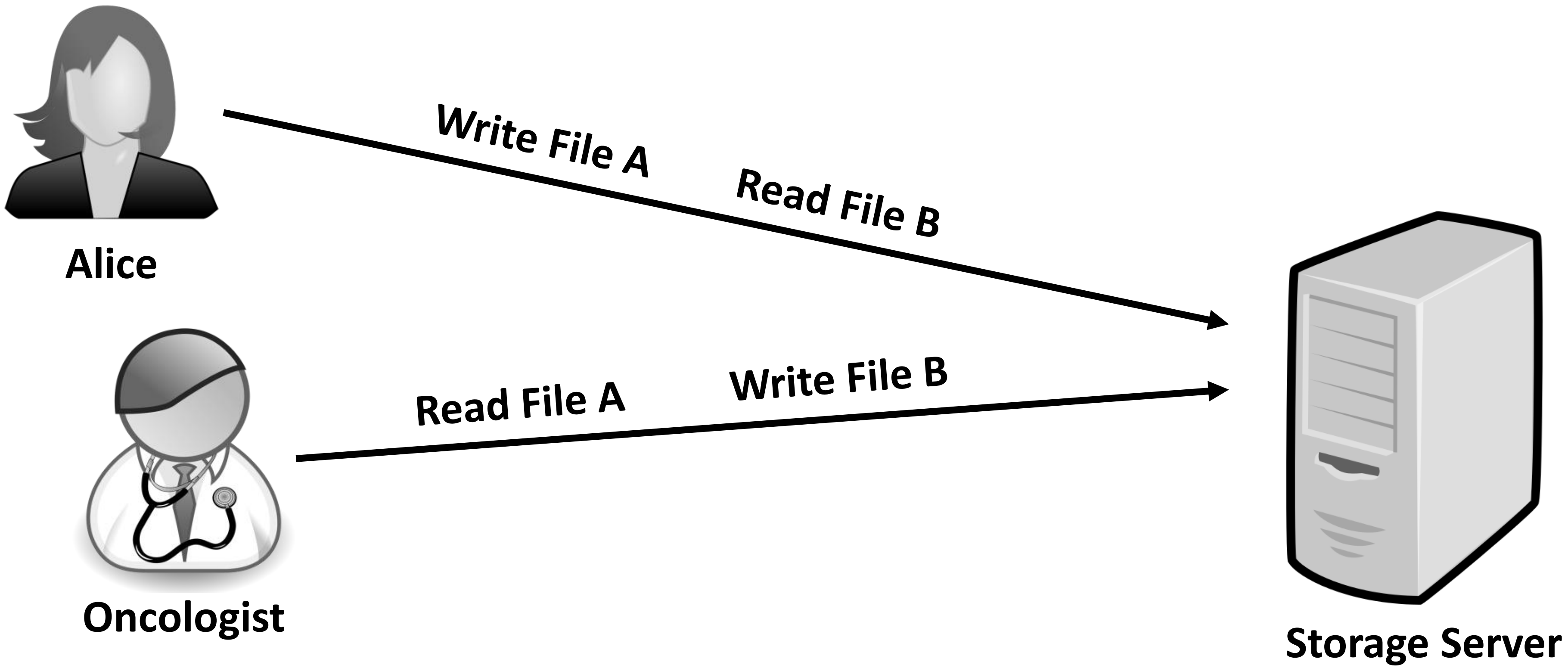
Ghostor: Toward a Secure Data-Sharing System from Decentralized Trust

*Yuncong Hu, ***Sam Kumar**, and Raluca Ada Popa

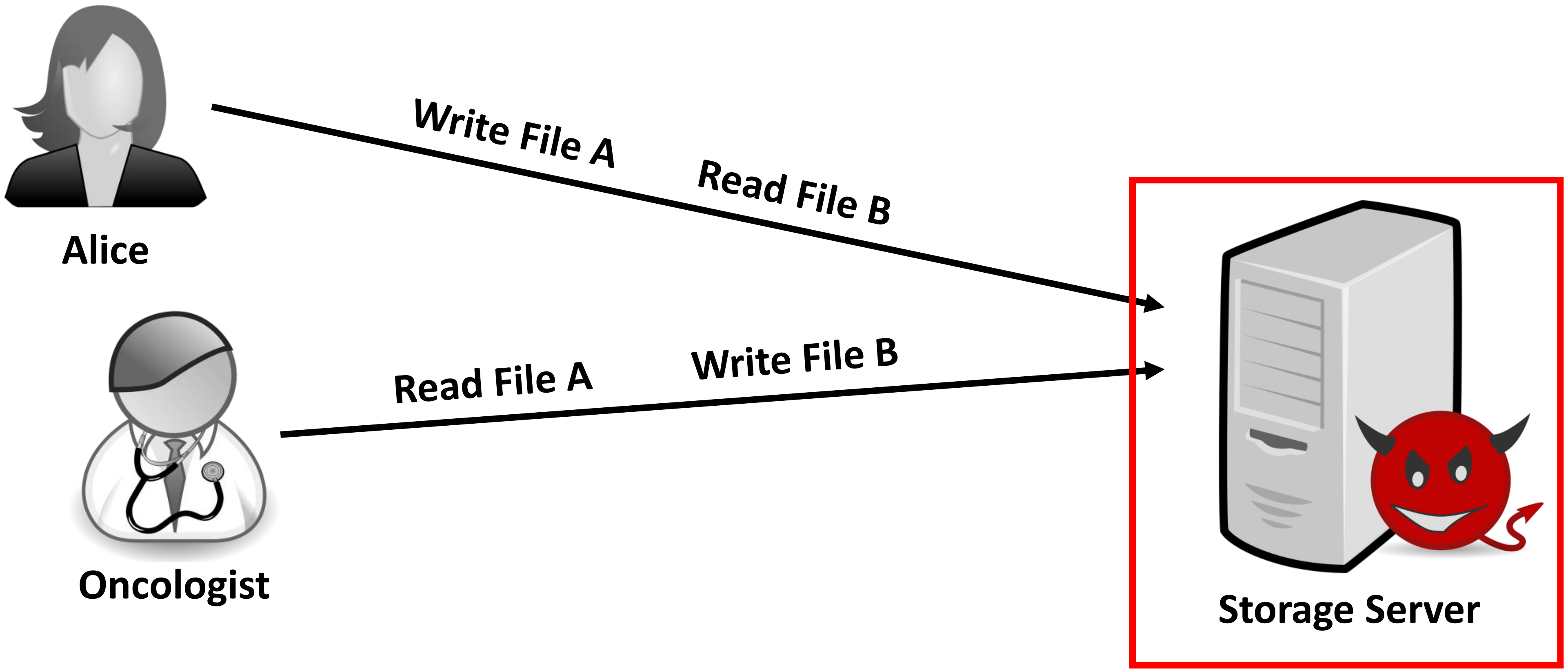
University of California, Berkeley

*Co-primary authors

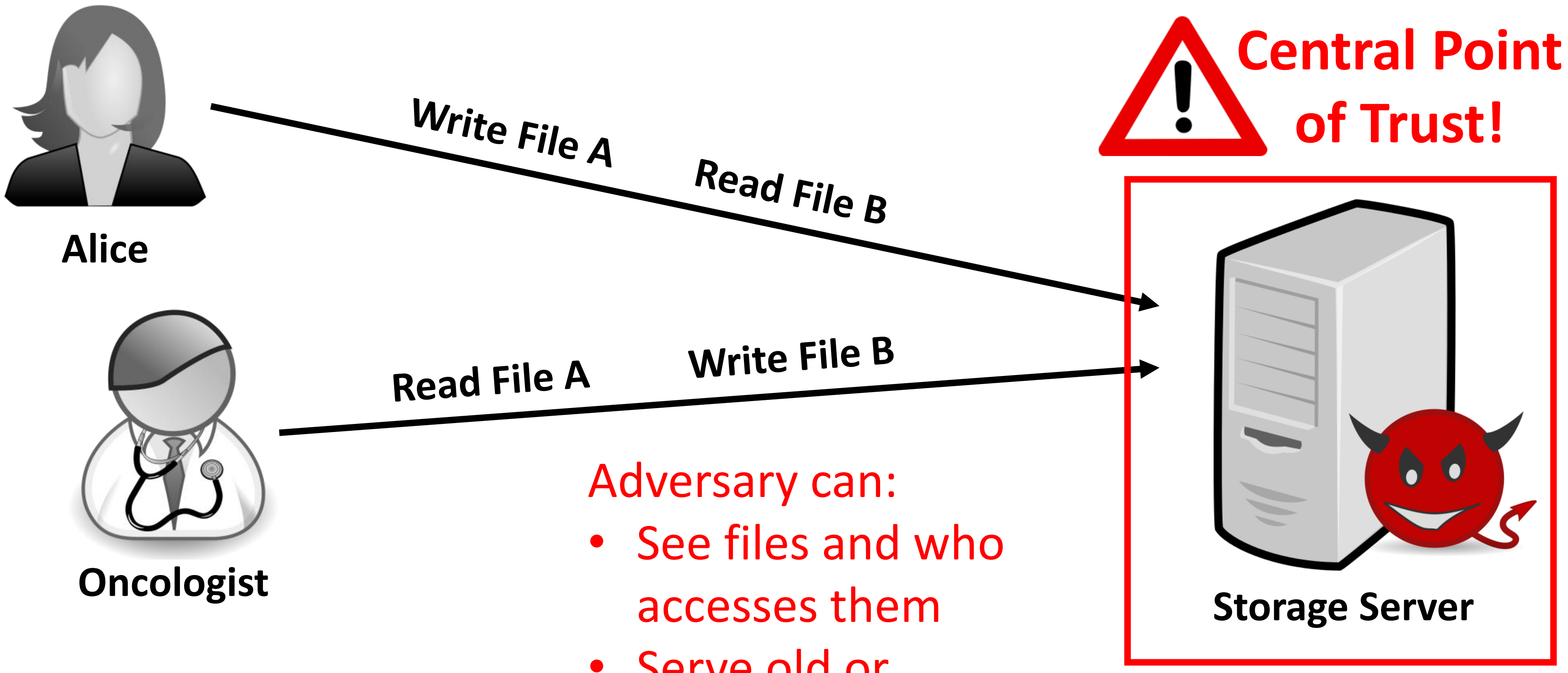
Motivating Example: Medical Record System



Threat Model



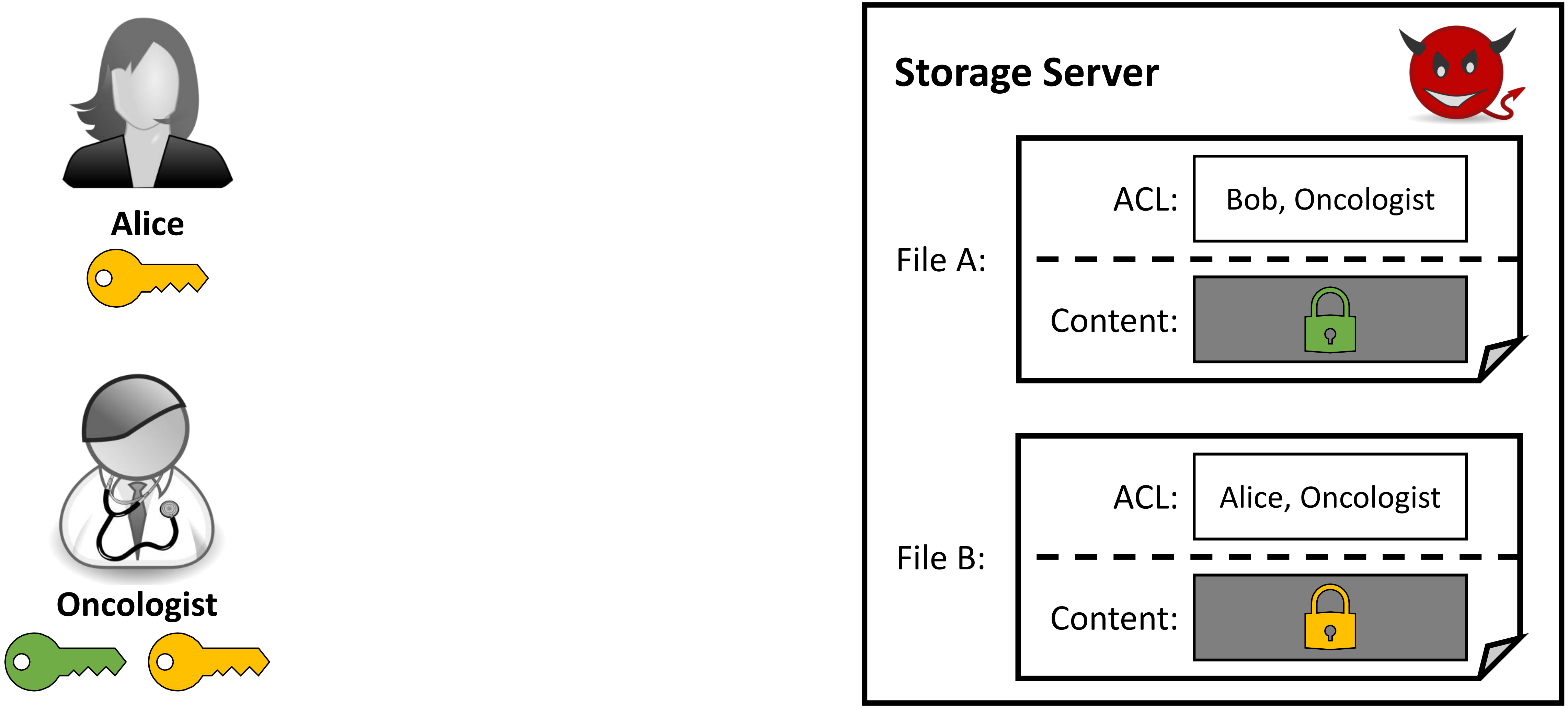
Existing Systems rely on *Centralized Trust*



Adversary can:

- See files and who accesses them
- Serve old or modified data

End-to-End Encryption [CFS, SiRiUS, Plutus, etc.]



