Privacy and Integrity in the Untrusted Cloud

Edward W. Felten

felten@cs.princeton.edu

PRINCETON UNIVERSITY

Joint work with:
Ariel J. Feldman, William P. Zeller,
Aaron Blankstein, Michael J. Freedman

Cloud deployment: Pro & Con

For user-facing apps:

Google docs, Google Calendar, Microsoft Office Live, facebook, twitter, Google+

Pro: Availability, reliability, global accessibility, convenience

Con: Users give up control over their data

Must trust provider for confidentiality & integrity

Threats to confidentiality

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

Twitter settles with FTC over security breaches

By Jacqui Cheng | Published 11 months ago

Ars Technica. Mar. 11, 2011

APRIL 28, 2010 | BY KURT OPSAHL

Facebook's Eroding Privacy Policy: A Timeline

EFF. Apr. 28, 2010

TODAY @ PCWORLD

TECHNOLOGY | FEBRUARY 22, 2012, 9:00 P.M. ET

State AGs Target New Google Privacy Policy

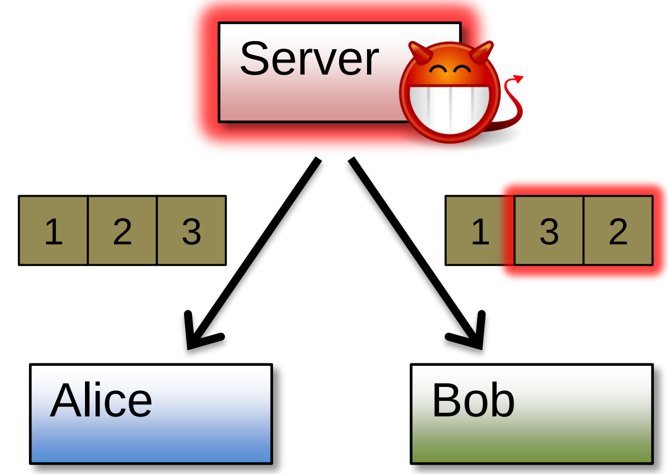
WSJ. Feb. 22, 2012

PC World. Dec. 6, 2011

Threats to integrity

Simple: Corrupting messages

Complex: Server equivocation



Does this happen? Yes!

(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)

Legal or market-based solution?

We're skeptical…

Users' limited information

- May not know what third party is doing

  (i.e. security is a "lemons market")
- May not find out until its too late
- Third party could change its behavior over time

Not enough to wait until damage is done

- Harm could be irreparable
- Quantifying harm is often hard

Our approach

Privacy & integrity by design:

- Benefit from cloud deployment
- Assume untrusted provider

Contributions:

- Practical cloud apps
- Preventing confidentiality violations
- Detecting and recovering from misbehavior

Outline

1. Introduction

2. SPORC:
   Cloud-based group collaboration [FZFF10]

3. Frientegrity:
   Privacy & integrity for online social networks [FBFF12]

4. Conclusion

SPORC goals

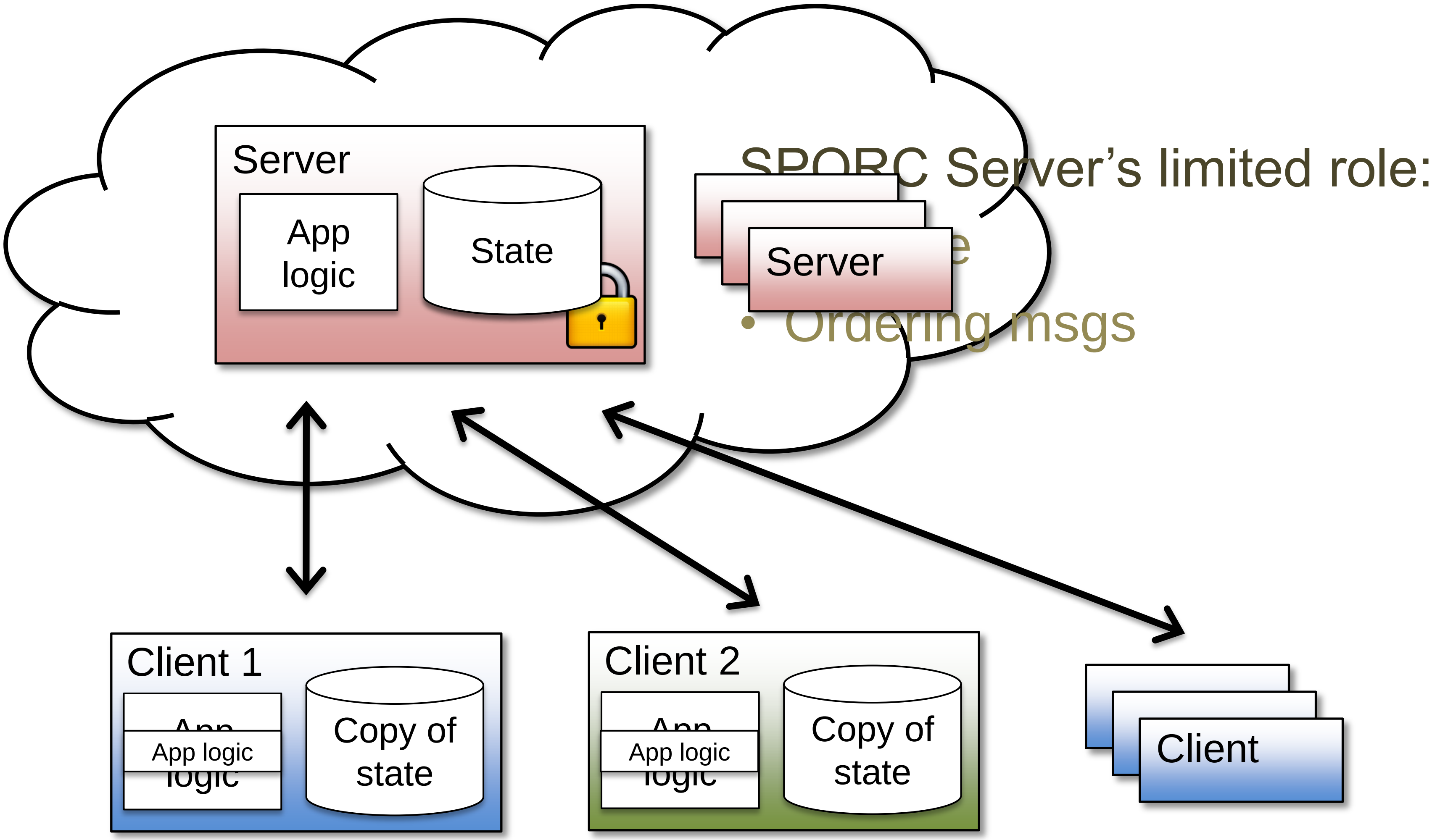
Collaborative editing of shared state

- Flexible framework
- Real-time
- Work offline

Untrusted servers

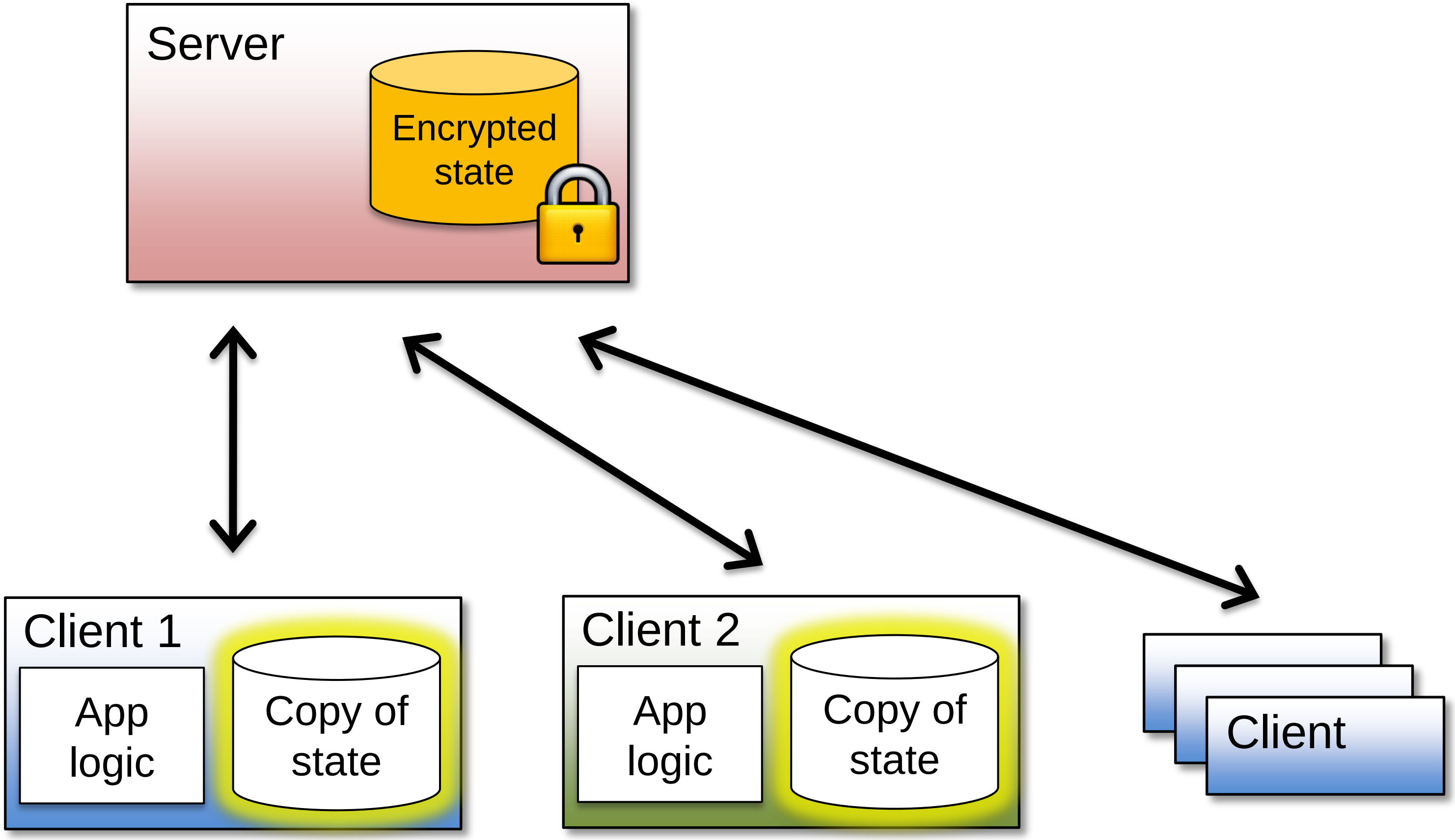
- Can't read user data
- Can't tamper with user data without risking detection
- Clients can recover from tampering

Making servers untrusted

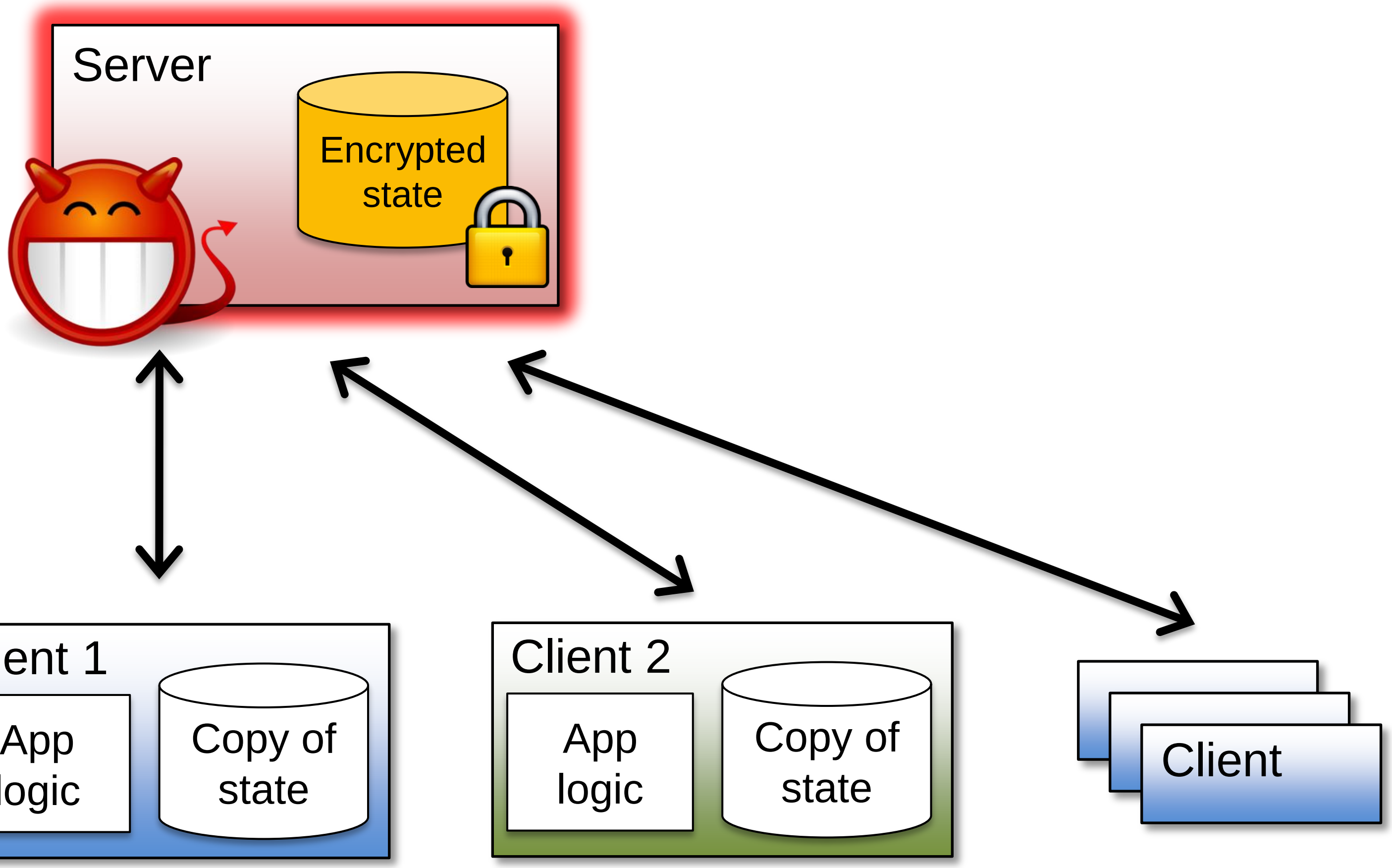


Problem #1: How do you keep clients' local copies consistent?

