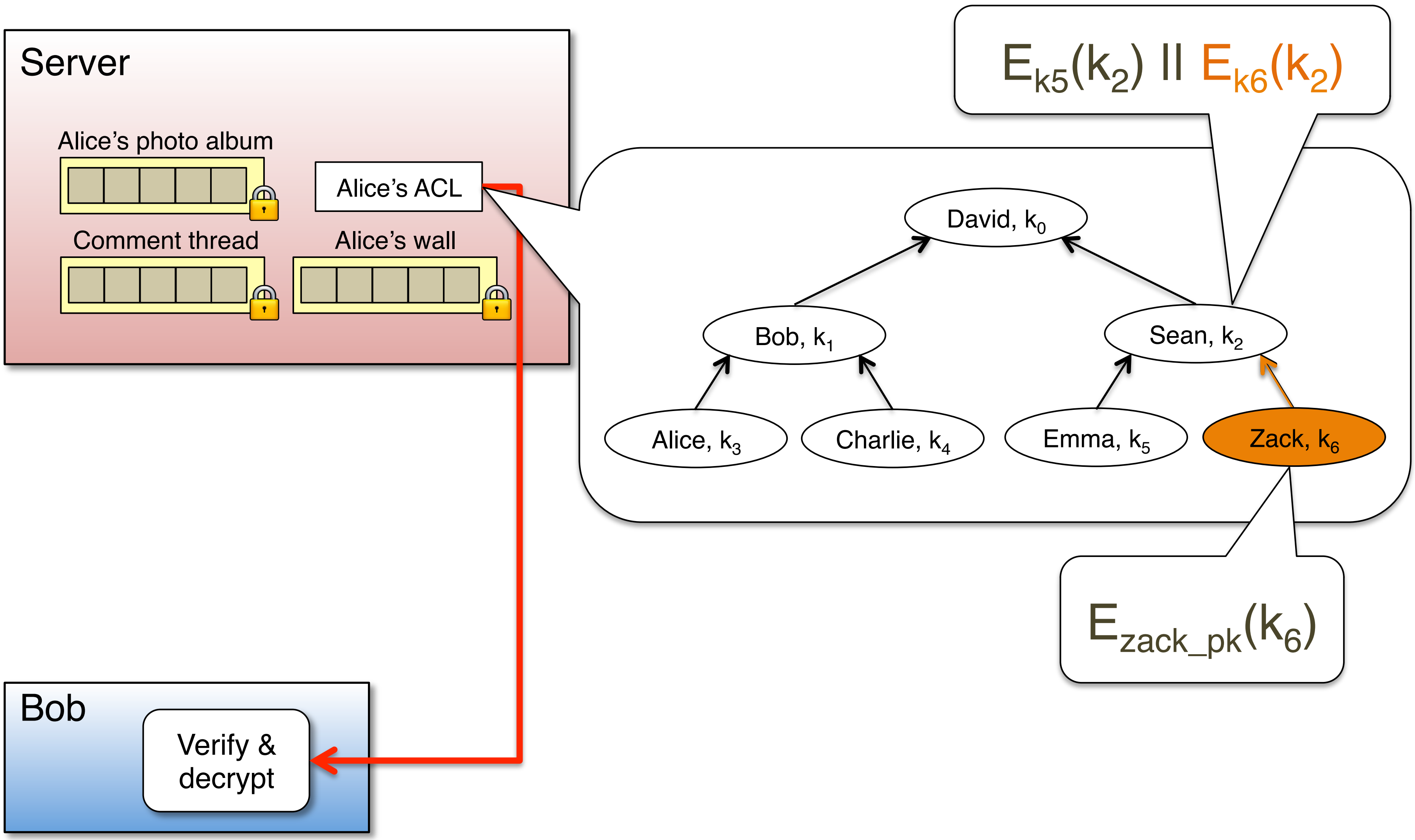
Efficient key distribution



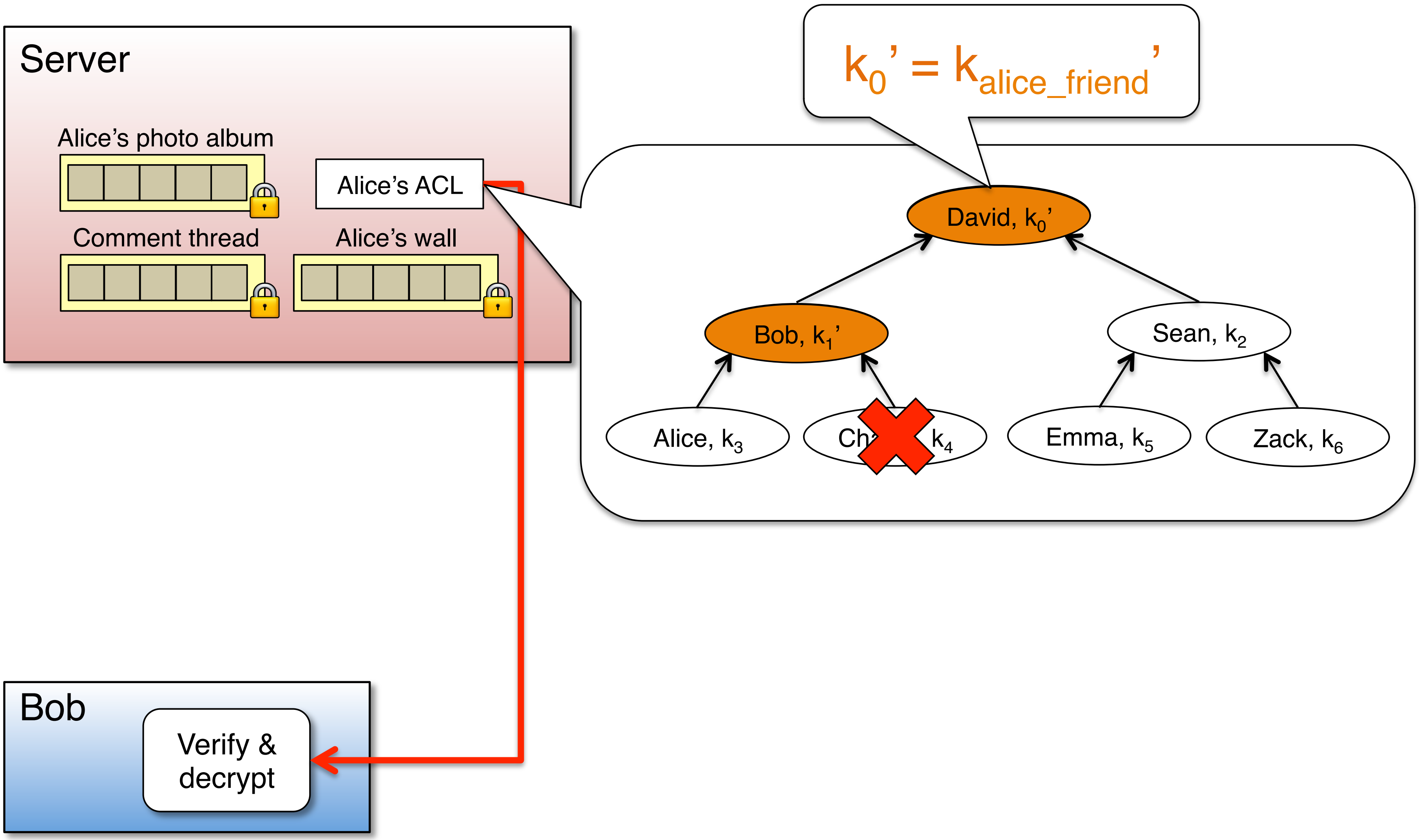
Key graph

[WGL98]

Adding a friend



Removing a friend



Efficient enough in practice?