Alice

Oncologist

Storage Server

File A:

ACL: Bob, Oncologist

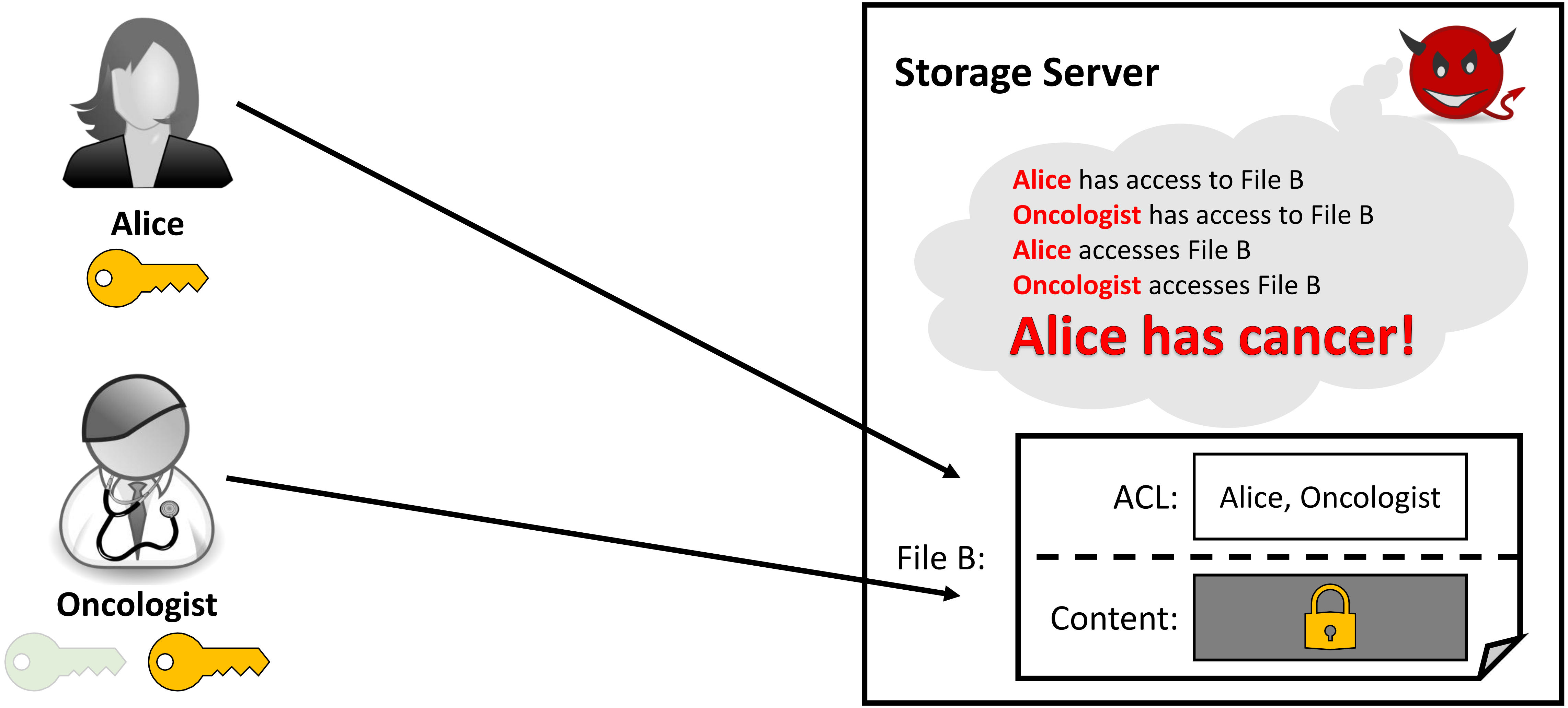
Content:

File B:

ACL: Alice, Oncologist

Content:

Privacy Leakage in E2EE Data Sharing



Alice

Oncologist

Storage Server

Alice has access to File B
Oncologist has access to File B
Alice accesses File B
Oncologist accesses File B

Alice has cancer!

File B:

ACL: Alice, Oncologist

Content:

Ghostor: Cryptographic Data-Sharing System

- Anonymity

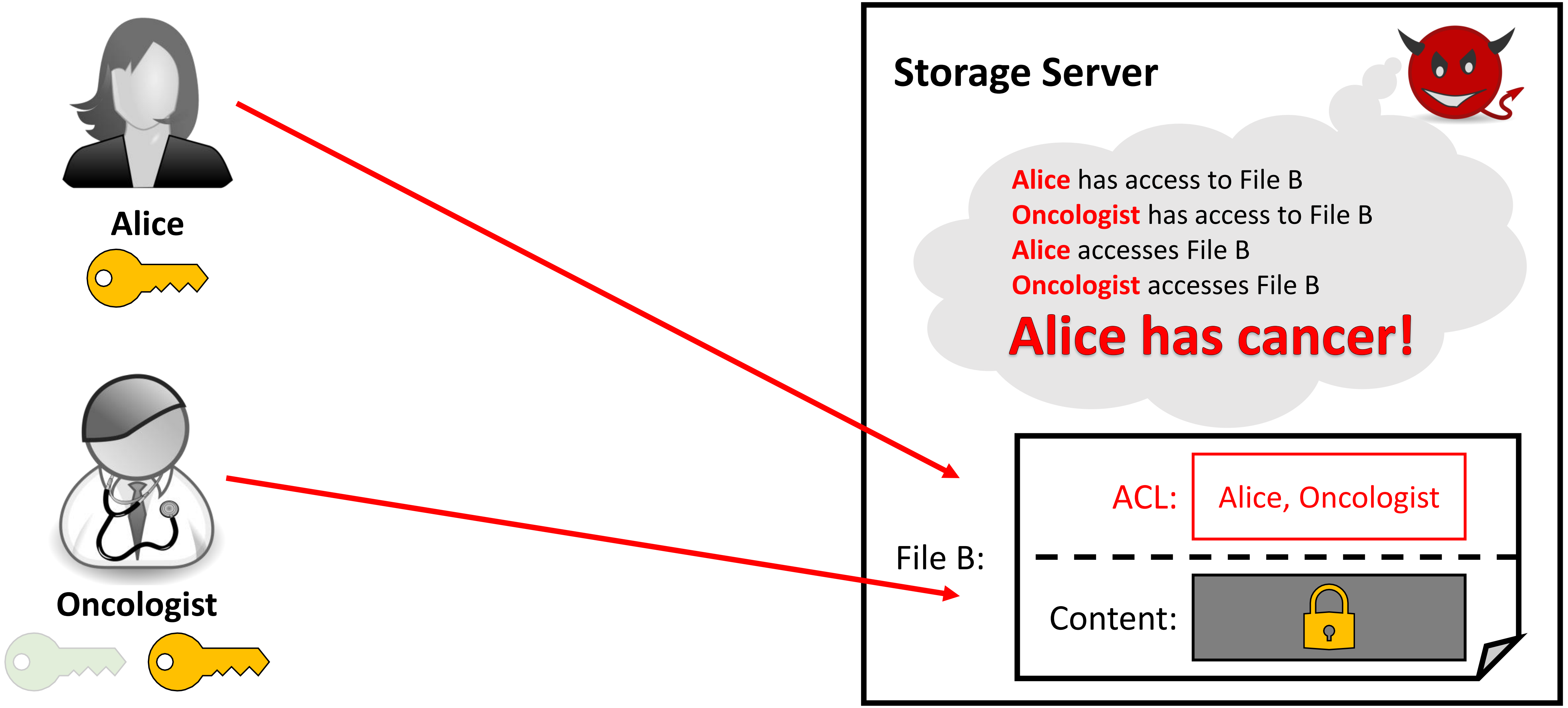
- Verifiable Linearizability

Ghostor: Cryptographic Data-Sharing System

- **Anonymity**

- Verifiable Linearizability

Privacy Leakage in E2EE Data Sharing



Alice

Oncologist

Storage Server

Alice has access to File B
Oncologist has access to File B
Alice accesses File B
Oncologist accesses File B

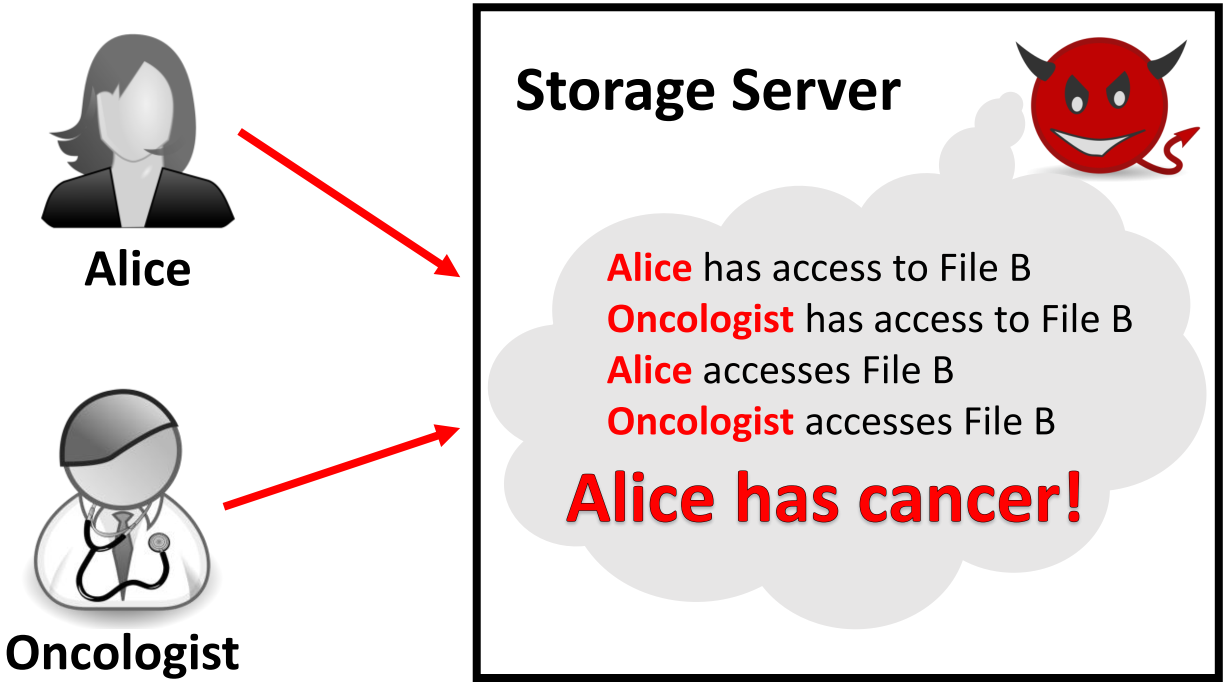
Alice has cancer!

File B:

ACL: Alice, Oncologist

Content:

E2EE Data Sharing vs. Ghostor’s Anonymity



Alice

Oncologist

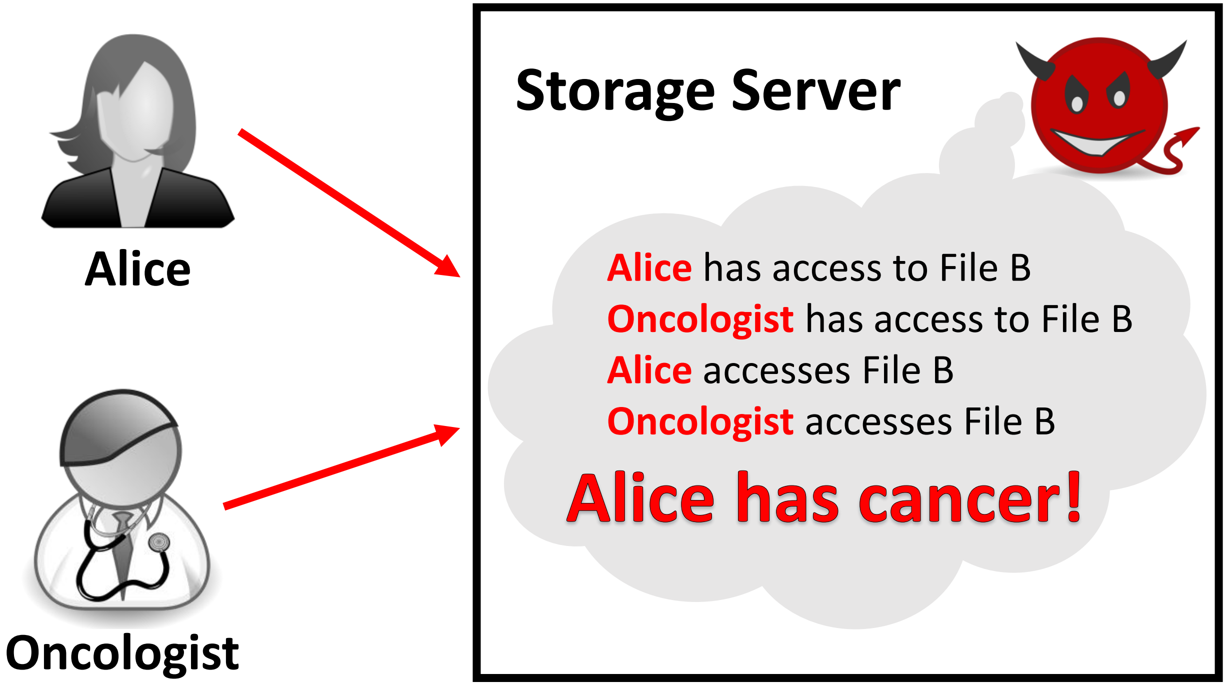
Storage Server

Alice has access to File B
Oncologist has access to File B
Alice accesses File B
Oncologist accesses File B

Alice has cancer!