(esp. with offline access)



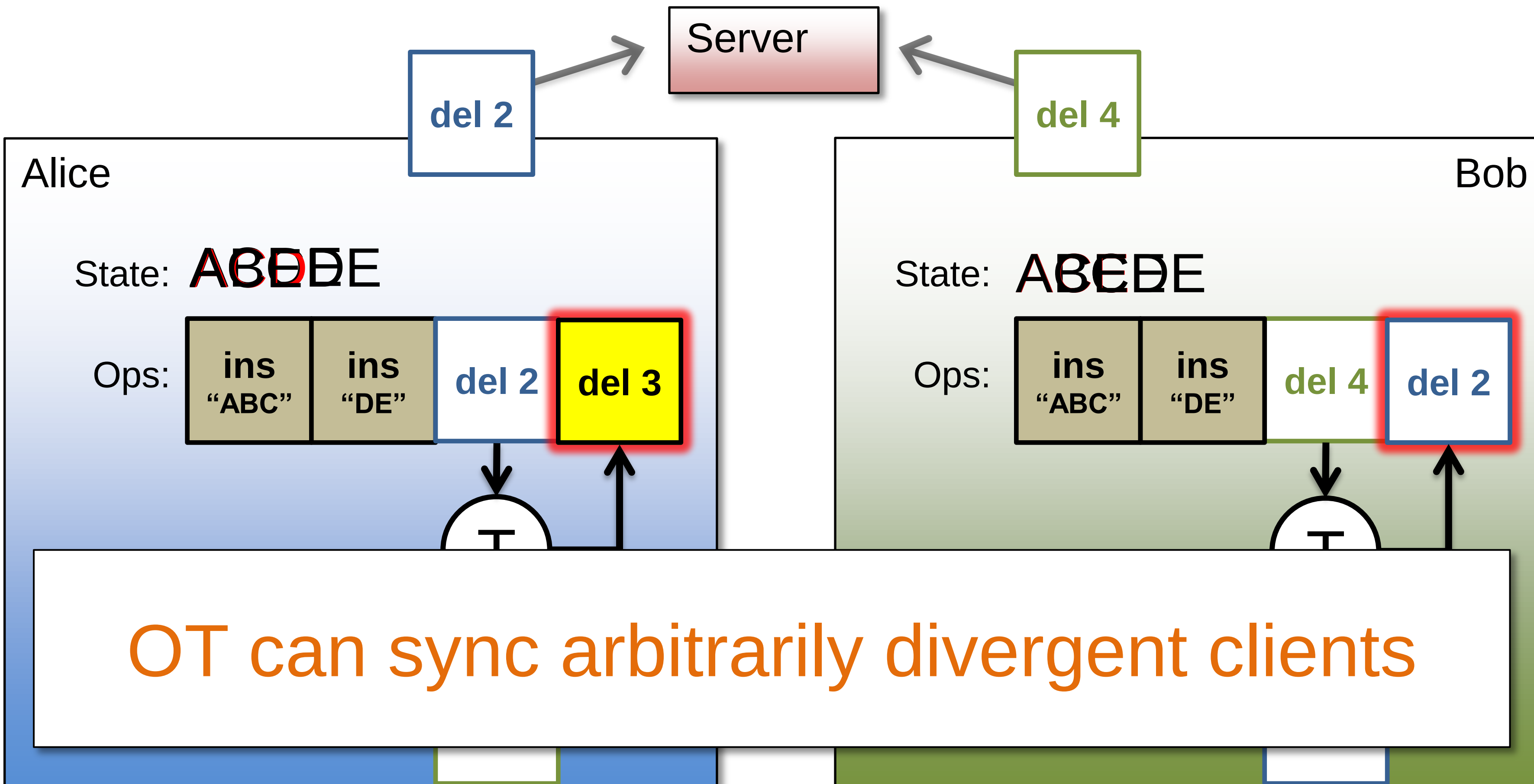
Problem #2: How do you deal with a malicious server?



Keeping clients in sync

Operational transformation (OT) [EG89]

(Used in Google Docs, EtherPad, etc.)

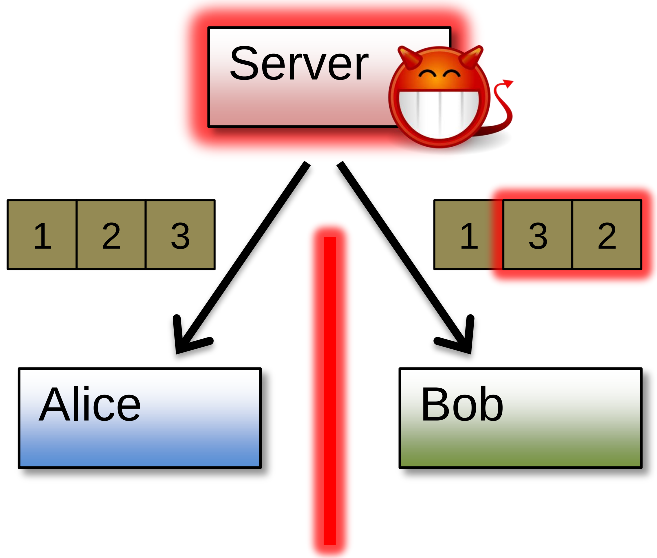


OT can sync arbitrarily divergent clients

Dealing with a malicious server

Digital signatures aren't enough

Server can equivocate

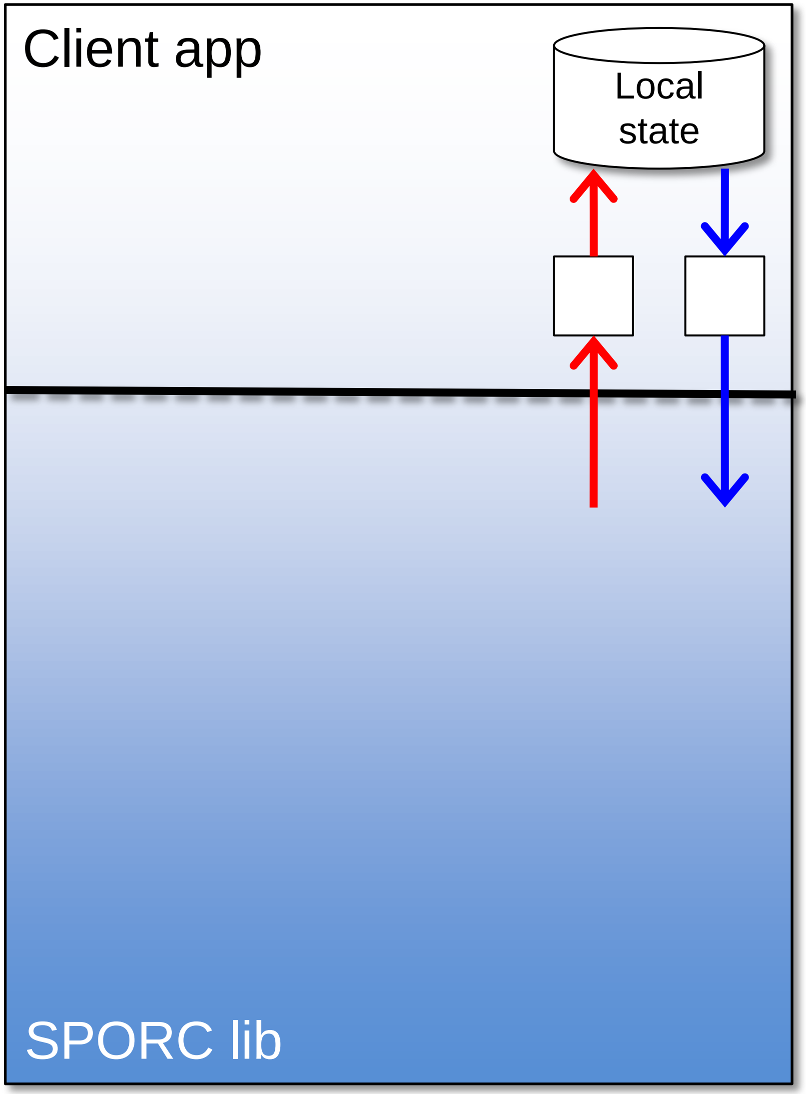


fork* consistency [LM07]

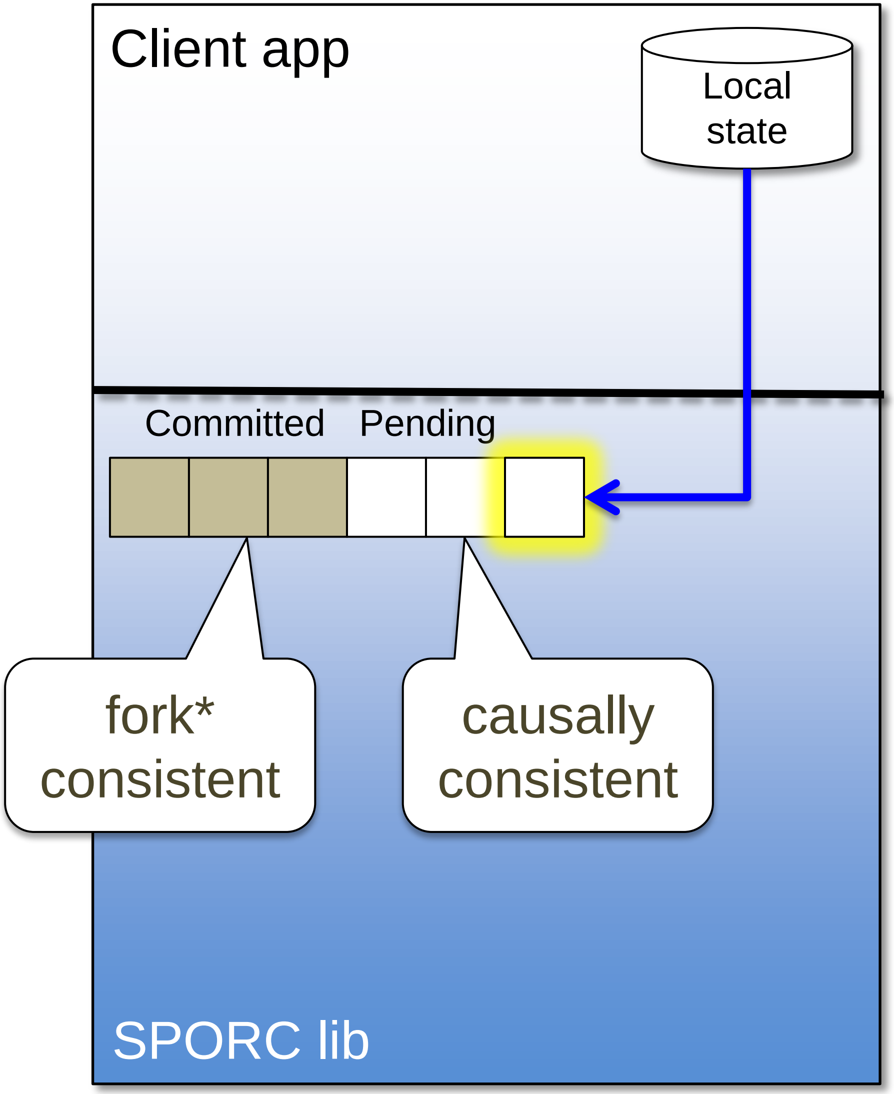
- Honest server: linearizability
- Malicious server: Alice and Bob detect equivocation after exchanging 2 messages
- Embed history hash in every message