Setup

- Java client & server
- Simulate basic Facebook features (each user has wall & ACL)
- 2048-bit RSA sign & verify batched via spliced signatures [CW10]
- Experiments on LAN (8-core 2.4 GHz Intel Xeon E5620s, Gigabit network)

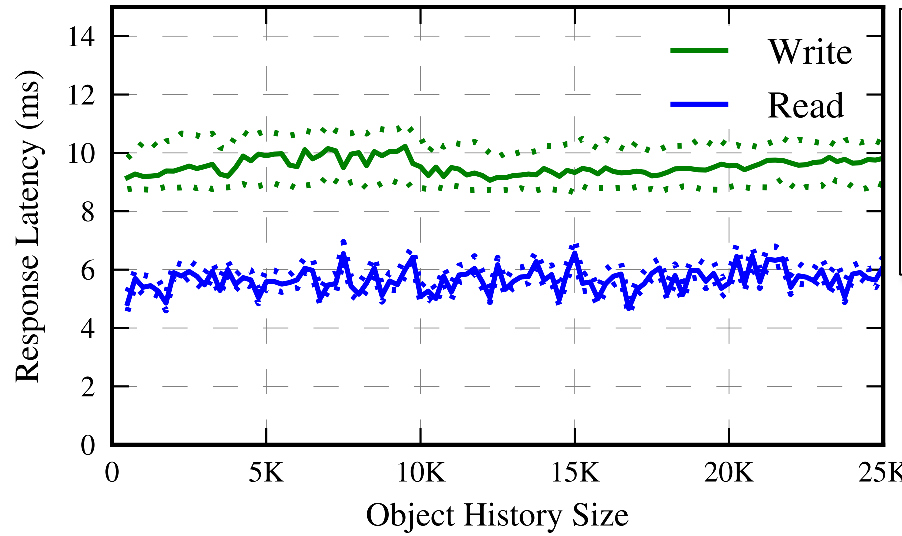
Measurements

- Latency of reads & writes to objects
- Latency of ACL changes
- Throughput (in paper)
- Effect of tolerating malicious users

Object read & write latency

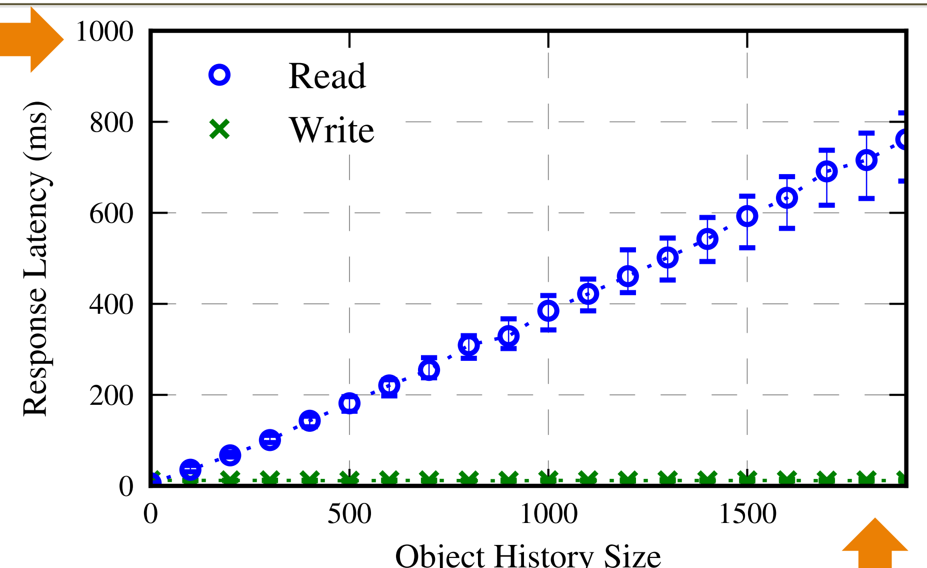
Frientegrity

(collaborative verification)