E2EE Data Sharing

E2EE Data Sharing vs. Ghostor's Anonymity

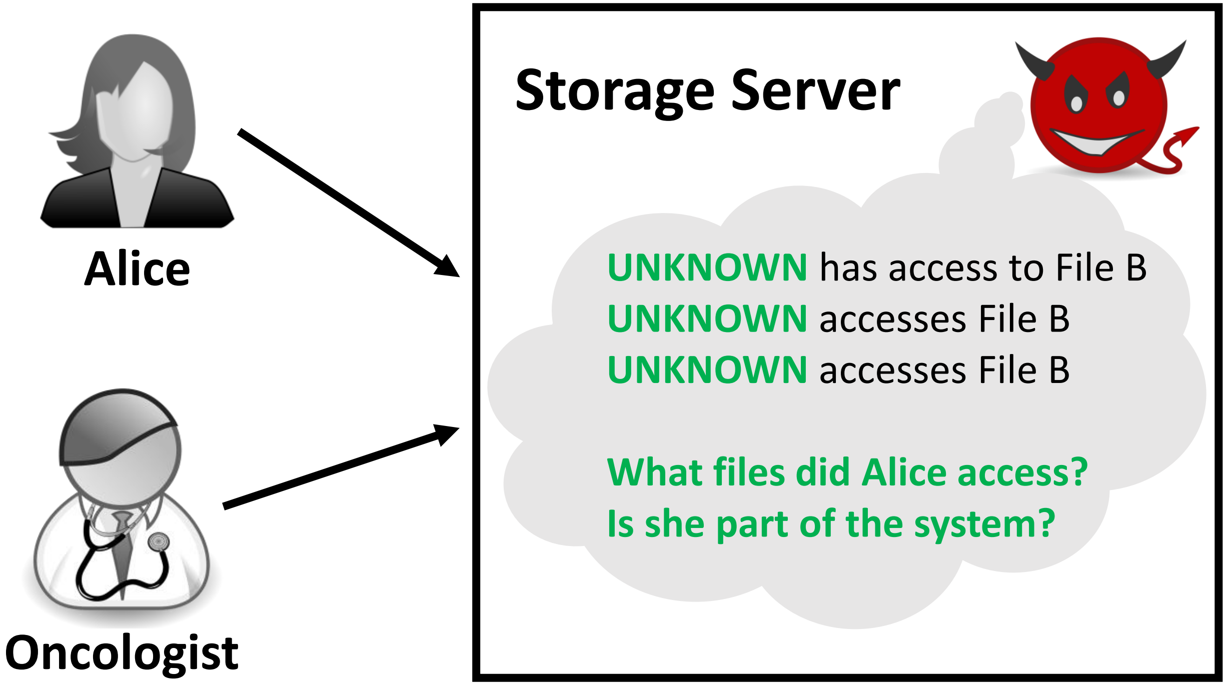


Storage Server

Alice has access to File B
Oncologist has access to File B
Alice accesses File B
Oncologist accesses File B

Alice has cancer!

E2EE Data Sharing



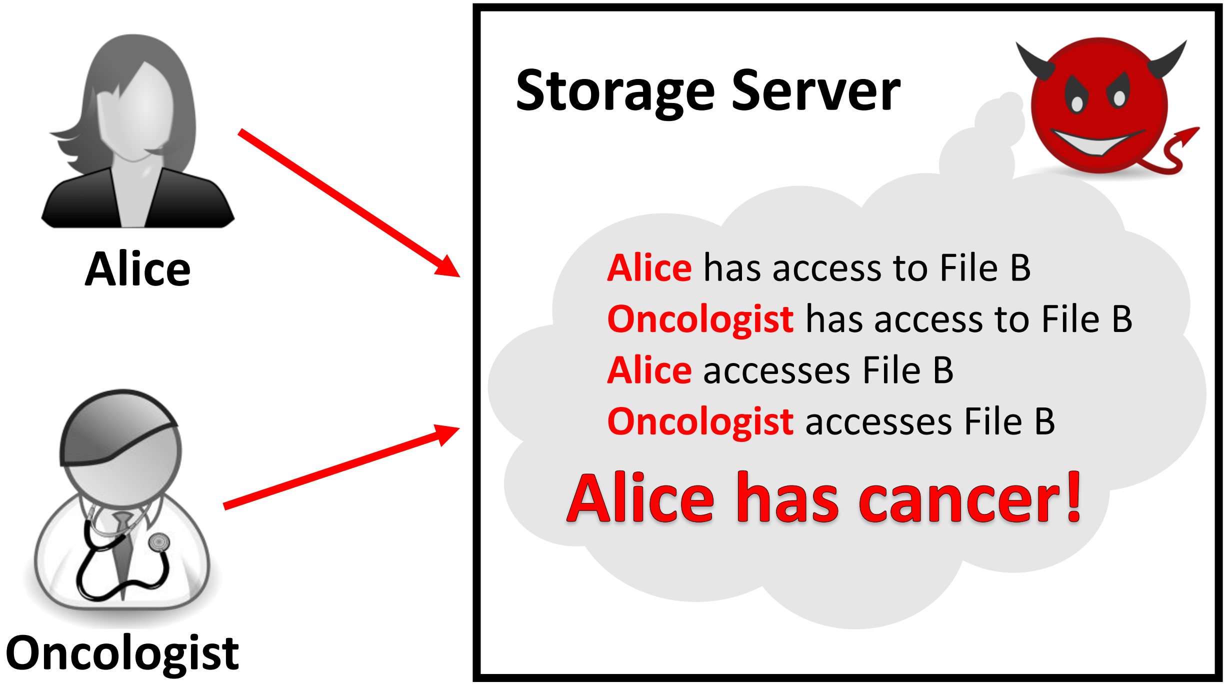
Storage Server

UNKNOWN has access to File B
UNKNOWN accesses File B
UNKNOWN accesses File B

What files did Alice access?
Is she part of the system?

Ghostor's Anonymity

E2EE Data Sharing vs. Ghostor's Anonymity

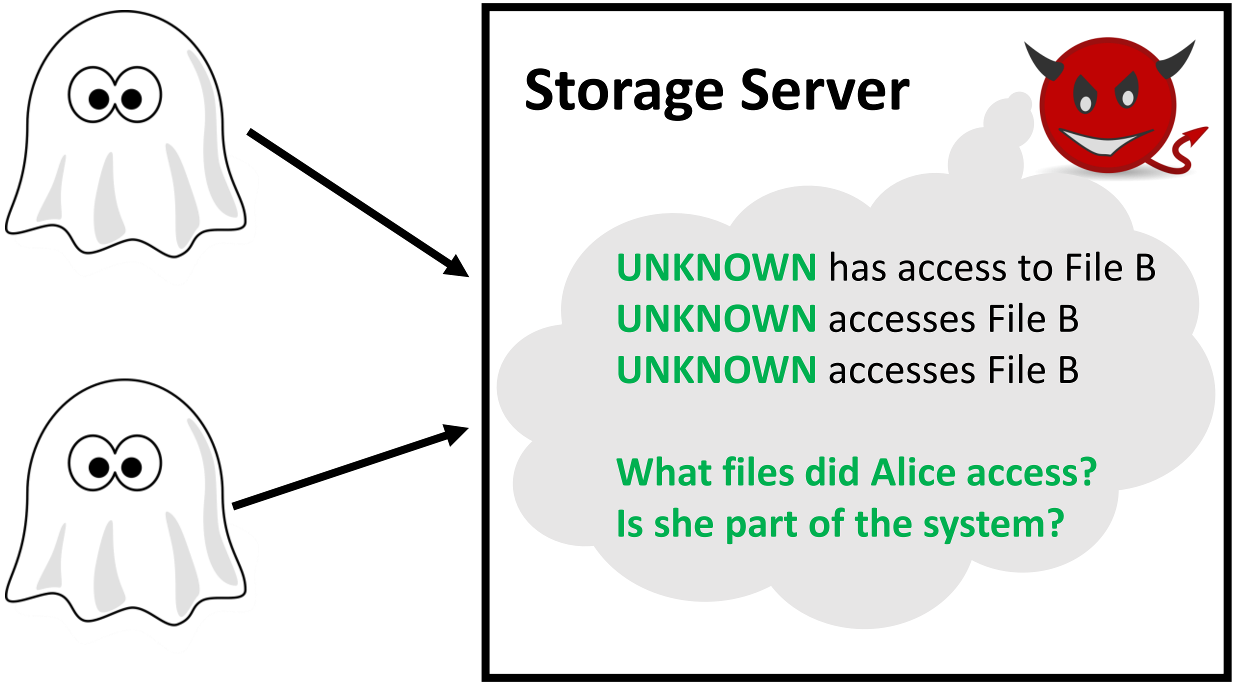


Storage Server

Alice has access to File B
Oncologist has access to File B
Alice accesses File B
Oncologist accesses File B

Alice has cancer!

E2EE Data Sharing



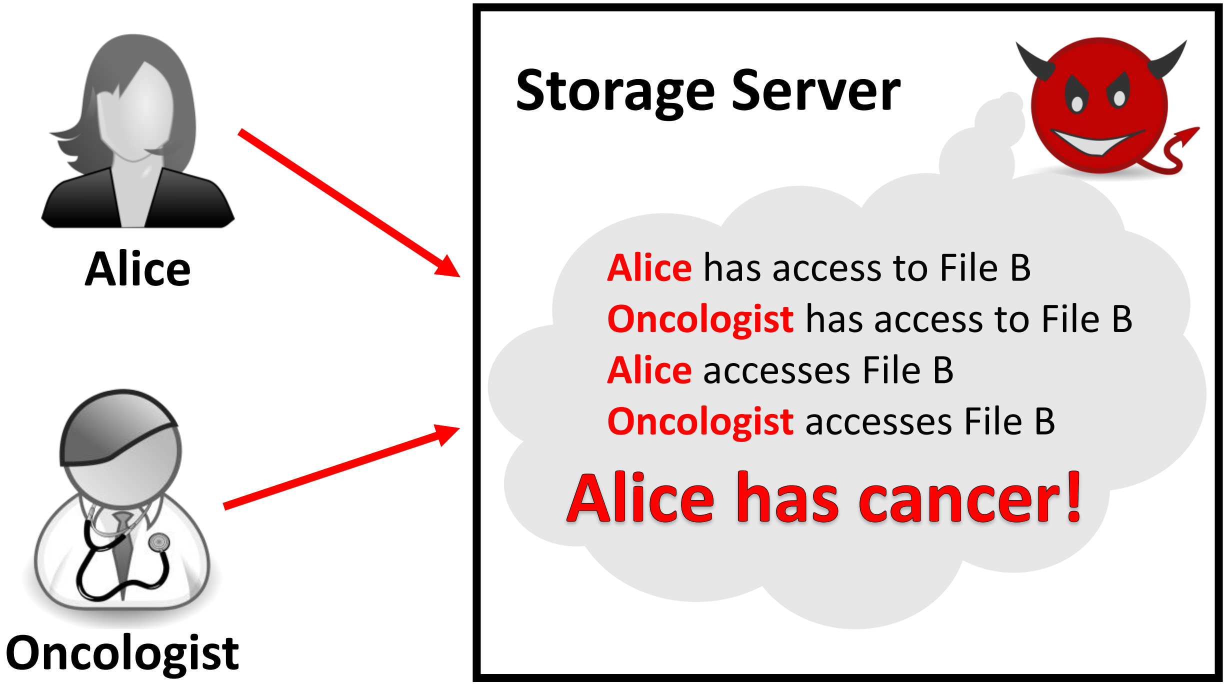
Storage Server

UNKNOWN has access to File B
UNKNOWN accesses File B
UNKNOWN accesses File B

What files did Alice access?
Is she part of the system?

Ghostor's Anonymity

E2EE Data Sharing vs. Ghostor's Anonymity



Alice

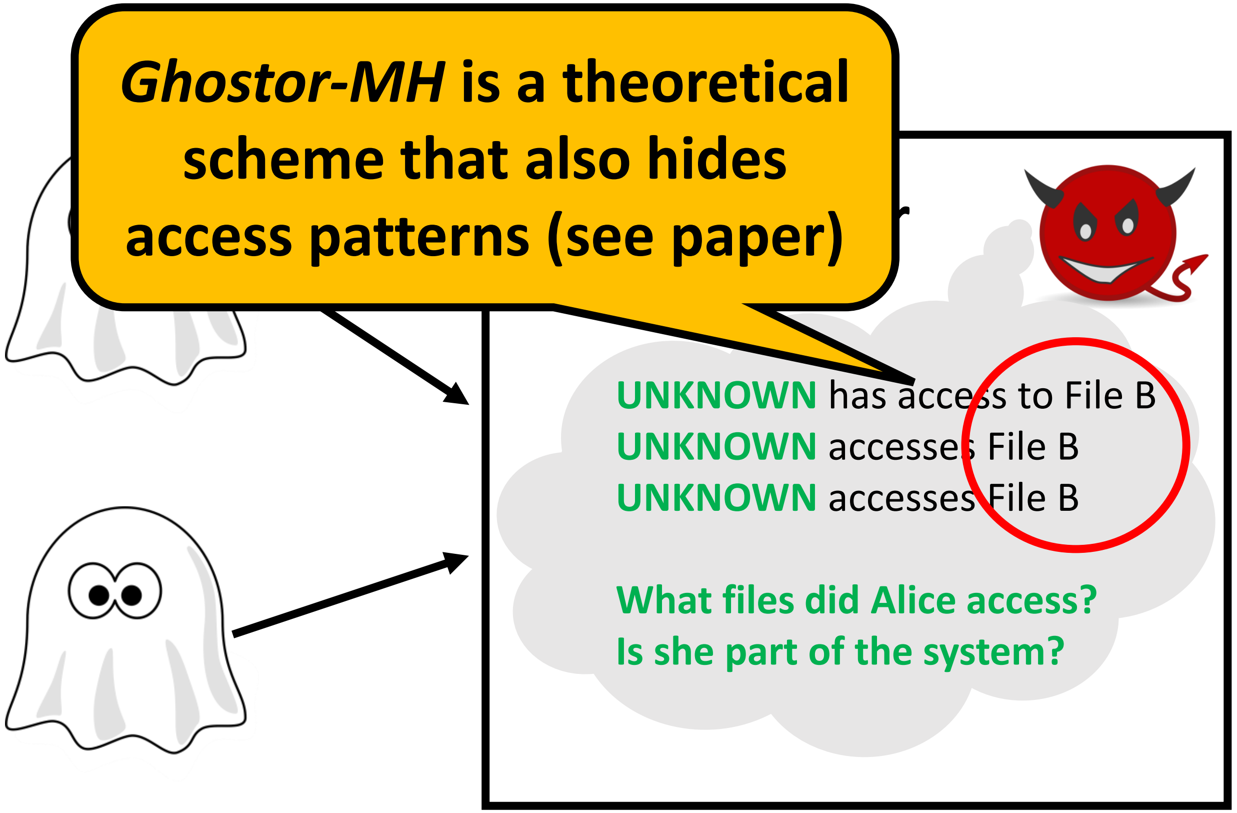
Oncologist

Storage Server

Alice has access to File B
Oncologist has access to File B
Alice accesses File B
Oncologist accesses File B

Alice has cancer!