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

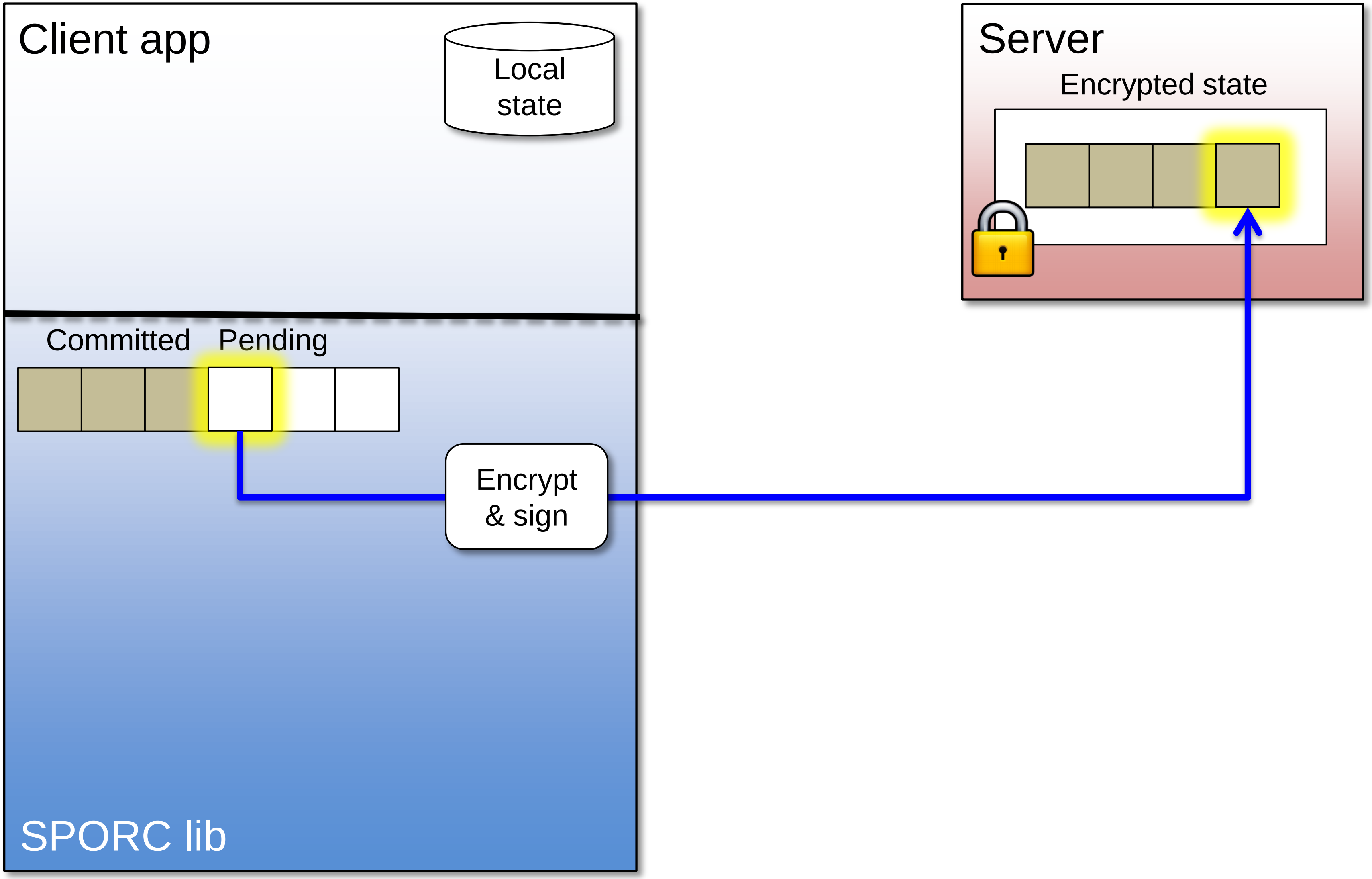
System design



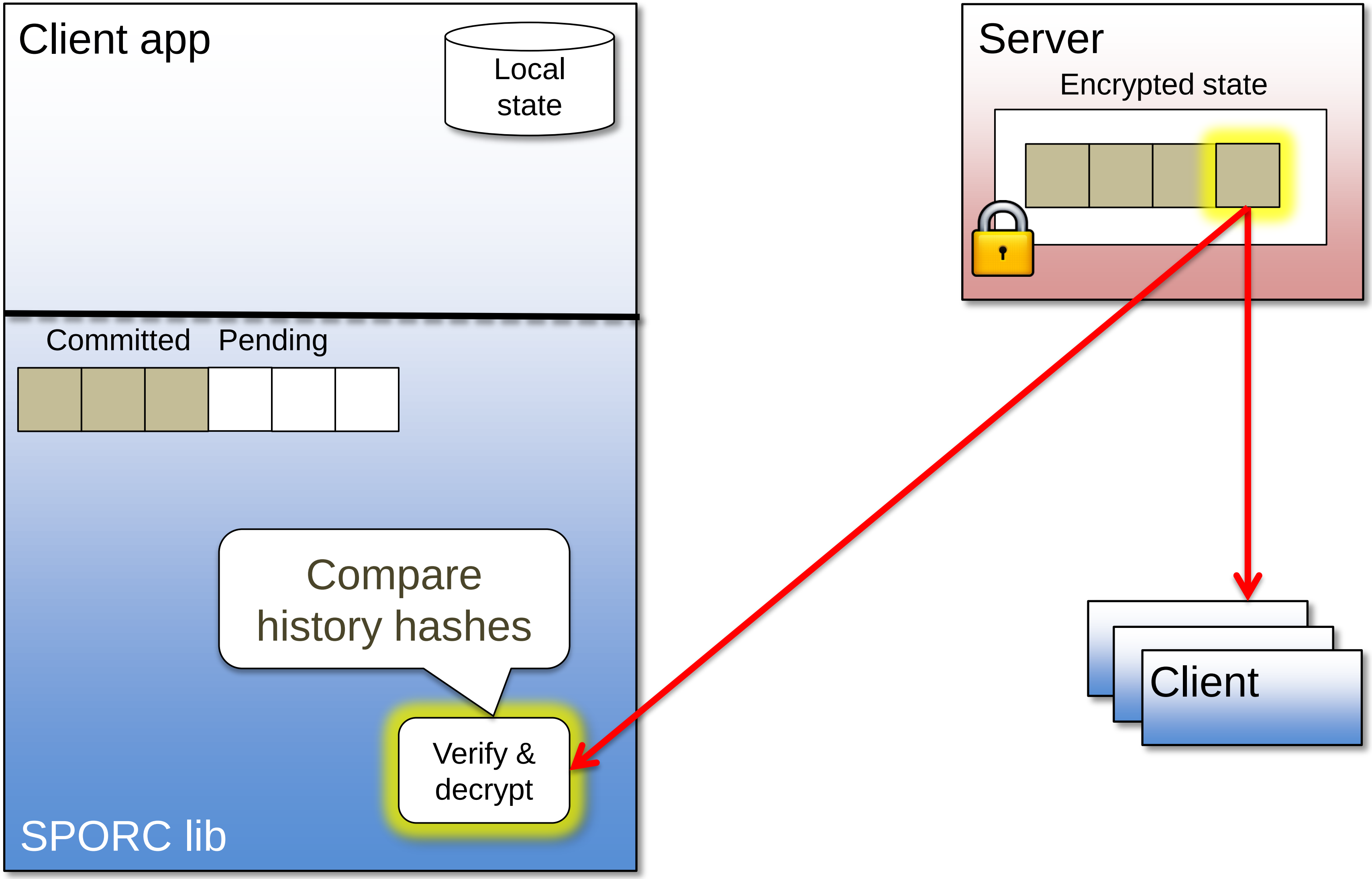
System design



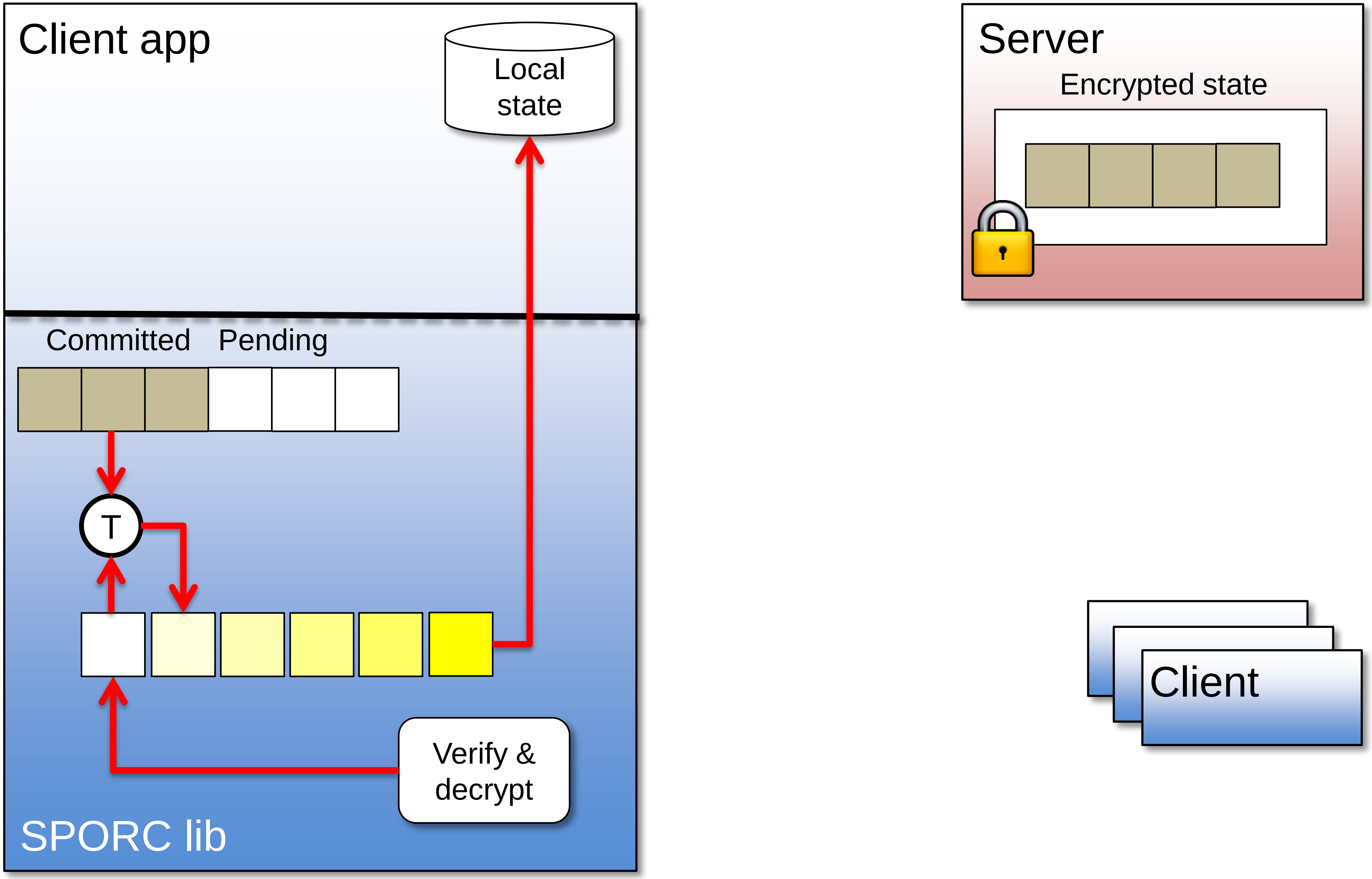
System design



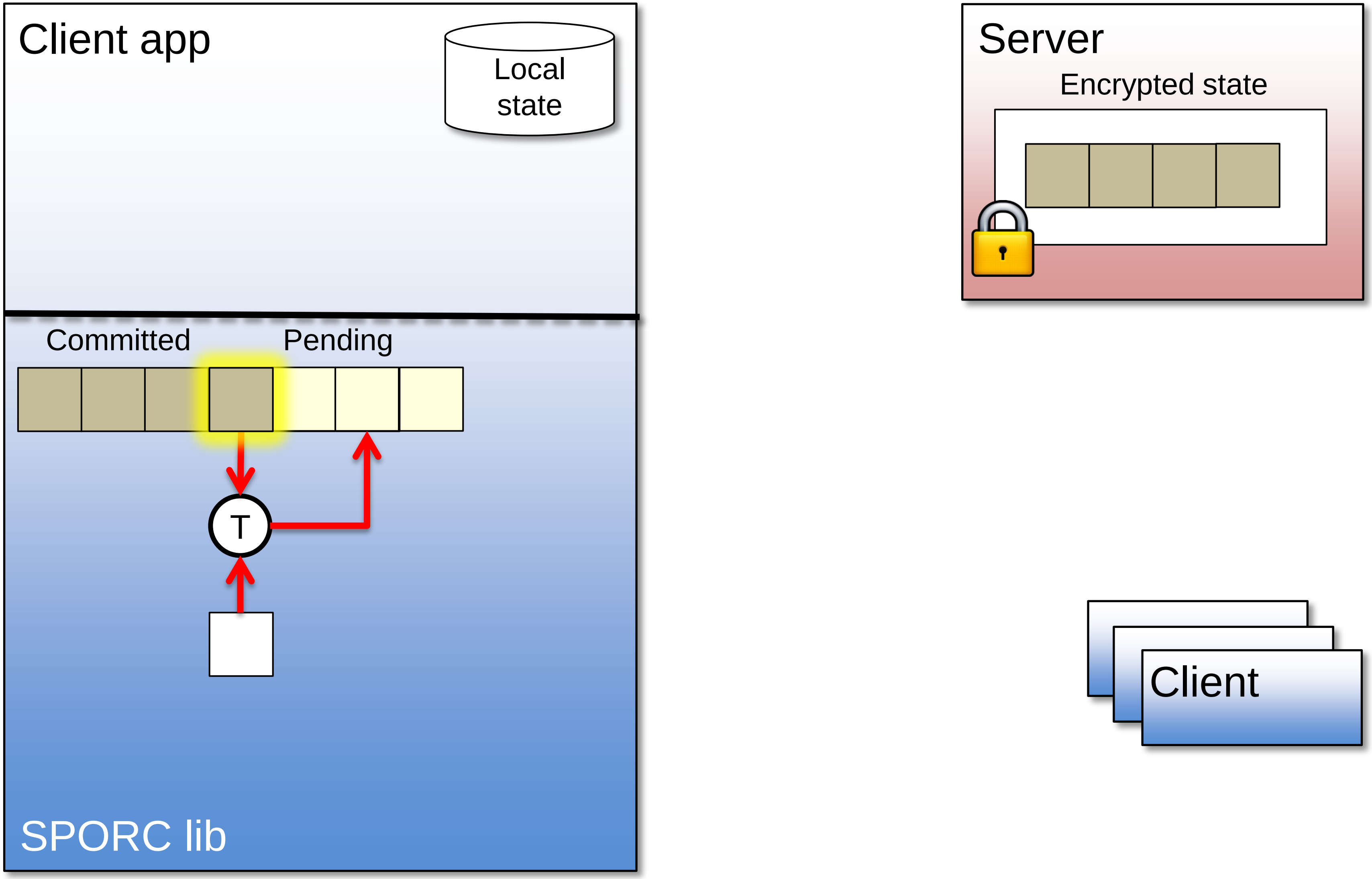
System design



System design



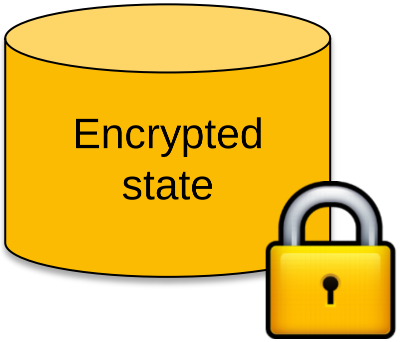