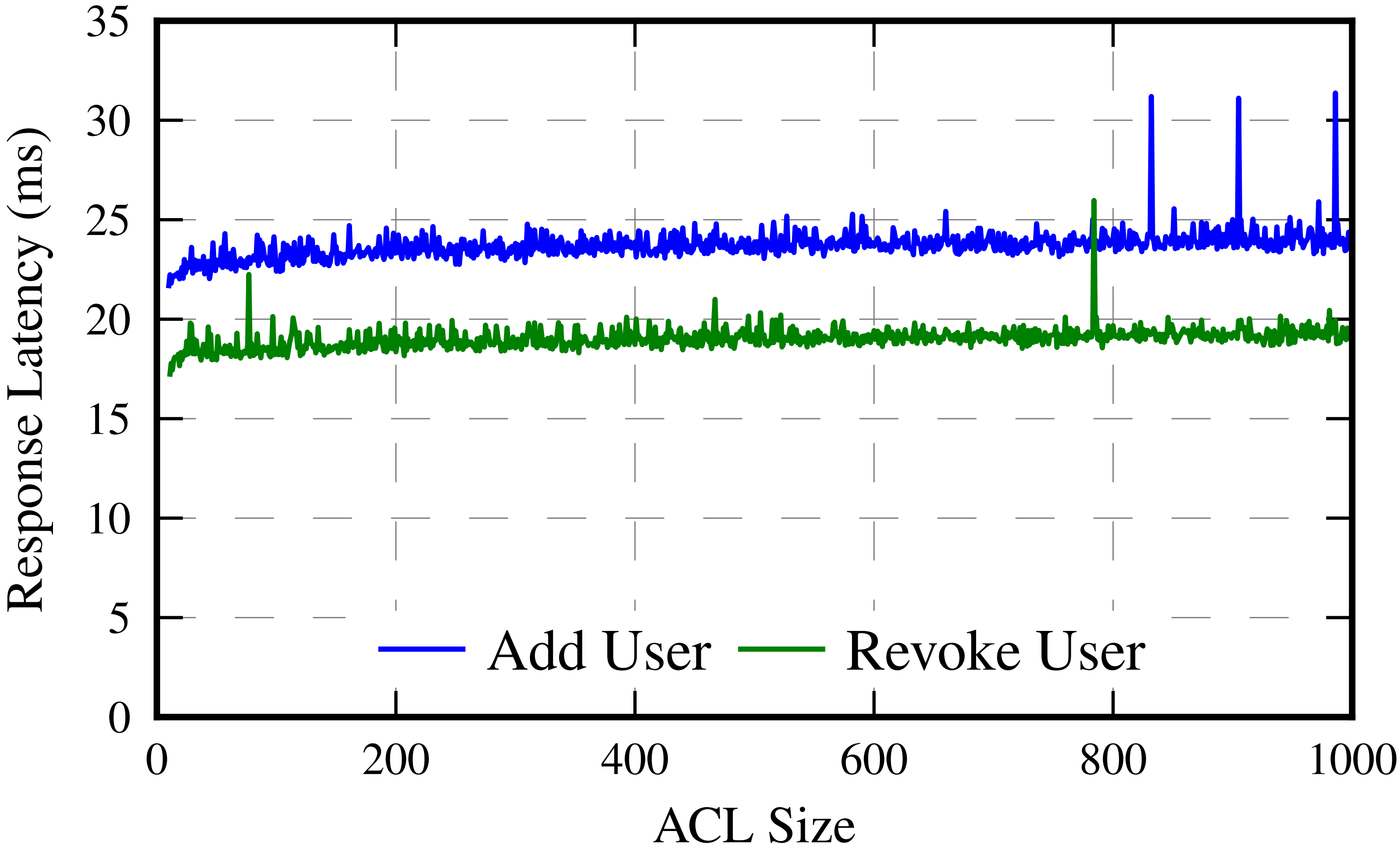


Constant cost of signatures dominates

Hash chain

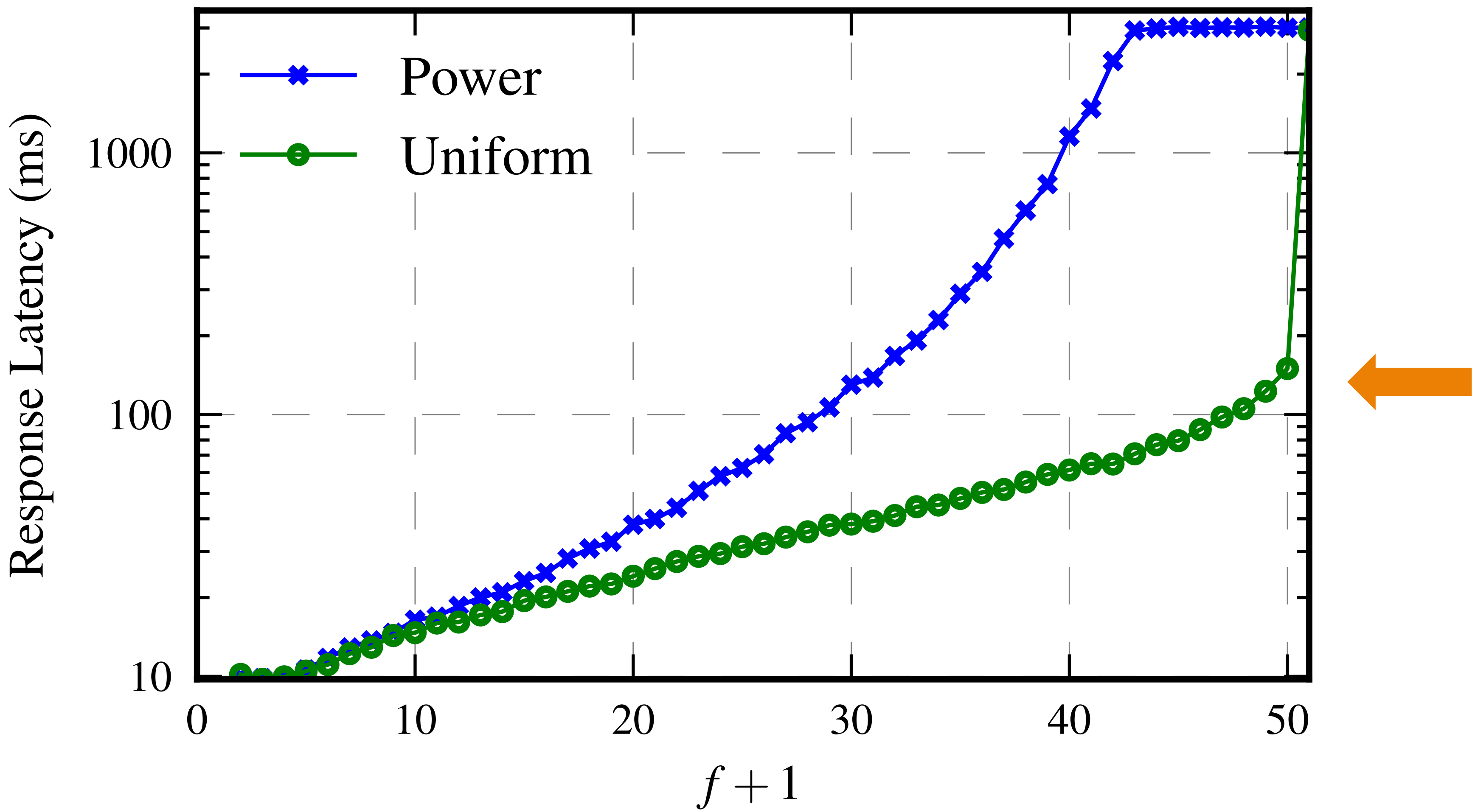


Latency of ACL changes



Tolerating malicious users

- 50 writers
- 5000 operations



Summary

Both confidentiality & integrity need protection

Benefit from centralization, but provider is untrusted

Clients collaborate to defend against equivocation

Scalable, verifiable access control & key distribution

Thank you

Questions?

http://arifeldman.com
ariel.feldman@cis.upenn.edu