E2EE Data Sharing



***Ghostor-MH* is a theoretical scheme that also hides access patterns (see paper)**

UNKNOWN has access to File B
UNKNOWN accesses File B
UNKNOWN accesses File B

What files did Alice access?
Is she part of the system?

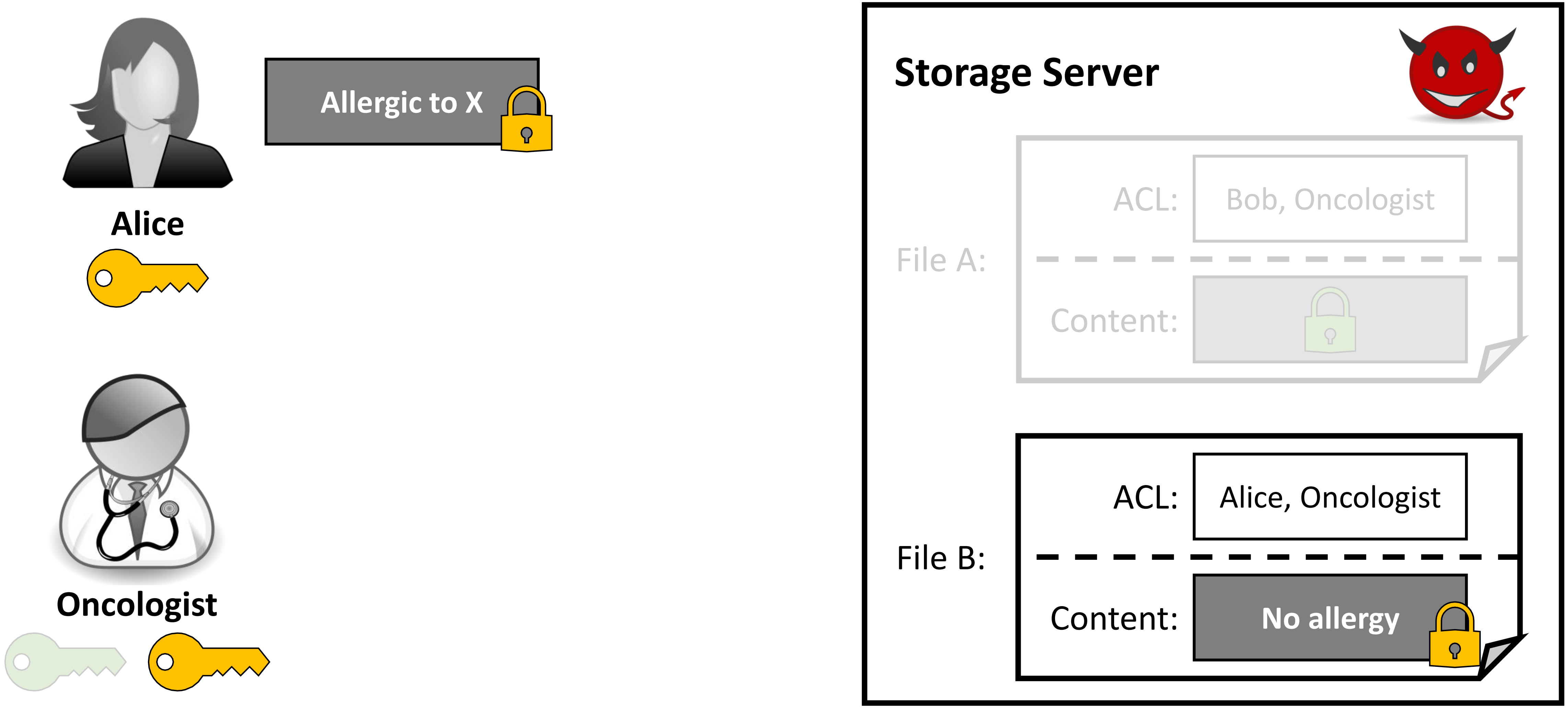
Ghostor's Anonymity

Ghostor: Cryptographic Data-Sharing System

- Anonymity

- **Verifiable Linearizability**

Verifiable Linearizability



Allergic to X

Alice

Oncologist

Storage Server

File A:

ACL: Bob, Oncologist

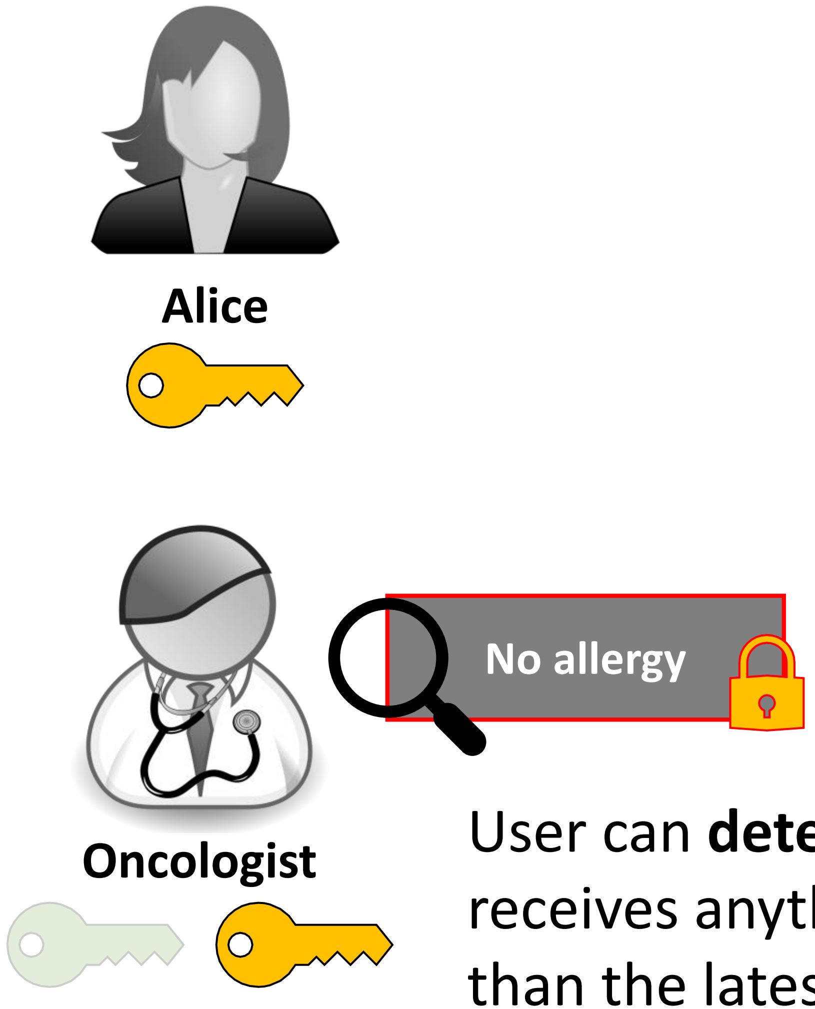
Content:

File B:

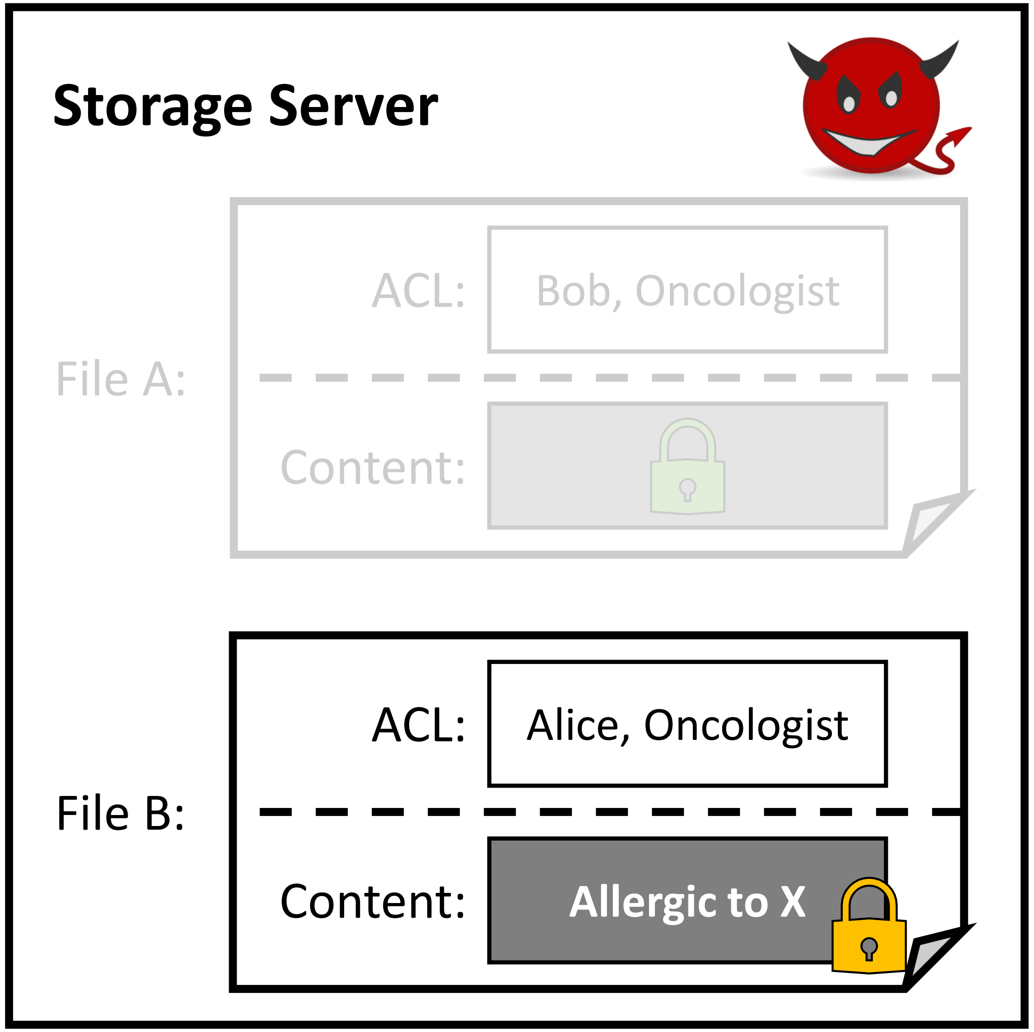
ACL: Alice, Oncologist

Content: No allergy

Verifiable Linearizability



User can **detect** if he receives anything other than the latest write



Comparison to Existing Work

Verena [KFPC16]: Verifiable Linearizability

AnonymousCloud [KH12]: Anonymity

Rely on Central Trust

- Split server into two parts and assume one is honest, or
- Assume semi-honest adversary