System design



Access control

Challenges

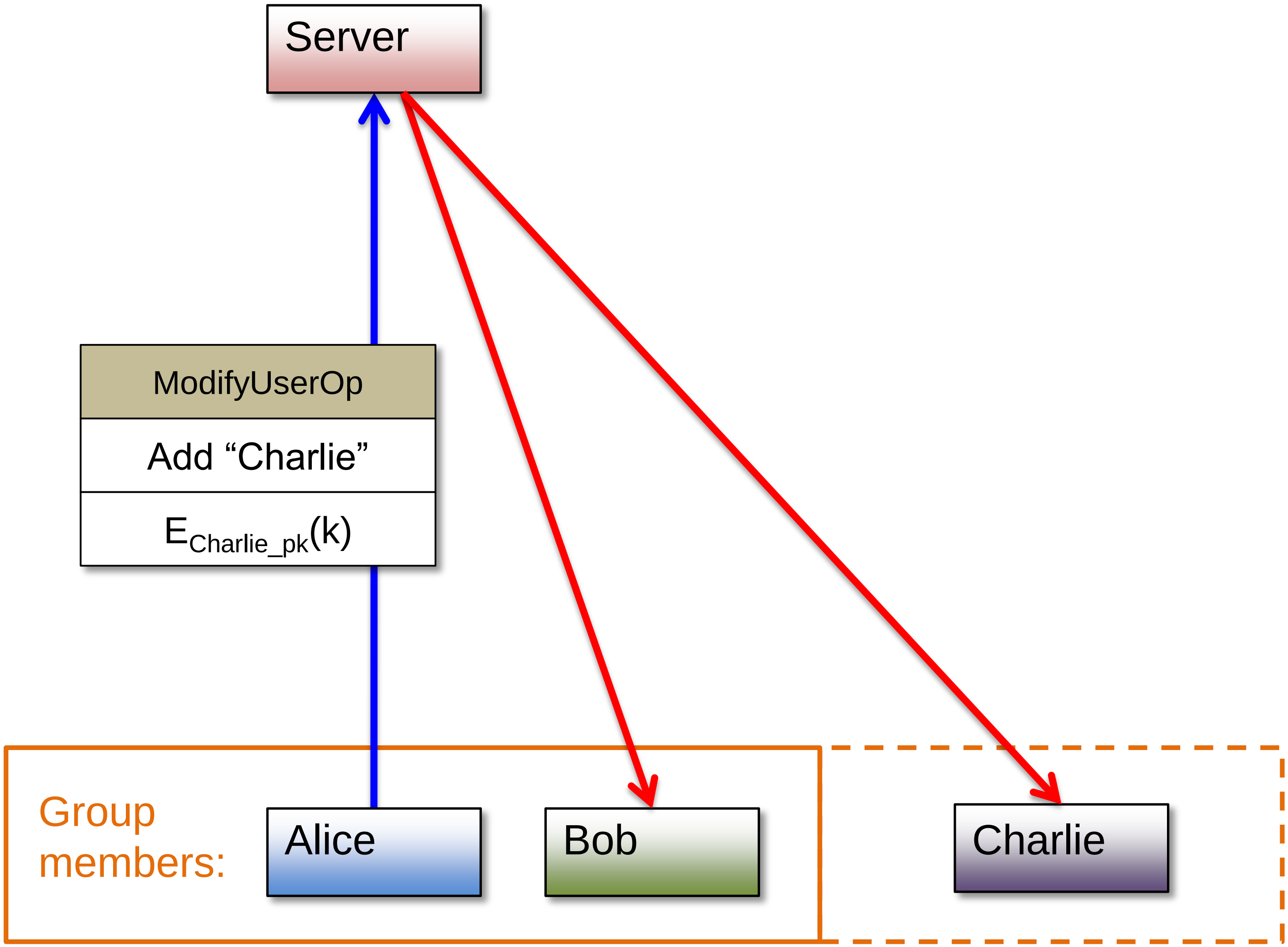
- Server can't enforce — it's untrusted!
- Preserving causality
- Concurrency makes it harder



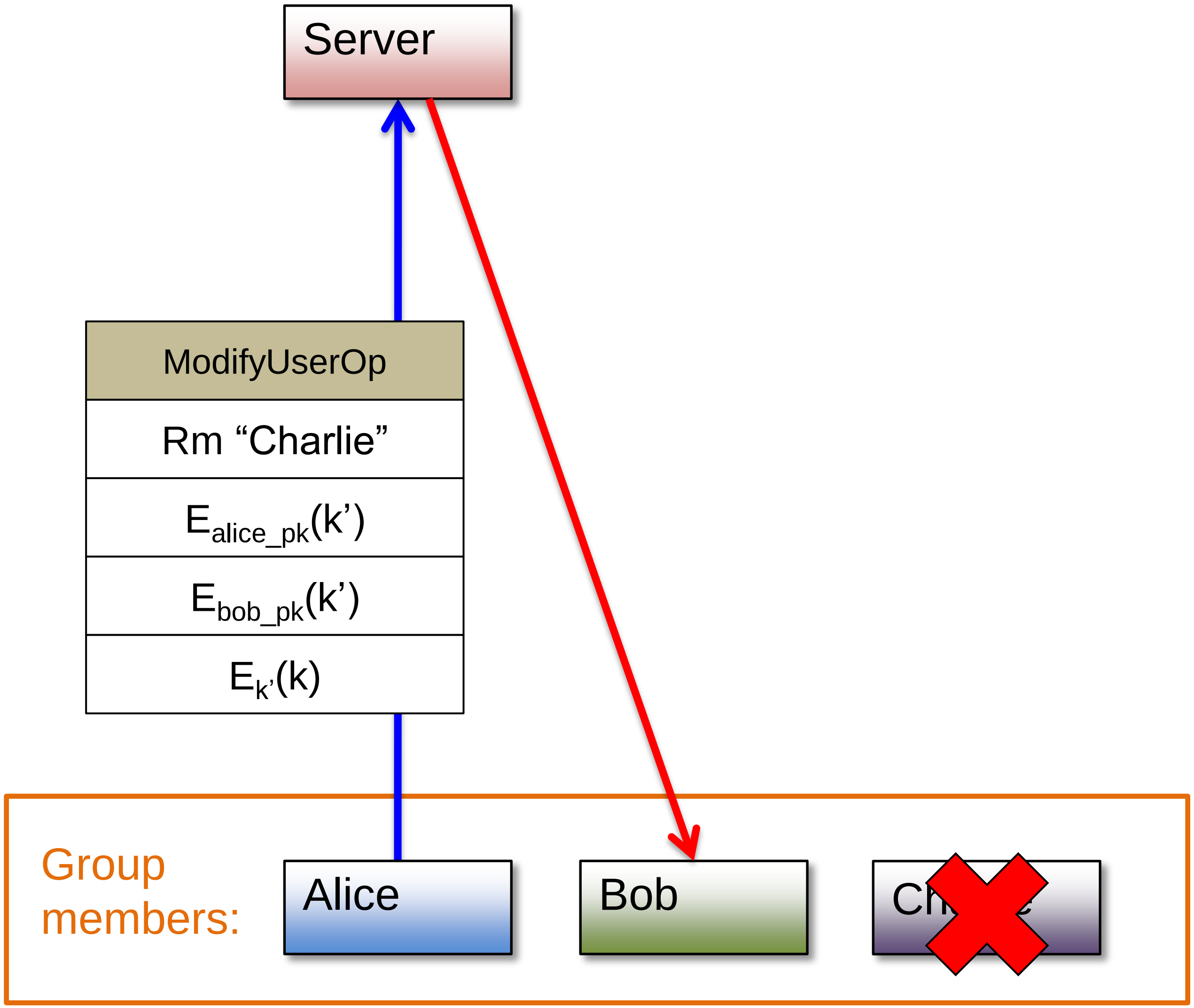
Solutions

- Ops encrypted with symmetric key shared by clients
- ACL changes are ops too
- Concurrent ACL changes handled with barriers

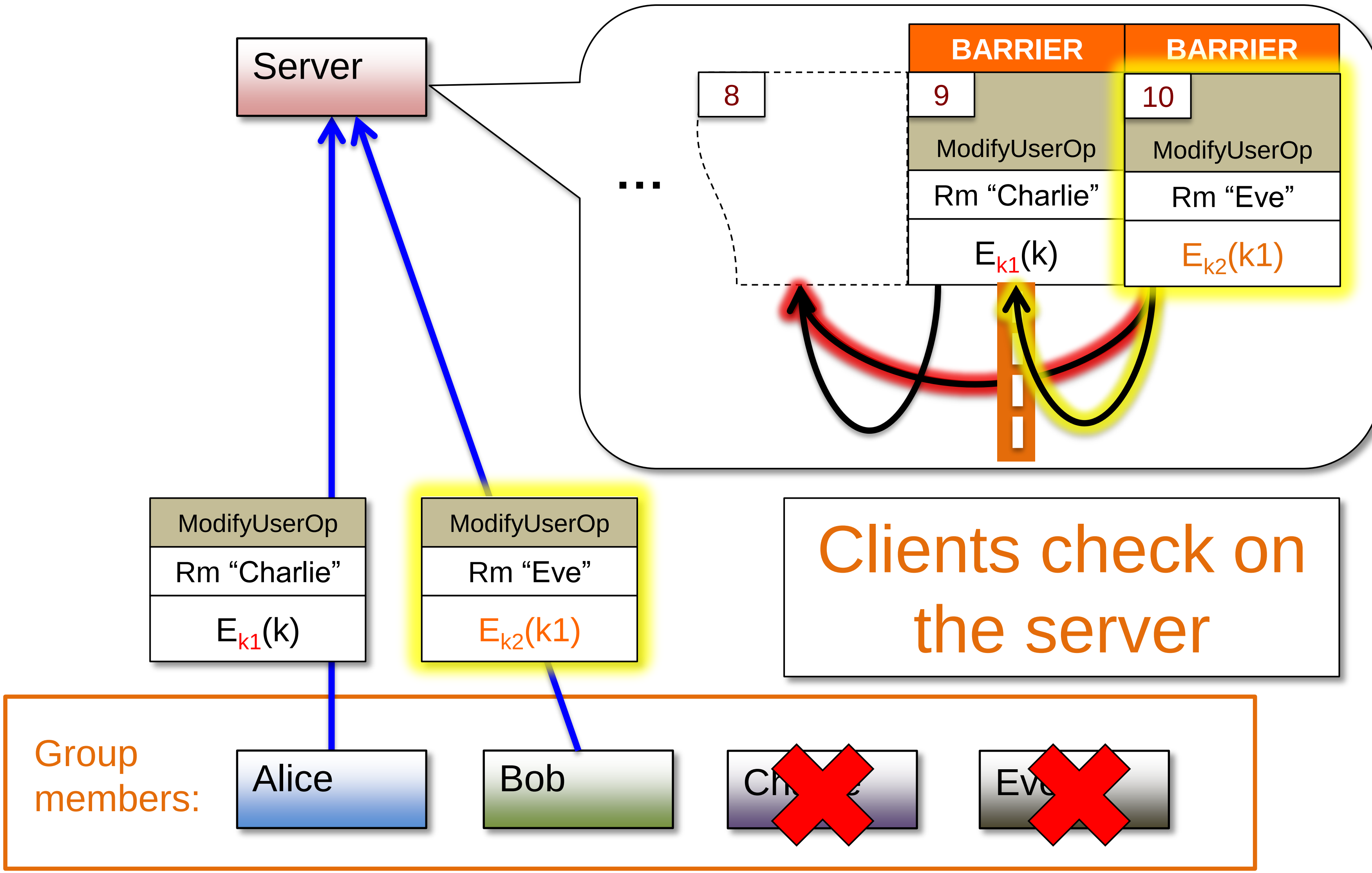
Adding a user



Removing a user

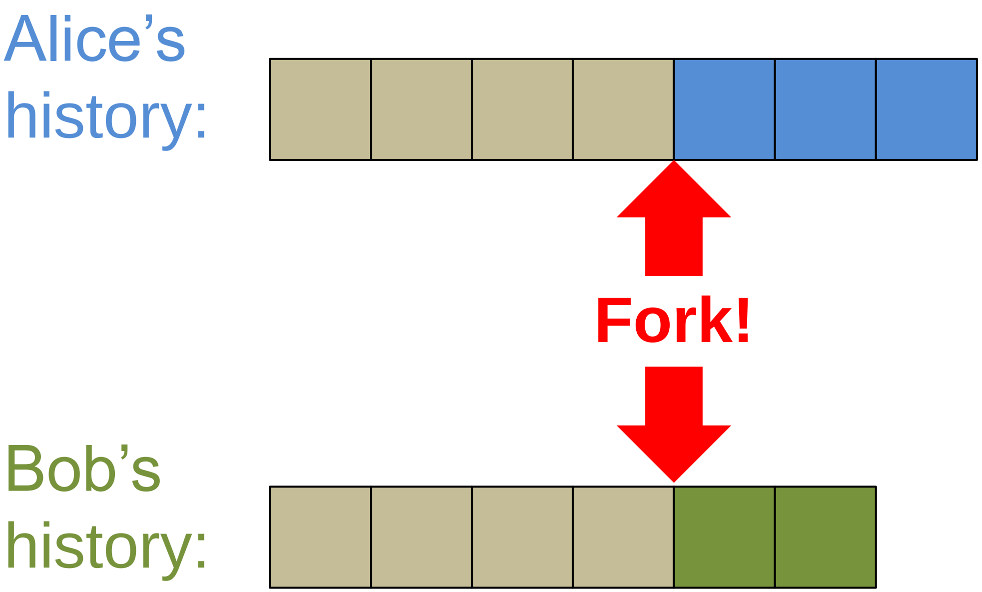


Barriers: dealing with concurrency



Clients check on the server

Recovering from a fork



Can use OT to resolve malicious forks too

Implementation

Client lib + generic server

App devs only need to define ops and provide a transformation function

Java CLI version + browser-based version (GWT)

Demo apps: key value store, browser-based collaborative text editor

Evaluation

Setup

- Tested Java CLI version
- 8-core 2.3 GHz AMD machines
  - 1 for server
  - 4 for clients (often >1 instance per machine)
- Gigabit LAN

Microbenchmarks

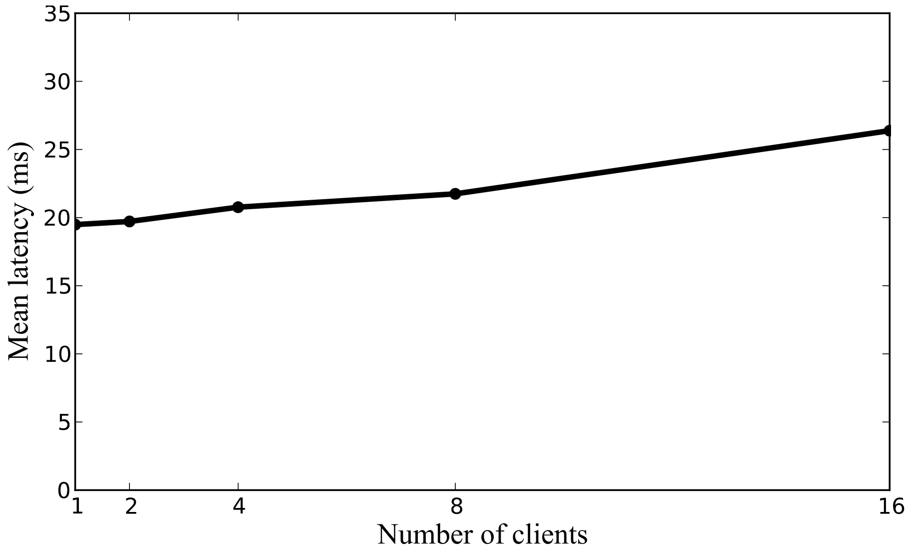
- Latency
- Server throughput
- Time-to-join (in paper)

Latency

(Text editor app)

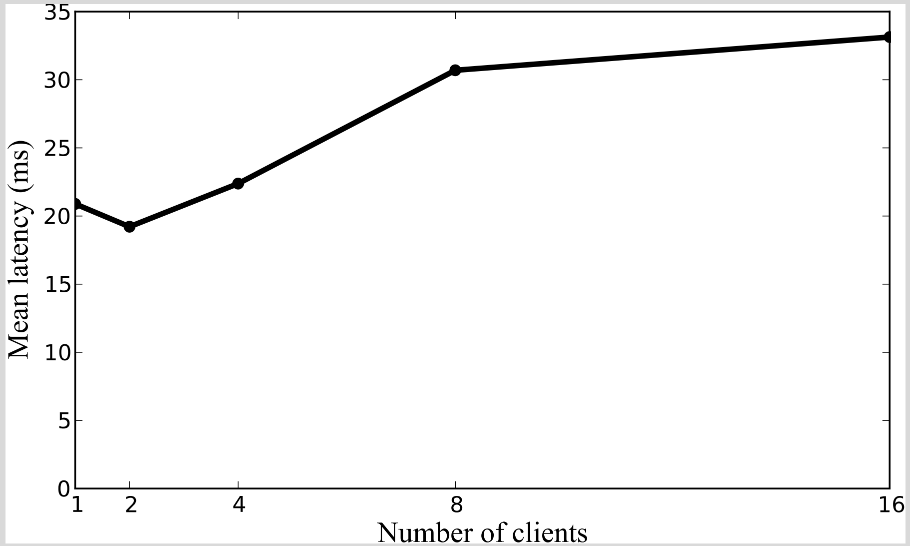
Low load

(1 client writer)

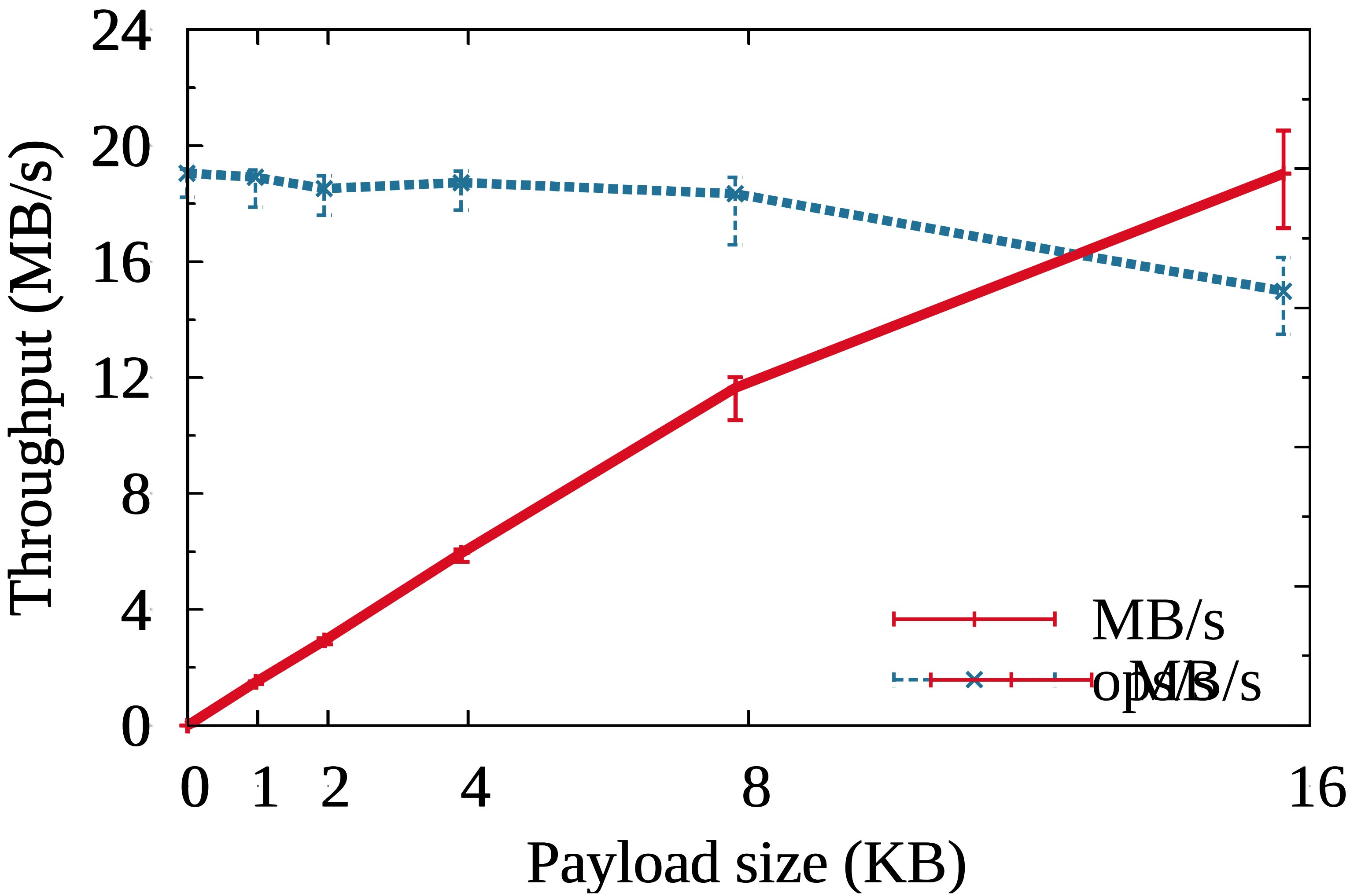


High load

(all clients are writers)



Server throughput



Summary

Practical cloud apps + untrusted servers

Dynamic access control and key distribution prevents confidentiality violations

OT + fork* consistency enables detection of and recovery from misbehavior

Outline

1. Introduction

2. SPORC:
   Cloud-based group collaboration [FZFF10]

3. Frientegrity:
   Privacy & integrity for online social networks [FBFF12]

4. Conclusion

Social network privacy & integrity

Particularly problematic:

Switching is difficult, provider tempted to repurpose data

Prior work:

1. Cryptographic (e.g. Persona, flyByNight, NOYB, Lockr, [BMP11])

Don't protect integrity

2. Decentralized (e.g. Diaspora, Safebook, eXO, PeerSON, PrPl)

Run your own server	OR	Trust a provider
(sacrifice availability, convenience, etc.)		(who you probably don't know)

Q: Why not SPORC?

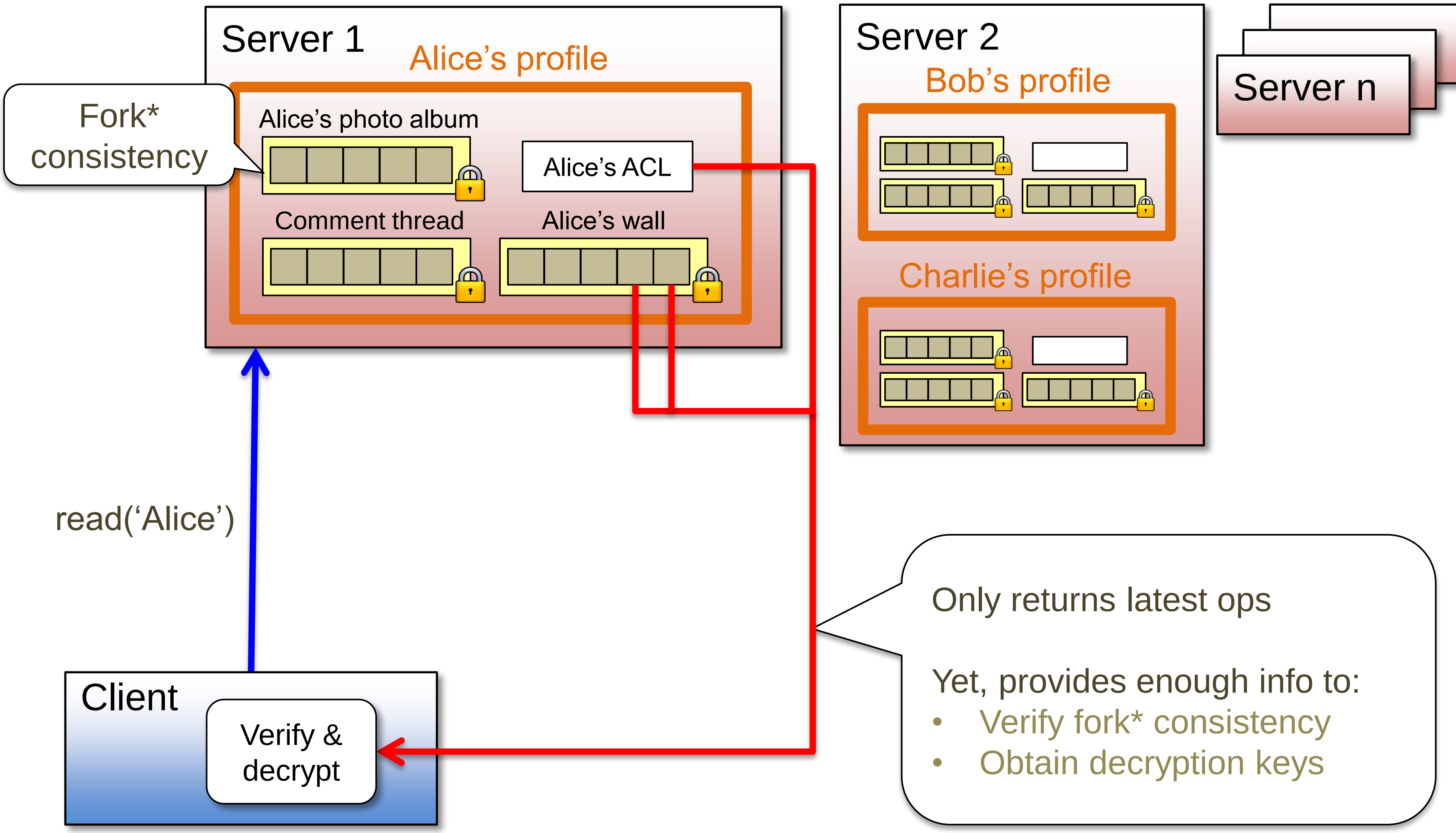
A: Scalability

SPORC provides	Social networks need
Each document is independent (Has its own ACL)	Multiple related objects (e.g. on a user's profile) (Under a single friend list)
Enforcing fork* consistency is O(n) (Downloads entire document)	Objects are large (e.g. Facebook wall) • Enforcing correctness must be fast • Only want latest changes
Few participants • ACL changes rare • Revoking access is O(n)	Many friends • "Friending" & "unfriending" common • Revoking access must be fast