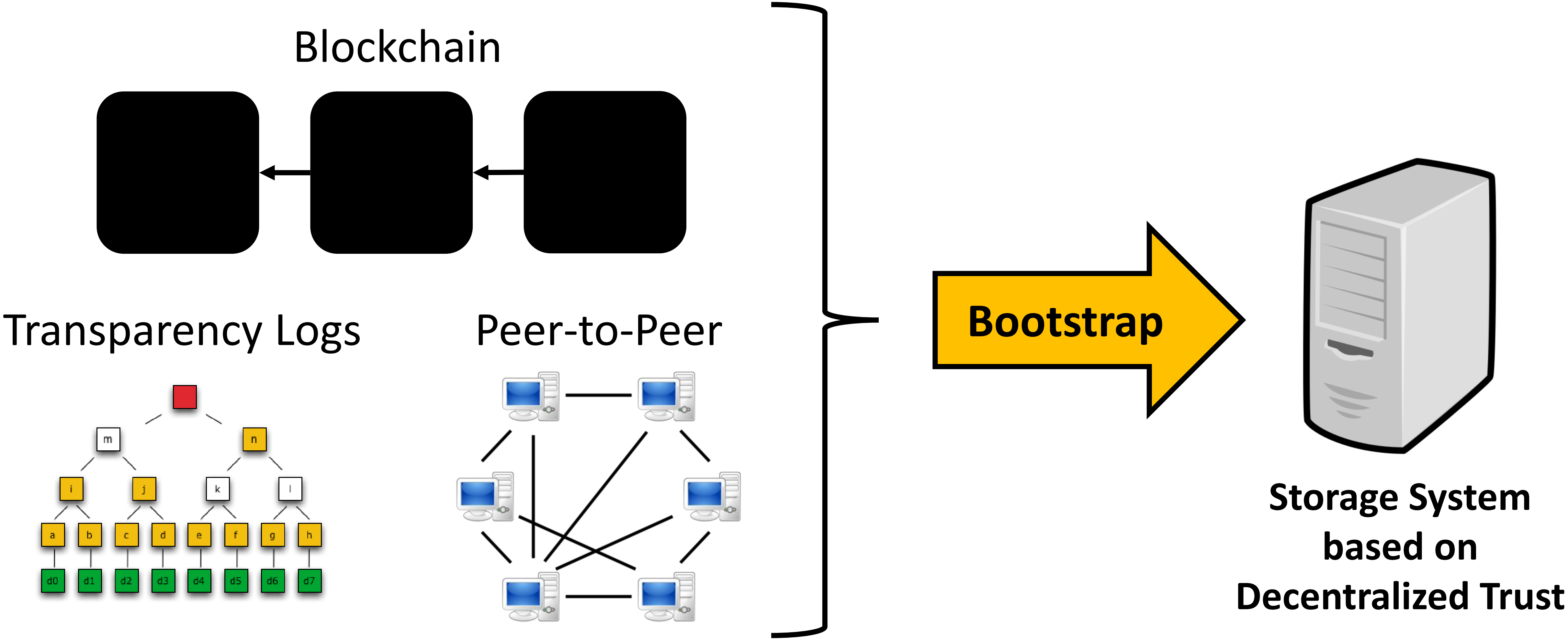
Ghostor

Anonymity and Verifiable Linearizability

Based on *Decentralized Trust*

- Avoids placing trust in a *few central* machines

Bootstrapping Decentralized Trust



Blockchain

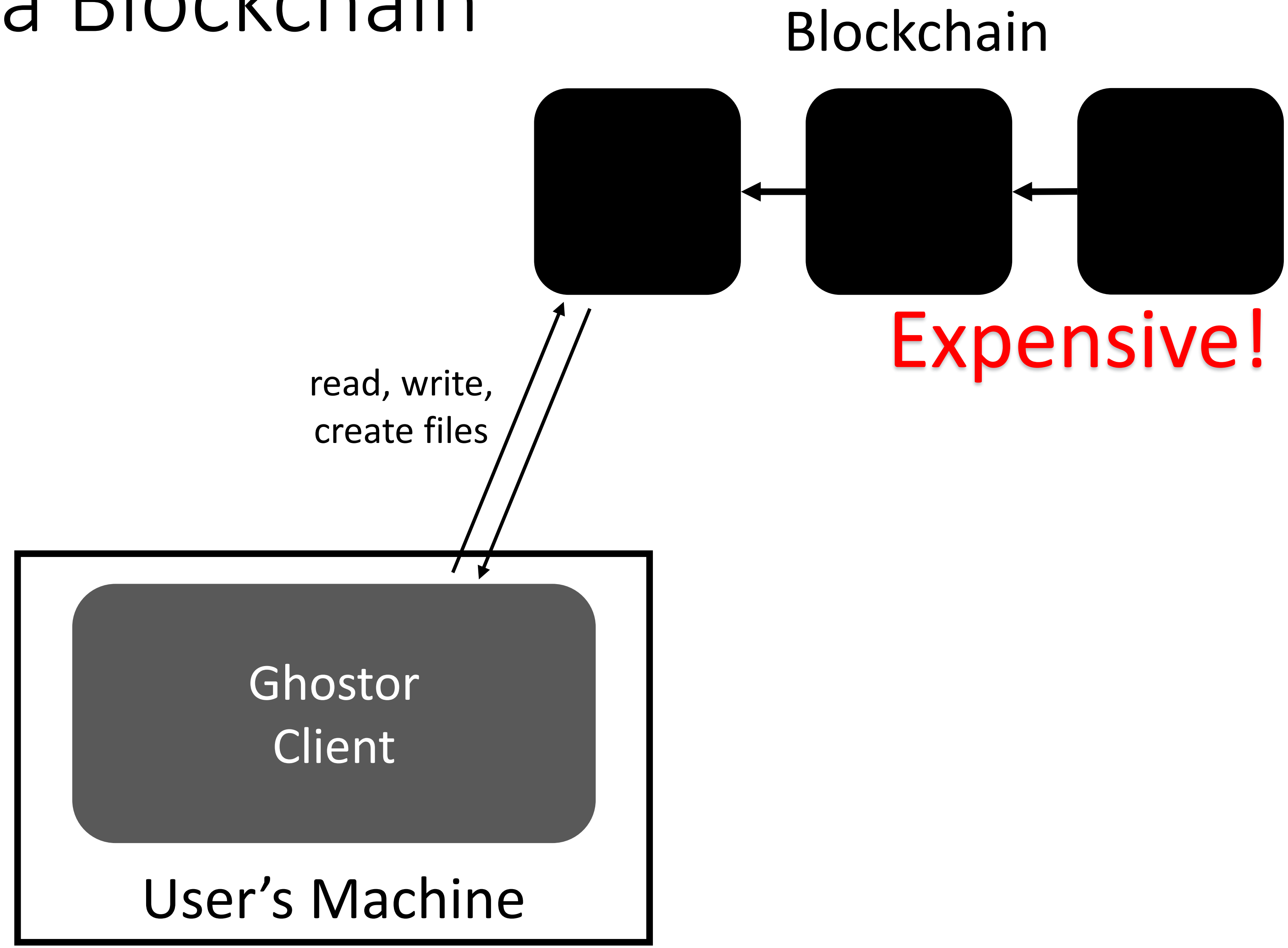
Transparency Logs

Peer-to-Peer

Bootstrap

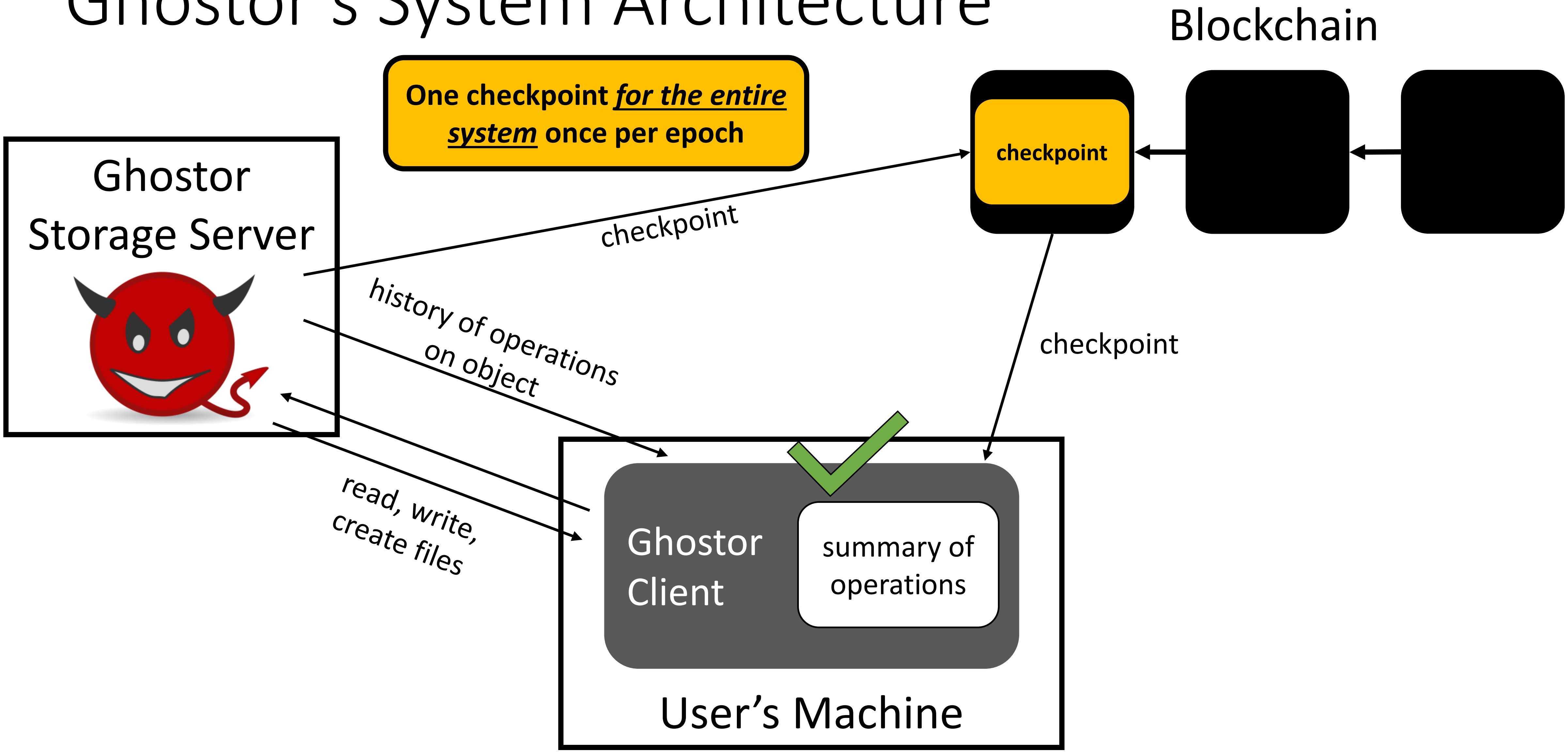
Storage System based on Decentralized Trust

Strawman: Use a Blockchain

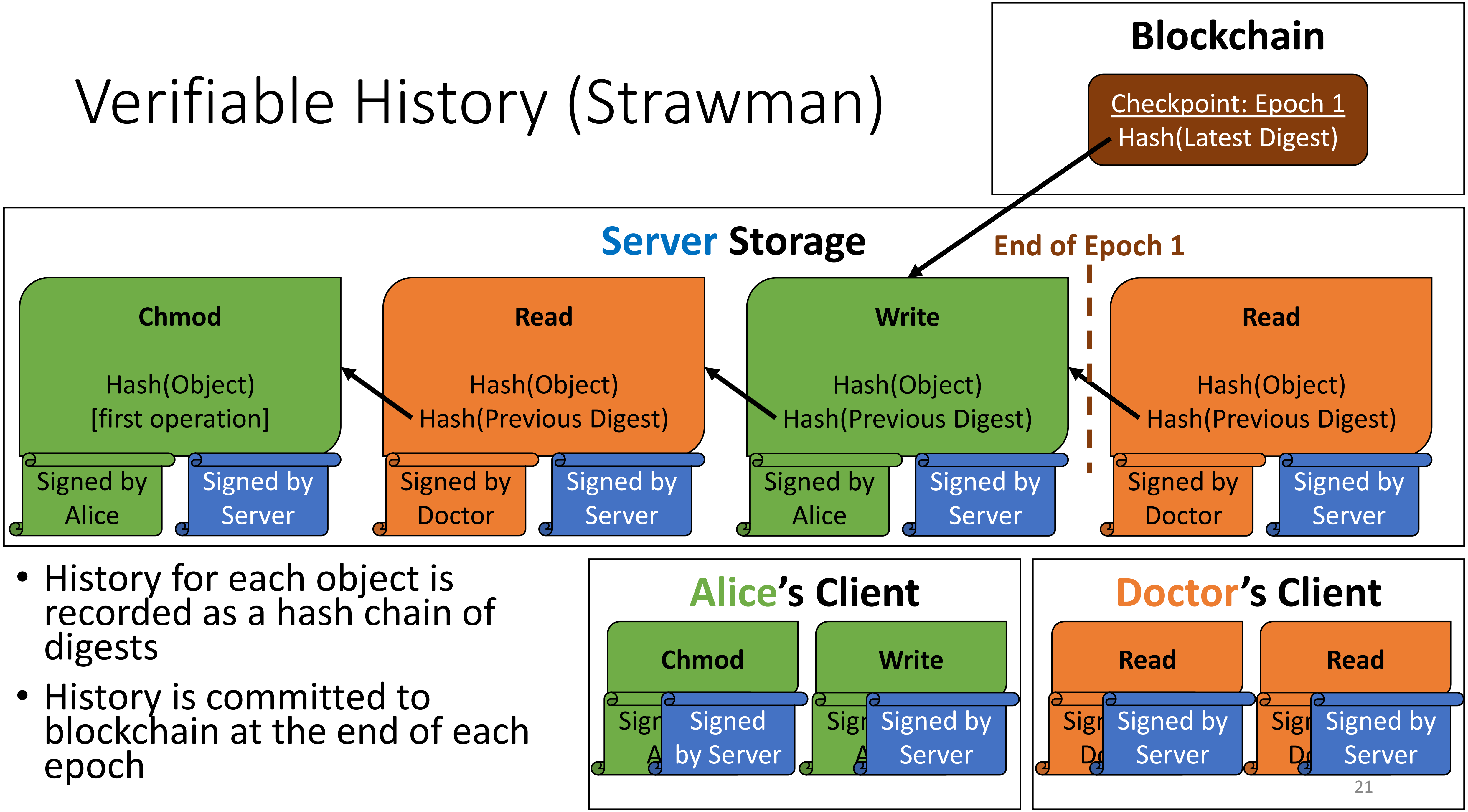


Ghostor’s System Architecture

One checkpoint *for the entire system* once per epoch



Verifiable History (Strawman)



- History for each object is recorded as a hash chain of digests
- History is committed to blockchain at the end of each epoch

How to make Verifiable History ***Anonymous***?