Frientegrity

Social networks need	Frientegrity provides
Multiple related objects (e.g. on a user's profile) (Under a single friend list)	Multiple related objects • Spread across servers • Share an ACL
Objects are large (e.g. Facebook wall) • Enforcing correctness must be fast • Only want latest changes	Clients enforce fork* consistency collaboratively Each client only downloads & verifies a small part of an object
Many friends • "Friending" & "unfriending" common • Revoking access must be fast	ACL operations O(log n)

Frientegrity overview



Enforcing fork* consistency in SPORC

op_0	op_1	op_2	op_3	op_4	op_5	op_6	op_7	op_8	op_9

$h_n = H(h_{n-1} \,||\, op_n)$

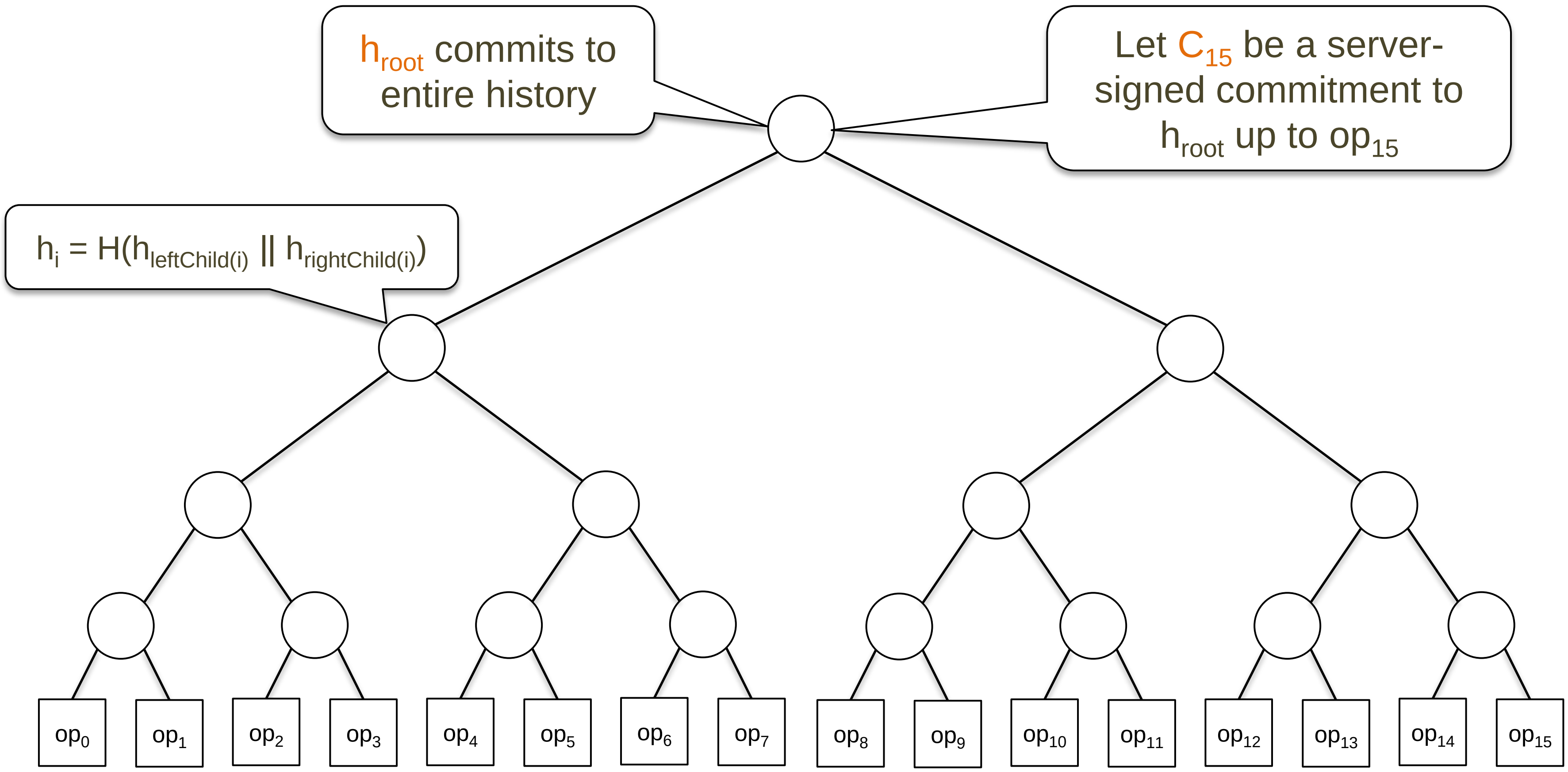
SPORC's hash chains are O(n)

(Also, must download entire history)

Prior systems were linear in history size or number of clients

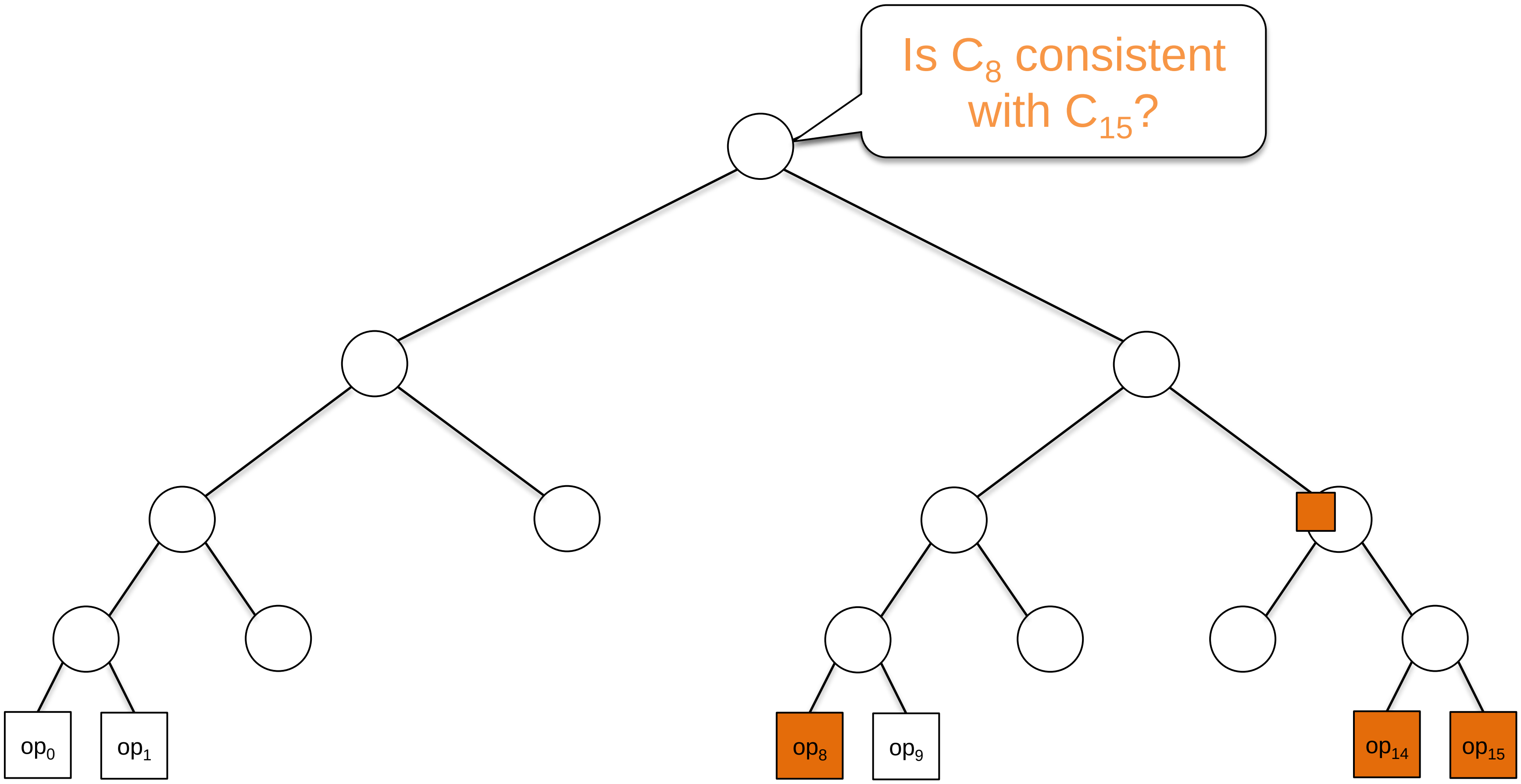
(e.g. SUNDR, Depot)

Objects in Frientegrity

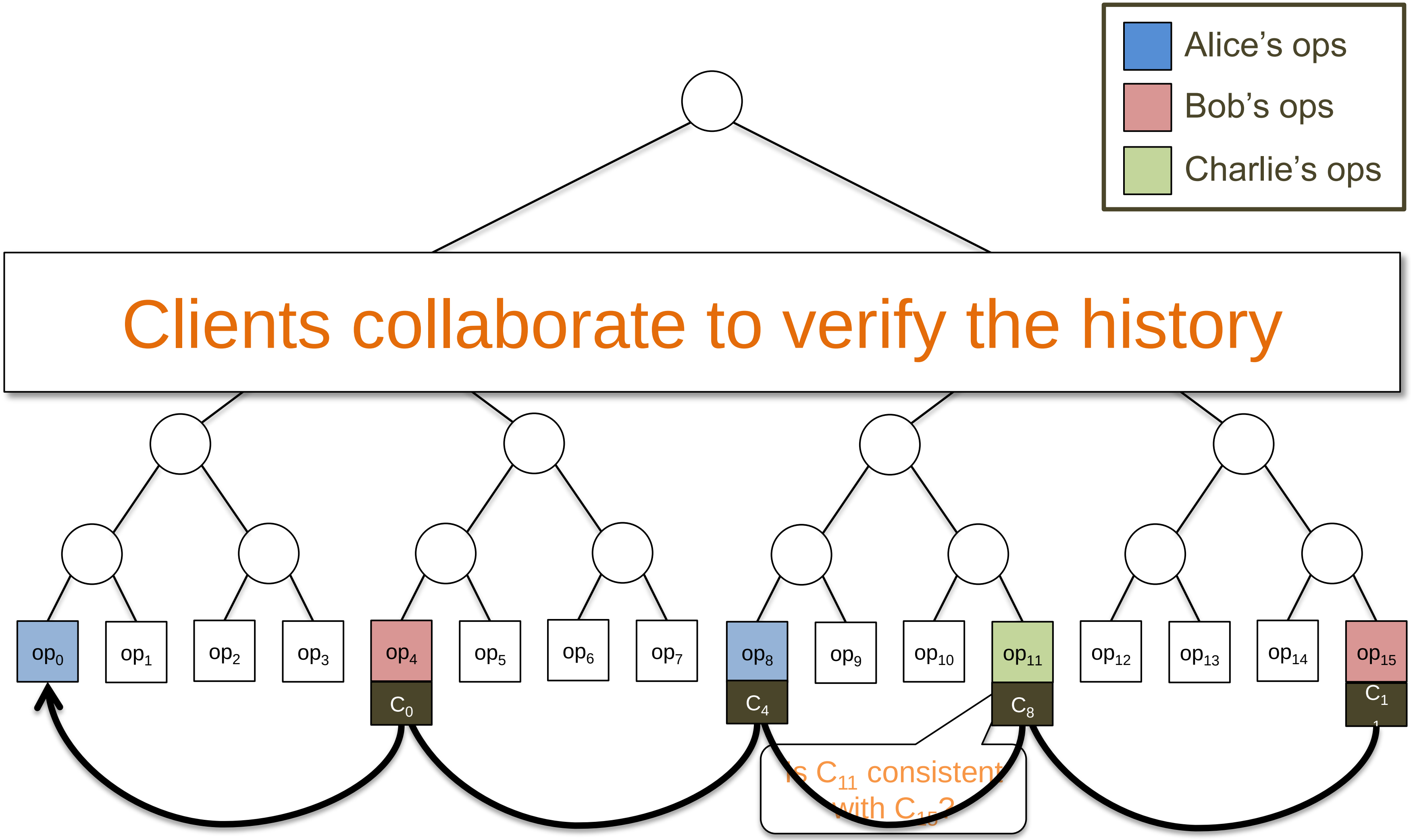


History tree [CW09]