- Signing keys are like *capabilities*
- Idea: have different users *share capabilities* for each object

Shared Capabilities

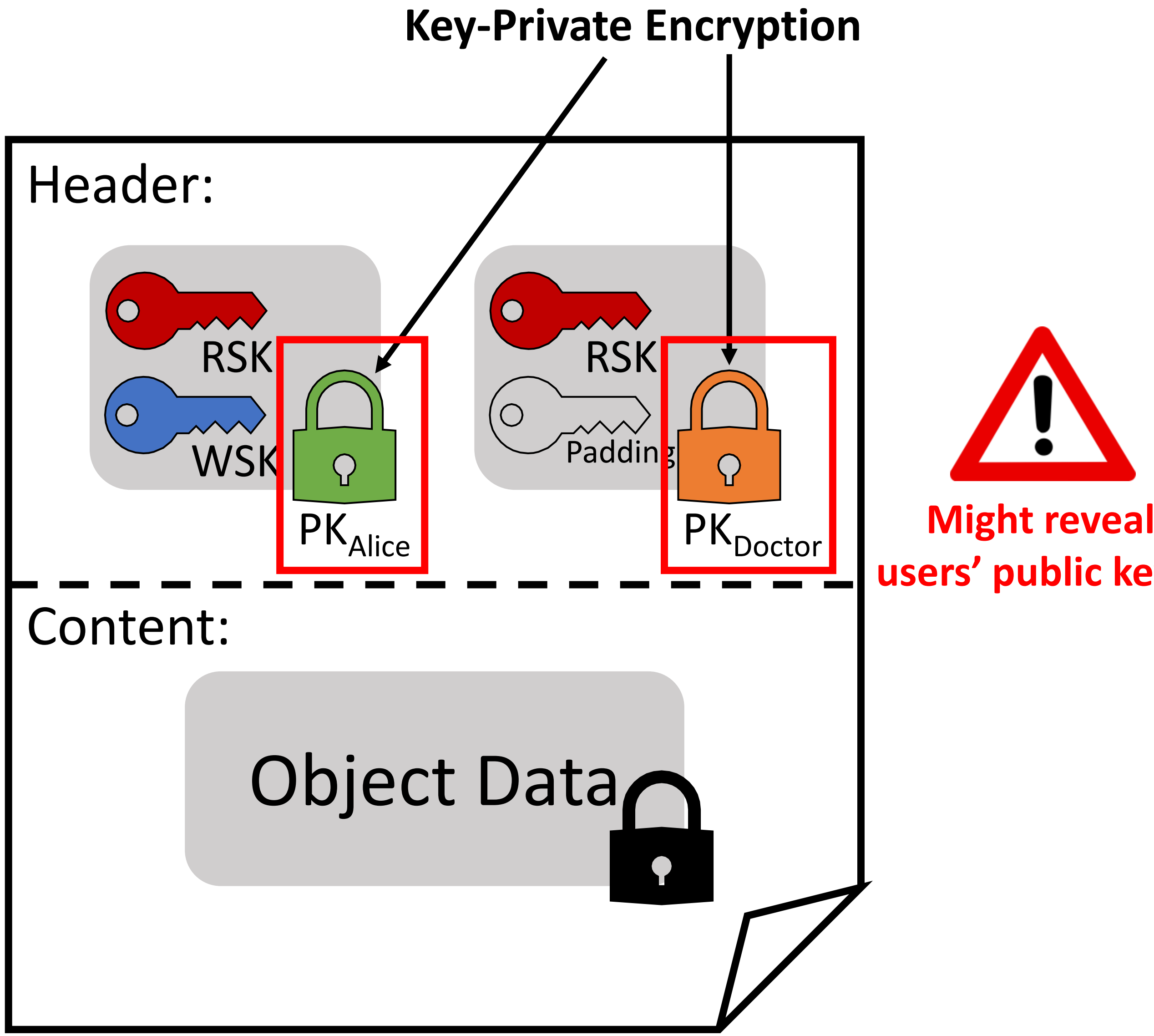
Permission Signing Key PSK

Stored by the object's owner

Reader Signing Key RSK

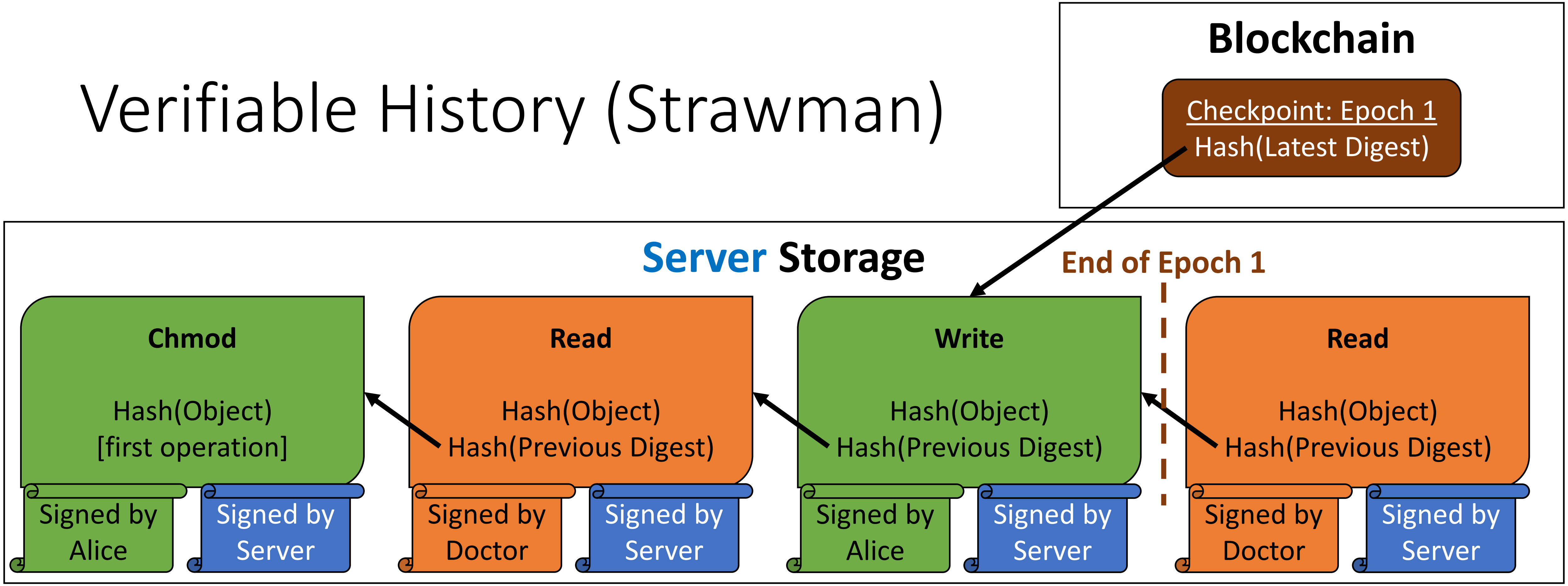
Writer Signing Key WSK

Anonymously Distributed Shared Capabilities



Might reveal users' public keys!

Verifiable History (Strawman)



Blockchain

Checkpoint: Epoch 1

Hash(Latest Digest)

Server Storage

End of Epoch 1

Chmod

Hash(Object)

[first operation]

Signed by Alice

Signed by Server

Read

Hash(Object)

Hash(Previous Digest)

Signed by Doctor

Signed by Server

Write

Hash(Object)

Hash(Previous Digest)

Signed by Alice

Signed by Server

Read

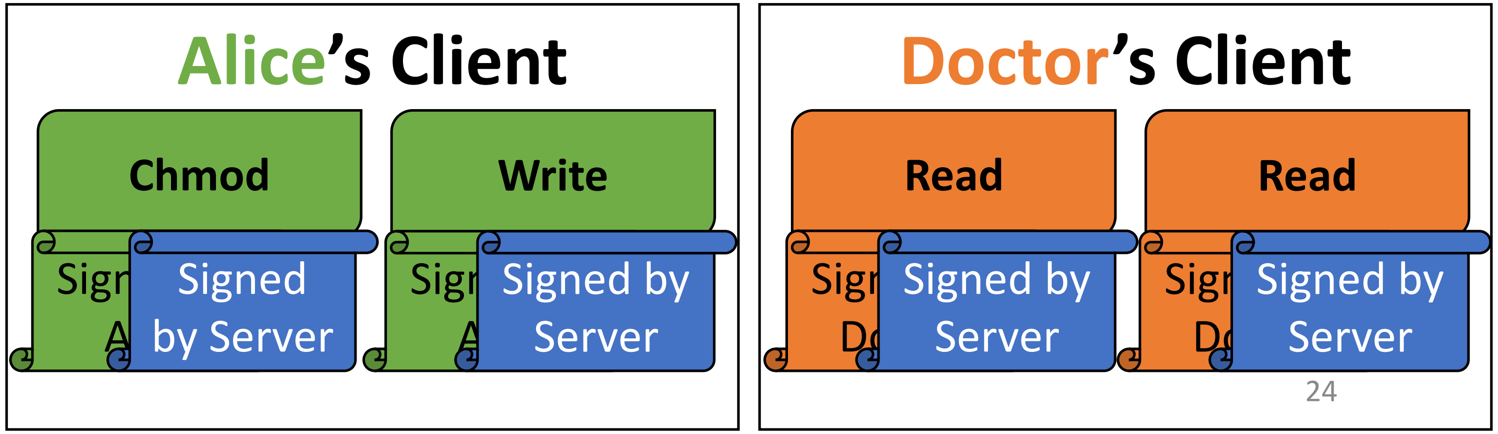
Hash(Object)

Hash(Previous Digest)

Signed by Doctor

Signed by Server

- History for each object is recorded as a hash chain of digests
- History is committed to blockchain at the end of each epoch



Alice's Client

Chmod

Sign A

Signed by Server

Write

Sign A

Signed by Server

Doctor's Client

Read

Sign Do

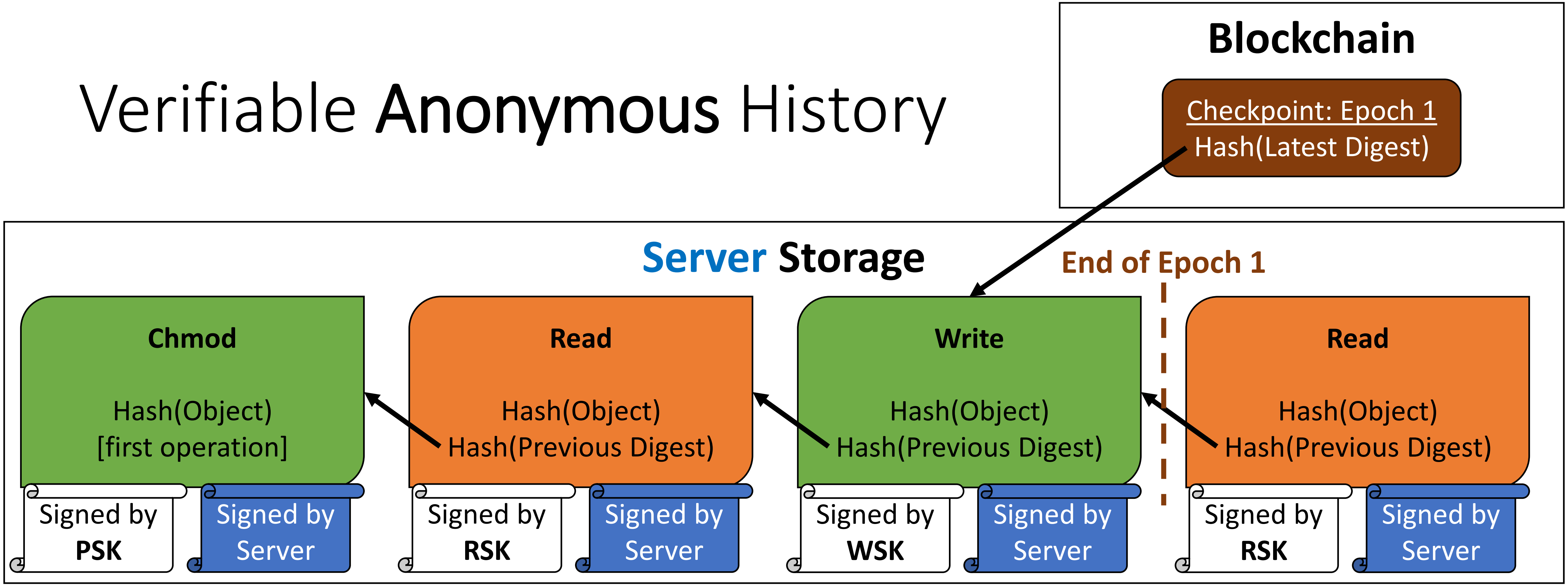
Signed by Server

Read

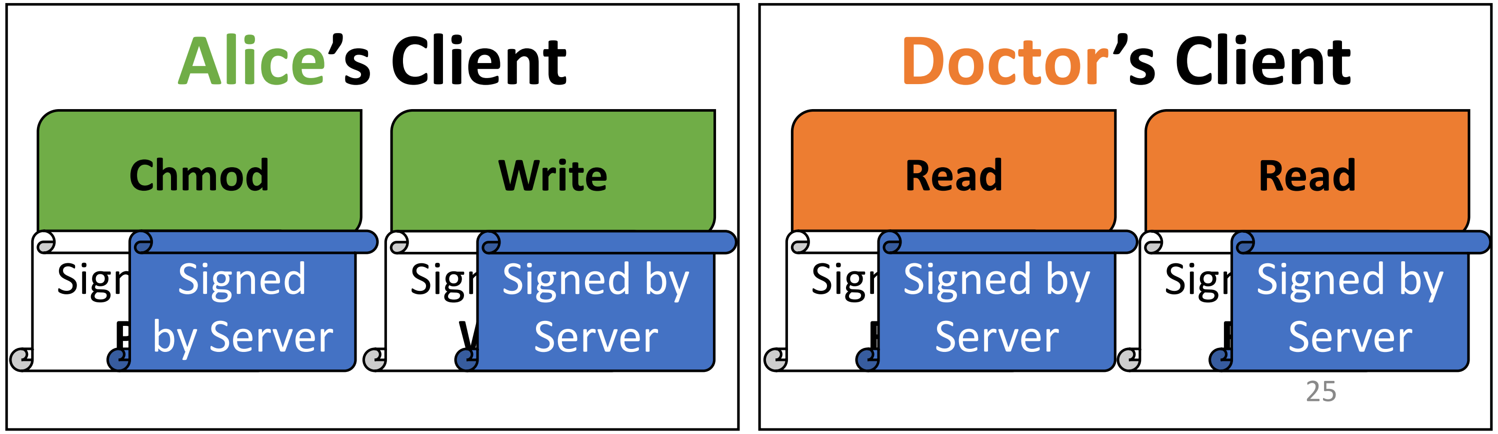
Sign Do

Signed by Server

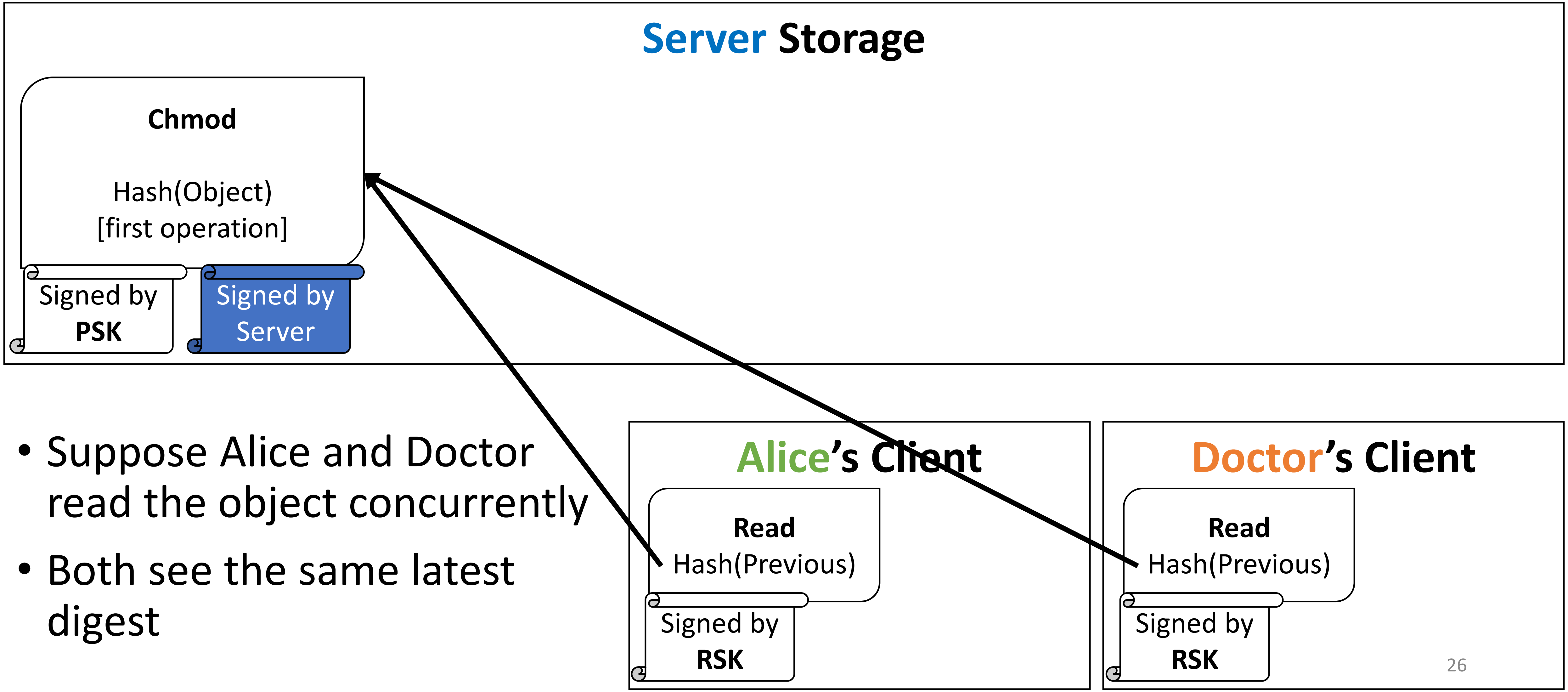
Verifiable **Anonymous** History



- History for each object is recorded as a hash chain of digests
- History is committed to blockchain at the end of each epoch

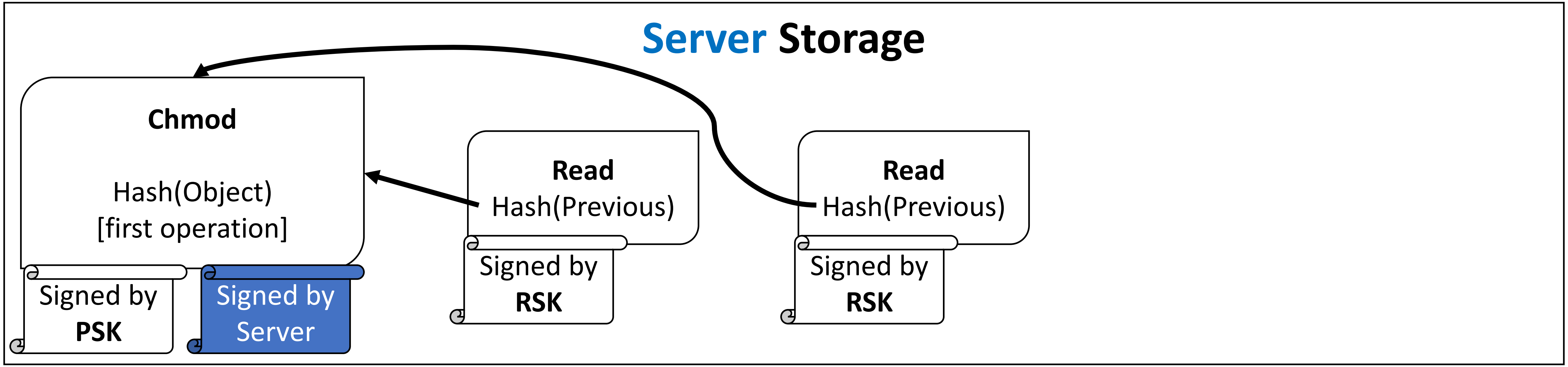


Additional Challenge: Concurrent Operations



- Suppose Alice and Doctor read the object concurrently
- Both see the same latest digest

Additional Challenge: Concurrent Operations

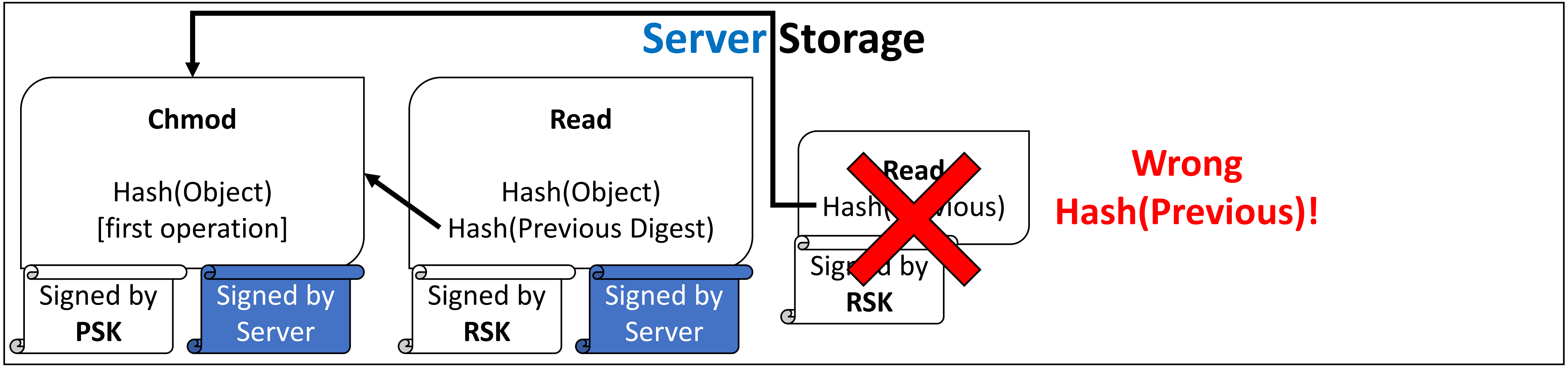


- Suppose Alice and Doctor read the object concurrently
- Both see the same latest digest

Alice’s Client

Doctor’s Client

Additional Challenge: Concurrent Operations

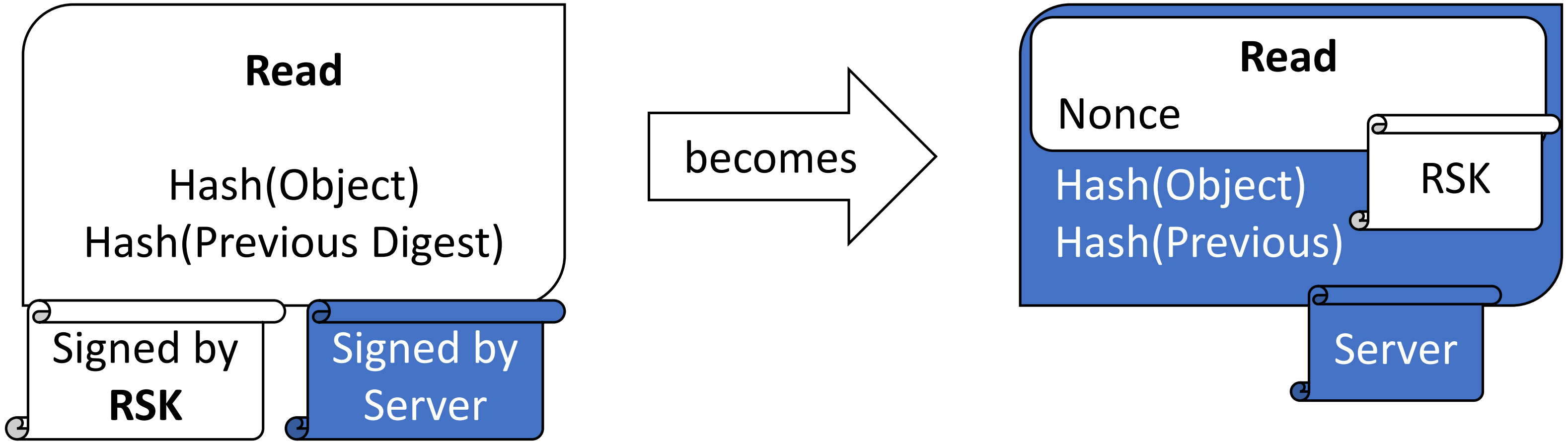


- Suppose Alice and Doctor read the object concurrently
- Both see the same latest digest

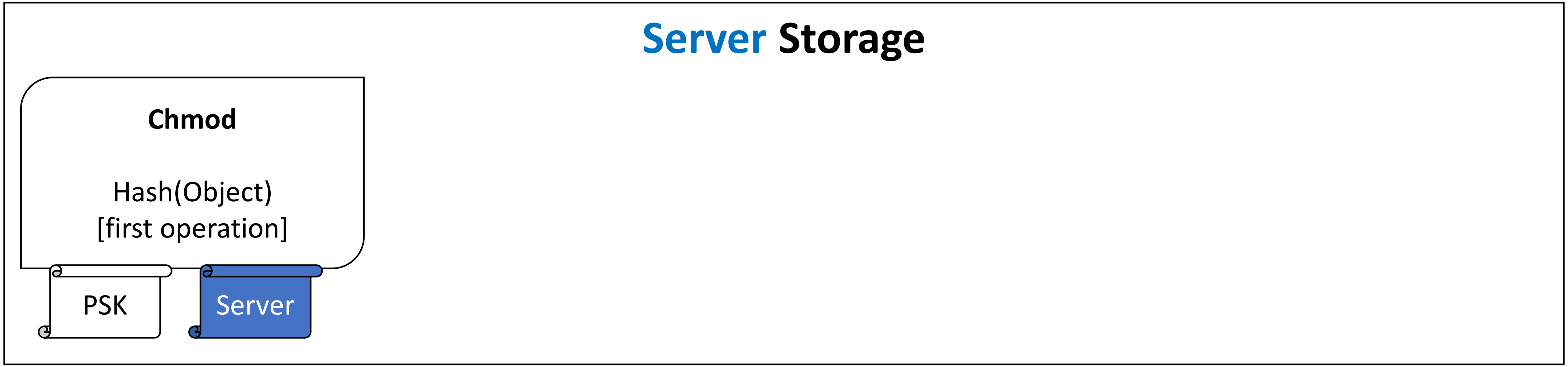
Alice's Client

Doctor's Client

Insight: Client Signs over only Some Fields



Concurrent Reads in Ghostor



Server Storage

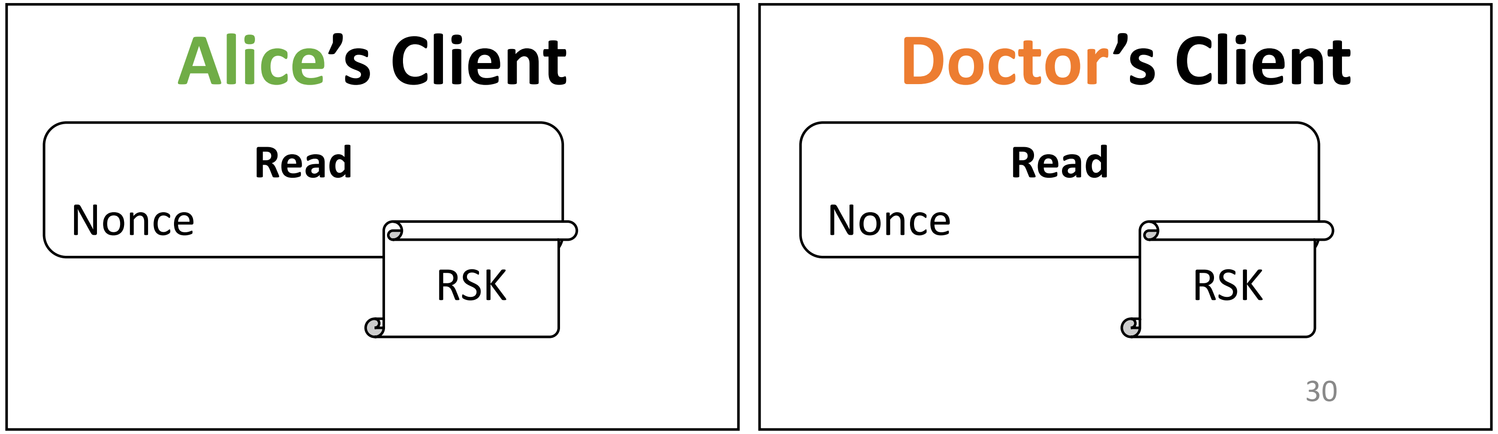
Chmod

Hash(Object)
[first operation]

PSK

Server

- Suppose Alice and Doctor read the object concurrently
- Both see the same latest digest