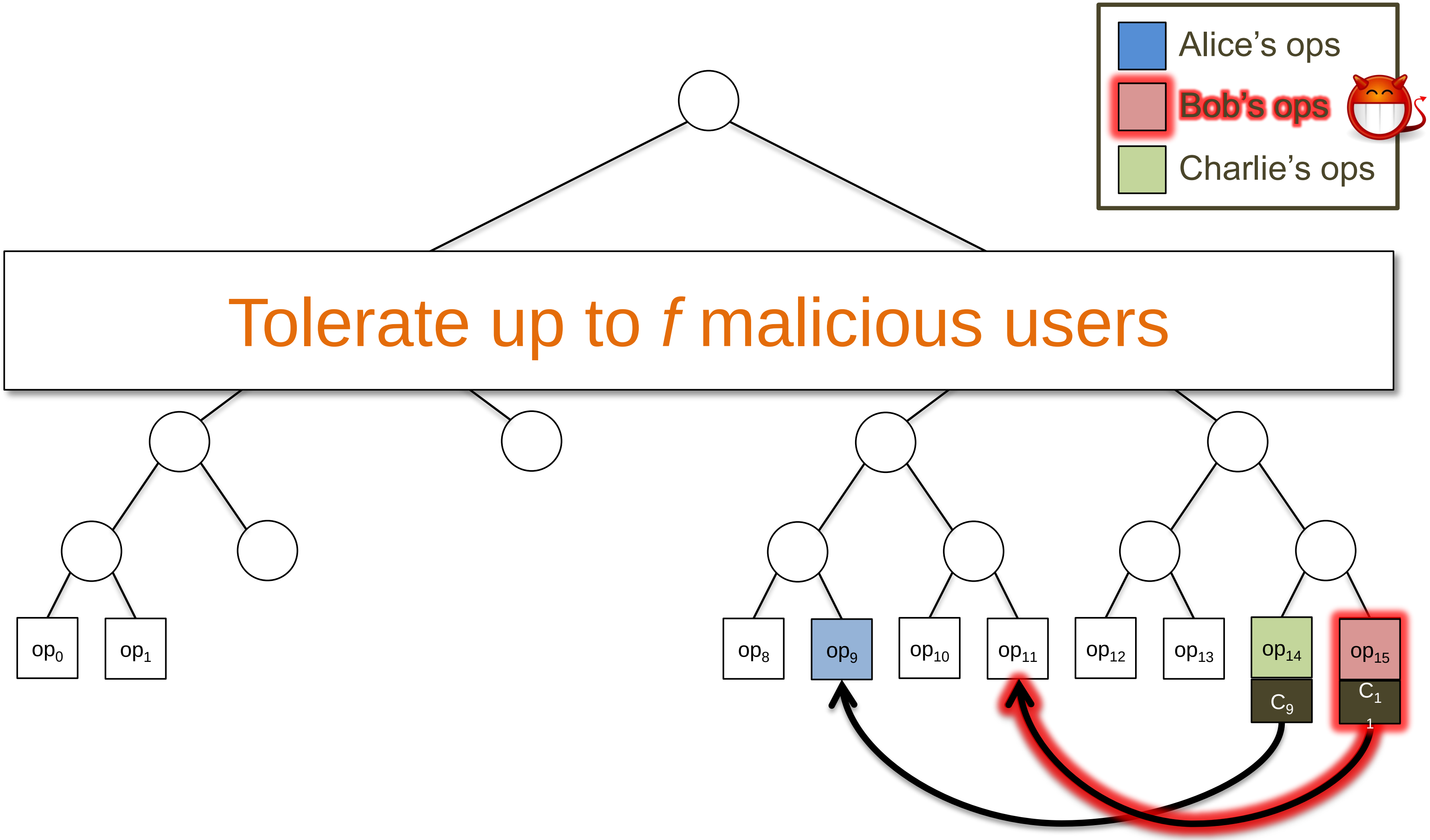
Objects (cont.)



Verifying an object



Tolerating malicious users

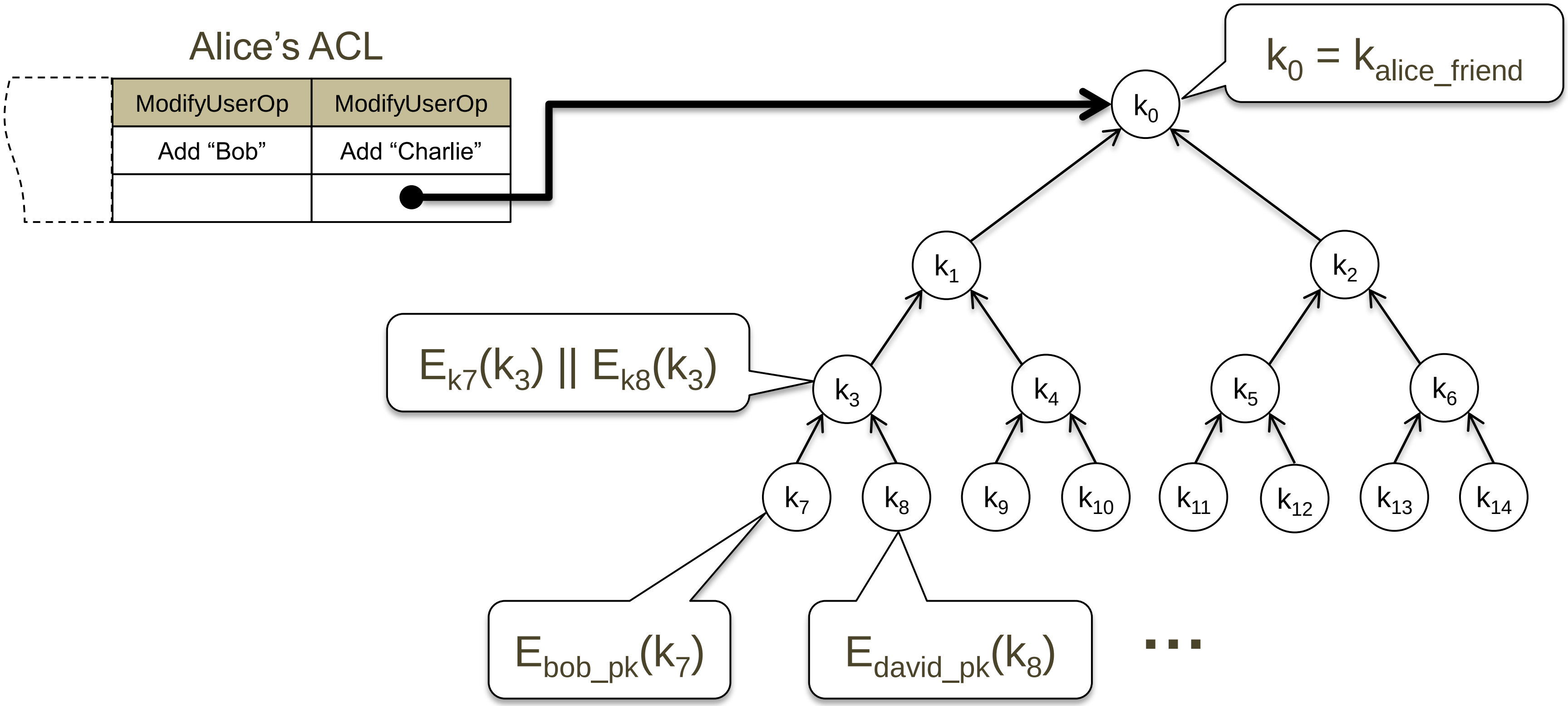


Scalable access control

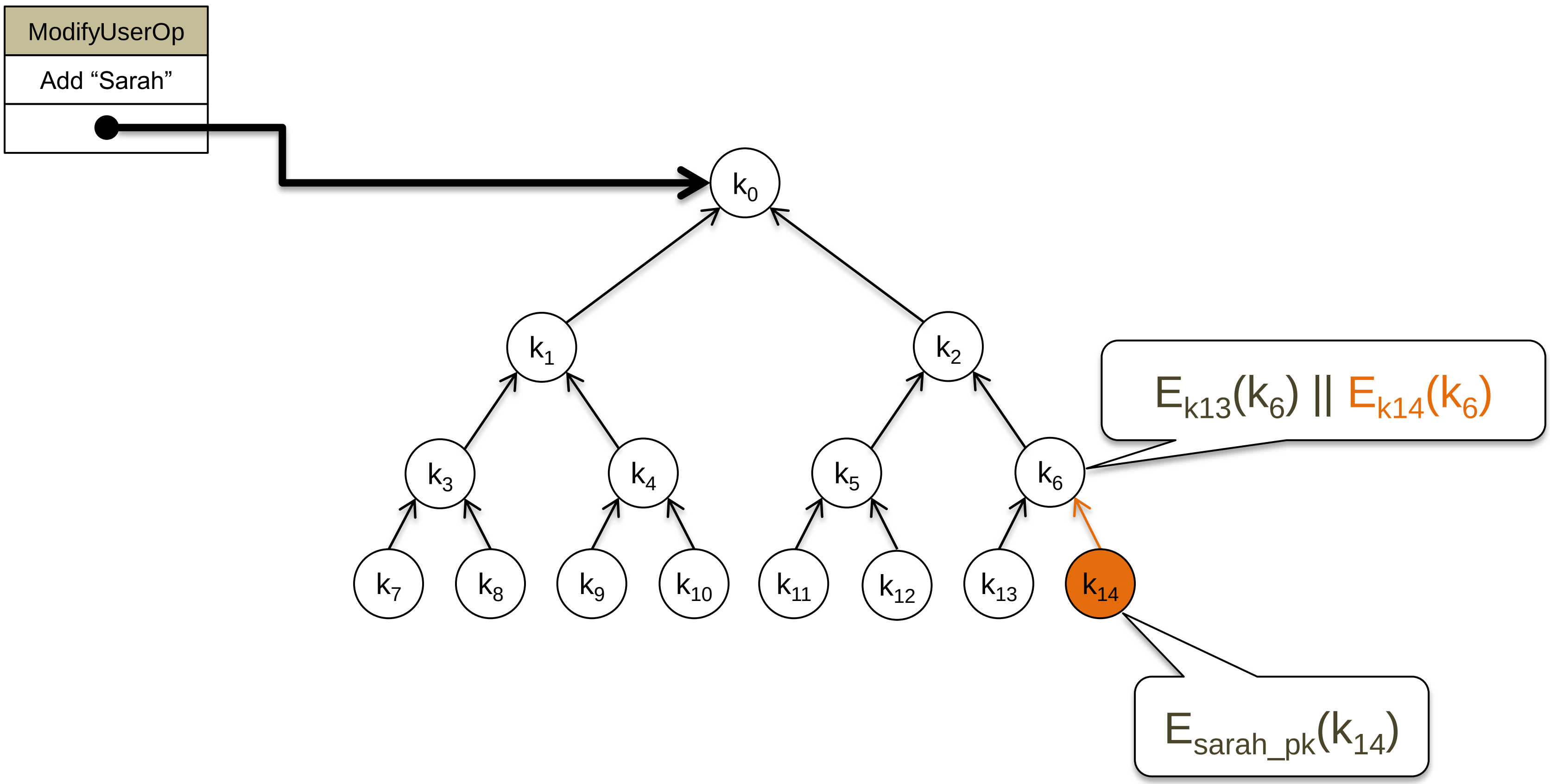
ModifyUserOp
Rm "Charlie"
$E_{alice\_pk}(k')$
$E_{bob\_pk}(k')$
⋮
$E_{k'}(k)$

SPORC membership ops are expensive

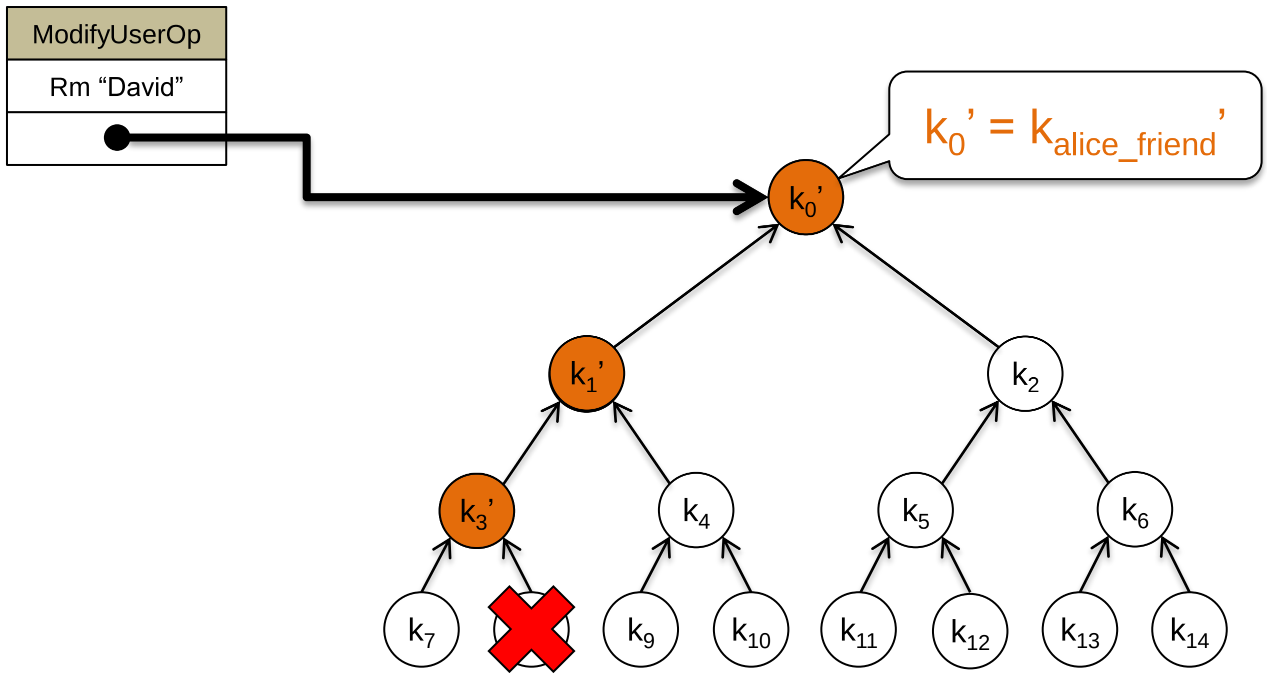
Instead, use a key graph [WGL98]



Adding a friend



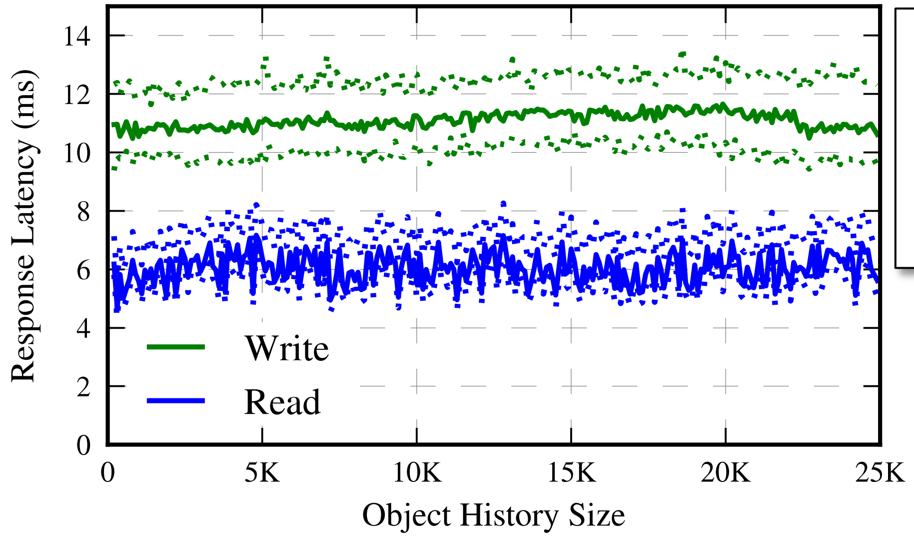
Removing a friend



Read & write latency

Frientegrity

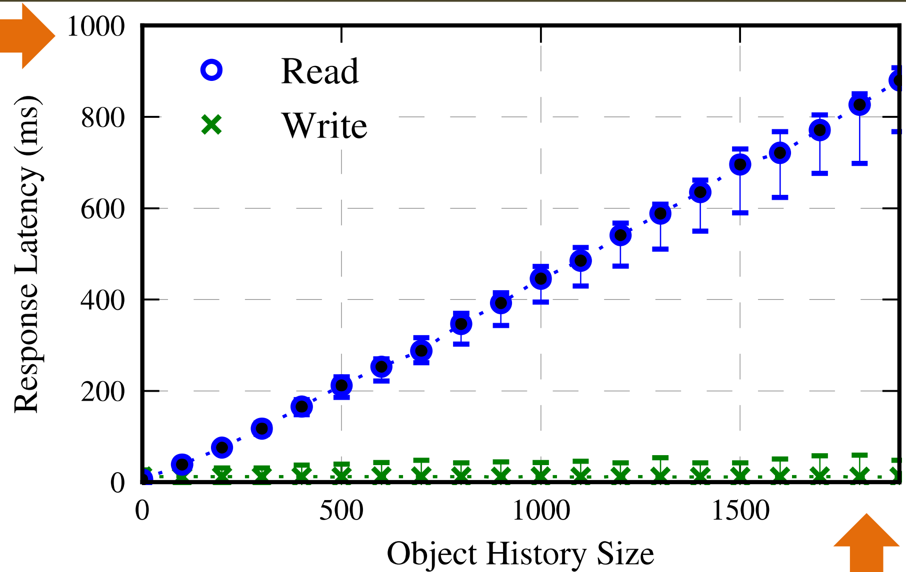
(collaborative verification)



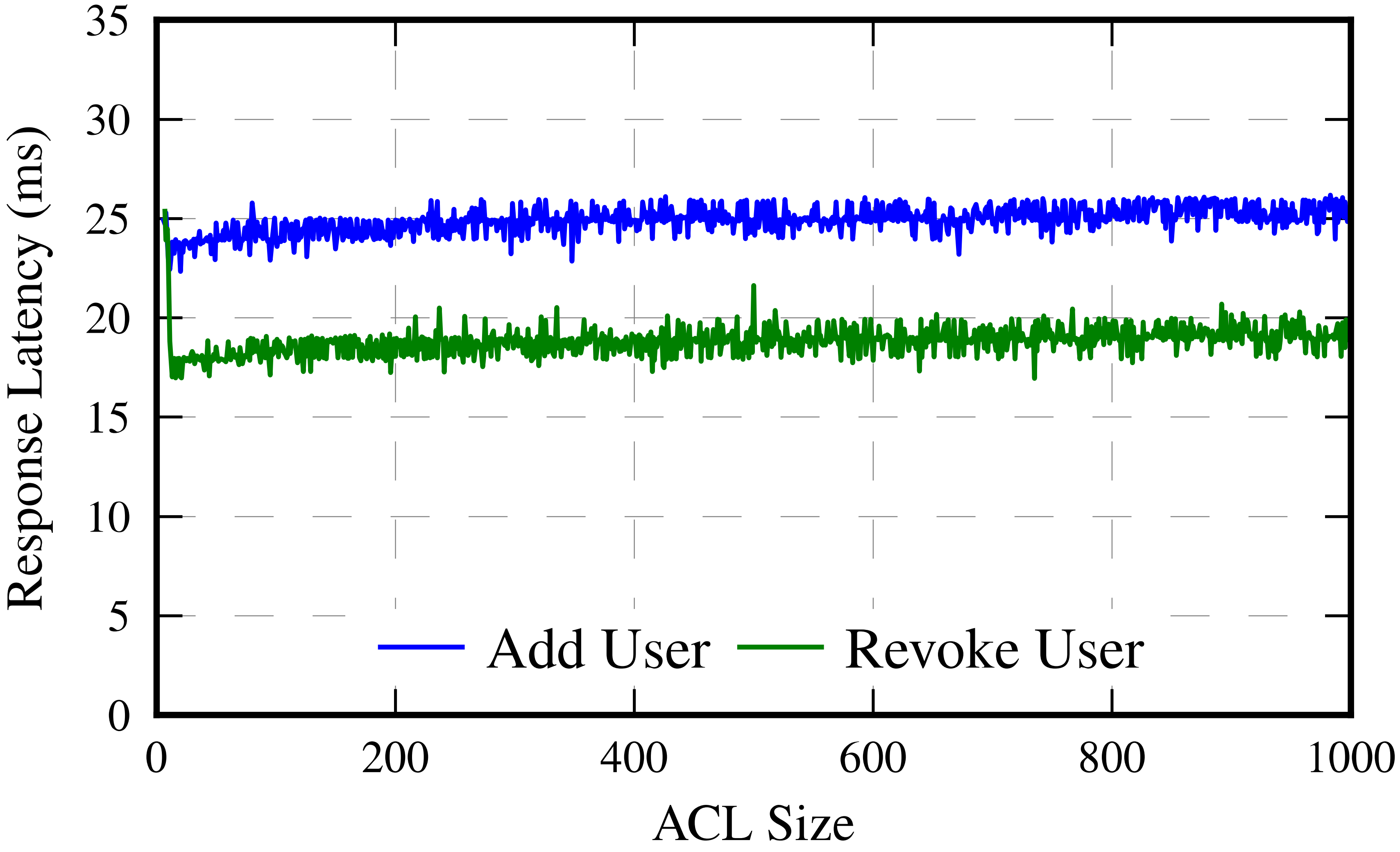
Constant cost of signatures dominates

SPORC

(hash chain)

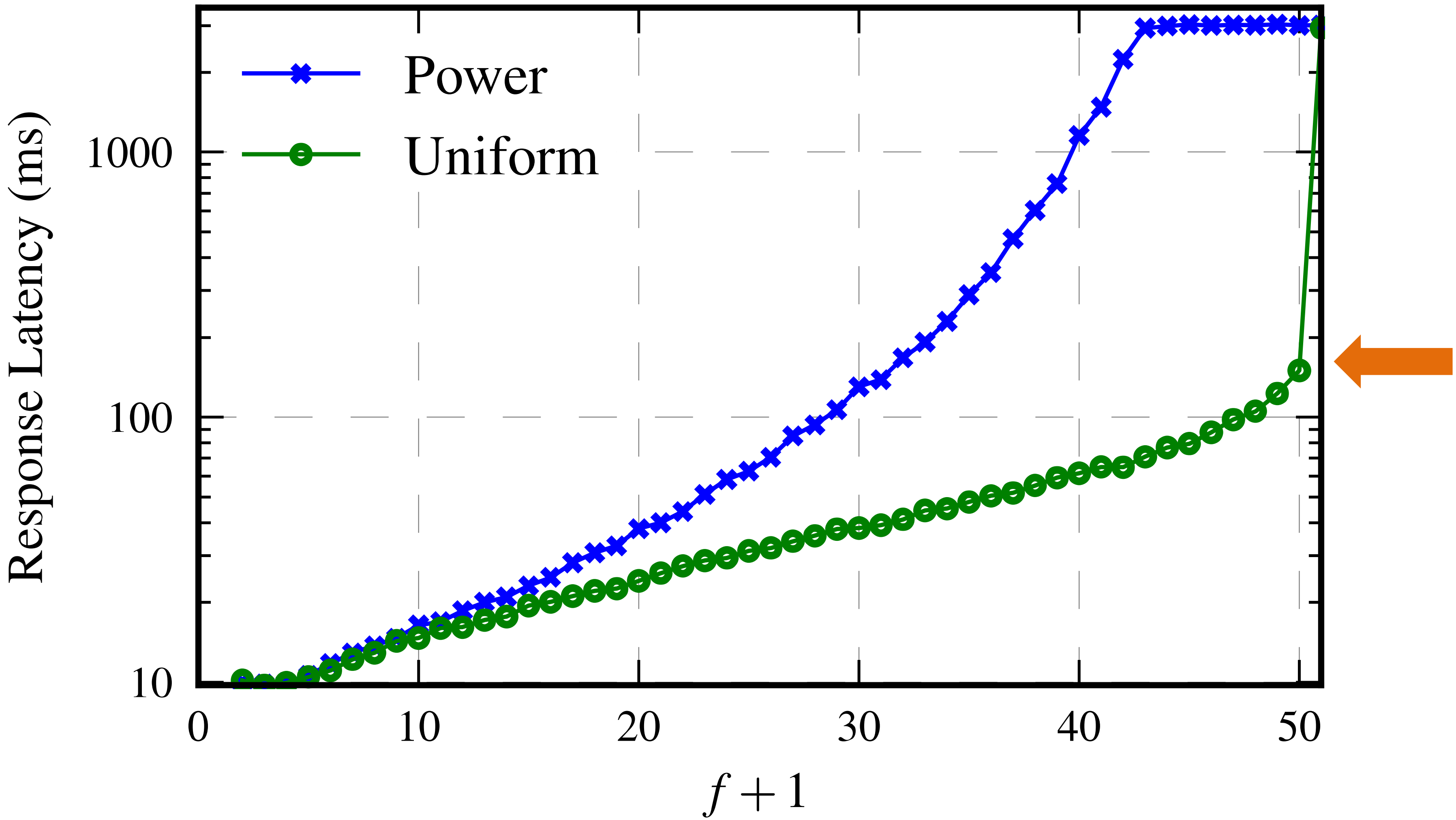


Latency of ACL changes



Effect of increasing *f*

- 50 writers
- 5000 operations



Summary

Online social networking + untrusted provider

Clients collaborate to defend against equivocation
(i.e. to enforce fork* consistency)

Tolerates up to f malicious users
(SPORC assumed trusted clients)

Scalable access control: key distribution & revocation are O(log n)

Outline

1. Introduction
2. SPORC:
   Cloud-based group collaboration [FZFF10]
3. Frientegrity:
   Privacy & integrity for online social networks [FBFF12]
4. Conclusion

Conclusion

Practical apps + untrusted provider are possible

- Assume actively malicious (Byzantine faulty) provider
- Privacy & integrity guaranteed by users' keys

Contributions:

- Frameworks for group collaboration & online social networking
- Detect and recover from equivocation
- Dynamic access control & key distribution that supports concurrency
- Protocols that scale to needs of real-time collaboration & large online social networks

Thank you

Questions?

felten@cs.princeton.edu

References

[FZFF10] A. J. Feldman, W. P. Zeller, M. J. Freedman, and E. W. Felten. “SPORC: Group Collaboration using Untrusted Cloud Resources.” OSDI 2010.

[FBFF12] A. J. Feldman, A. Blankstein, M. J. Freedman, and E. W. Felten. “Social Networking with Frientegrity: Privacy and Integrity with an Untrusted Provider” USENIX Security 2012. *To appear.*