Alice's Client

Read

Nonce

RSK

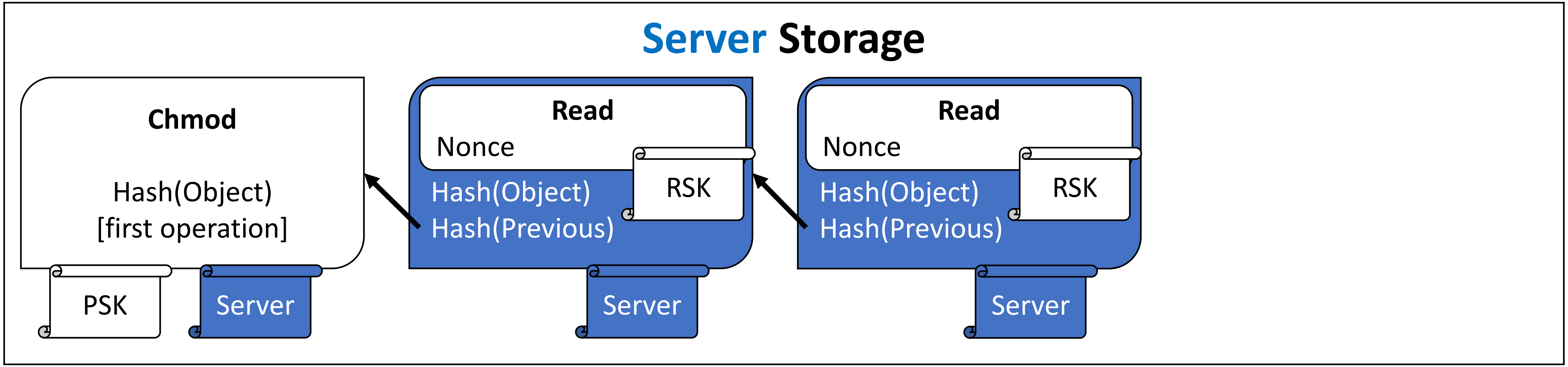
Doctor's Client

Read

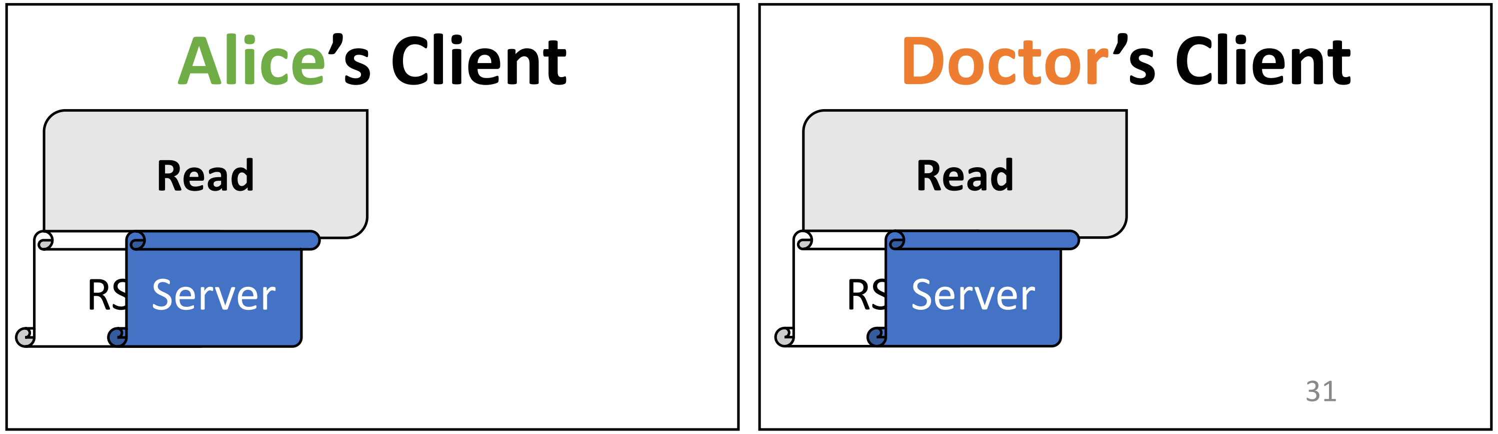
Nonce

RSK

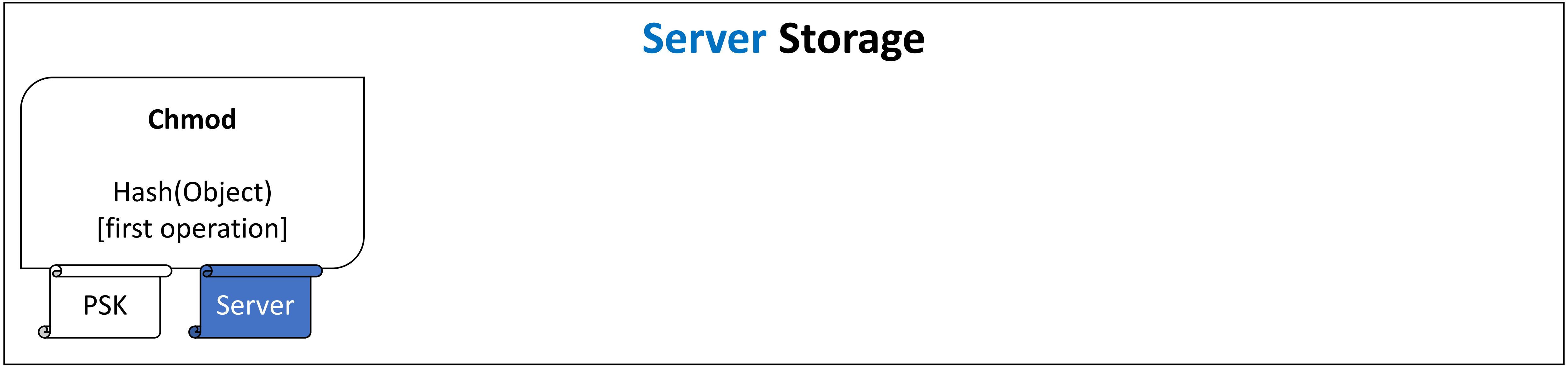
Concurrent Reads in Ghostor



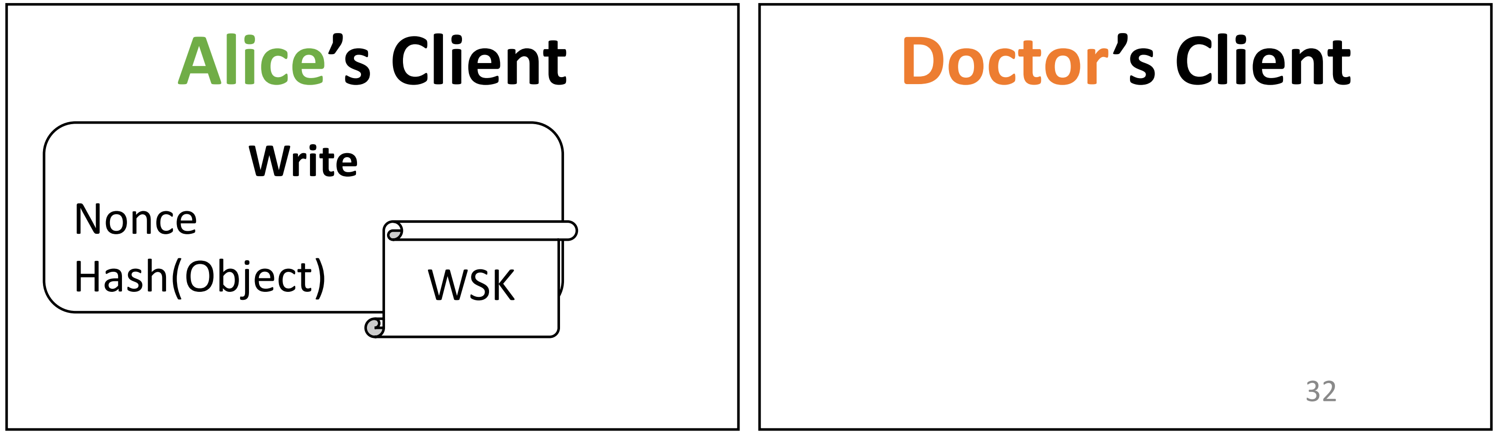
- Suppose Alice and Doctor read the object concurrently
- Both see the same latest digest



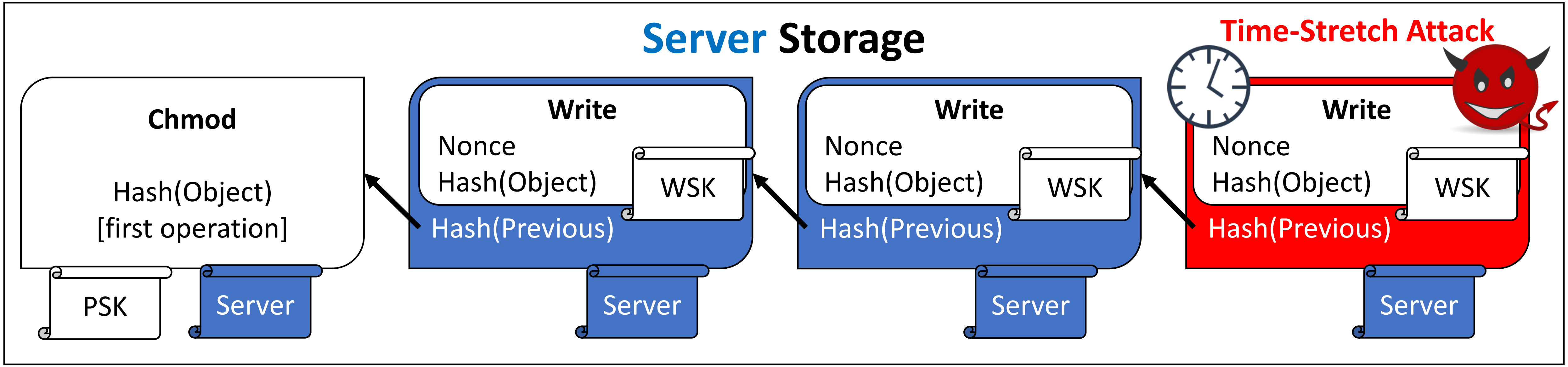
This Technique Does Not Work for Writes



- Suppose Alice writes the file



This Technique Does Not Work for Writes



Server Storage

Time-Stretch Attack

Chmod

Hash(Object)
[first operation]

PSK

Server

Write

Nonce
Hash(Object)

WSK

Hash(Previous)

Server

Write

Nonce
Hash(Object)

WSK

Hash(Previous)

Server

Write

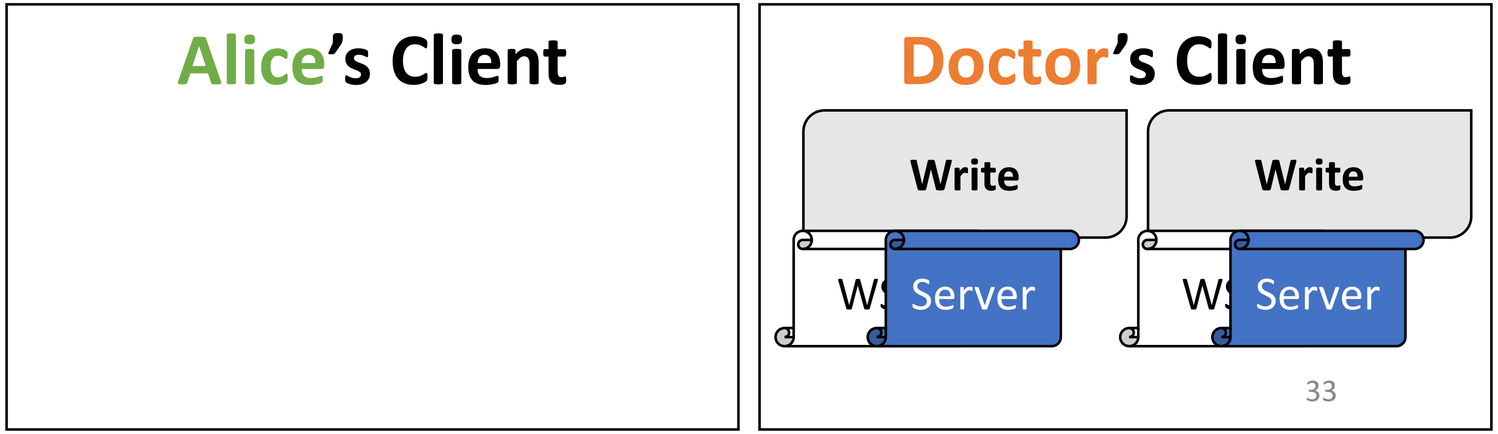
Nonce
Hash(Object)

WSK

Hash(Previous)

Server

- Suppose Alice writes the file



Alice's Client

Doctor's Client

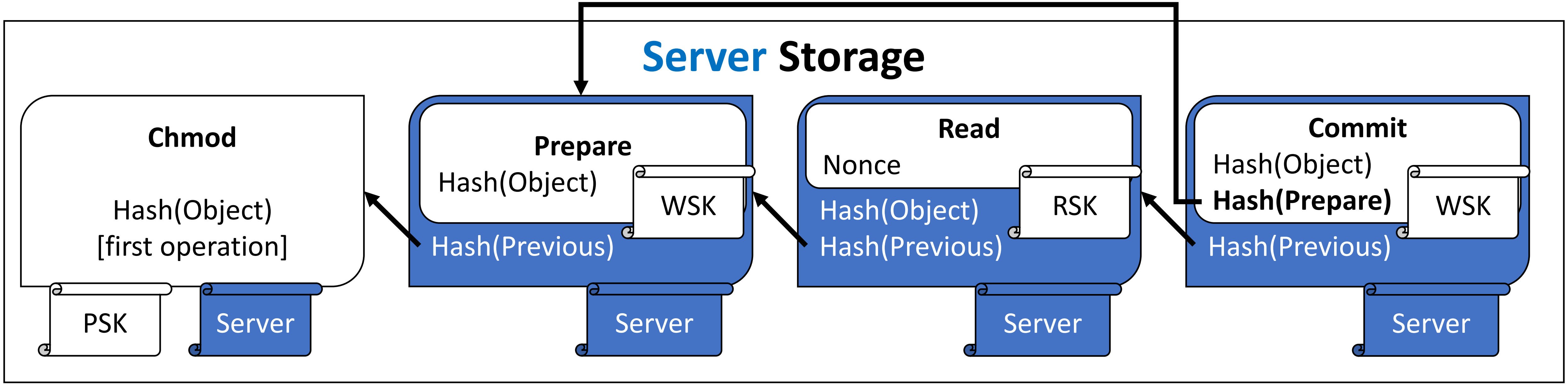
Write

WS Server

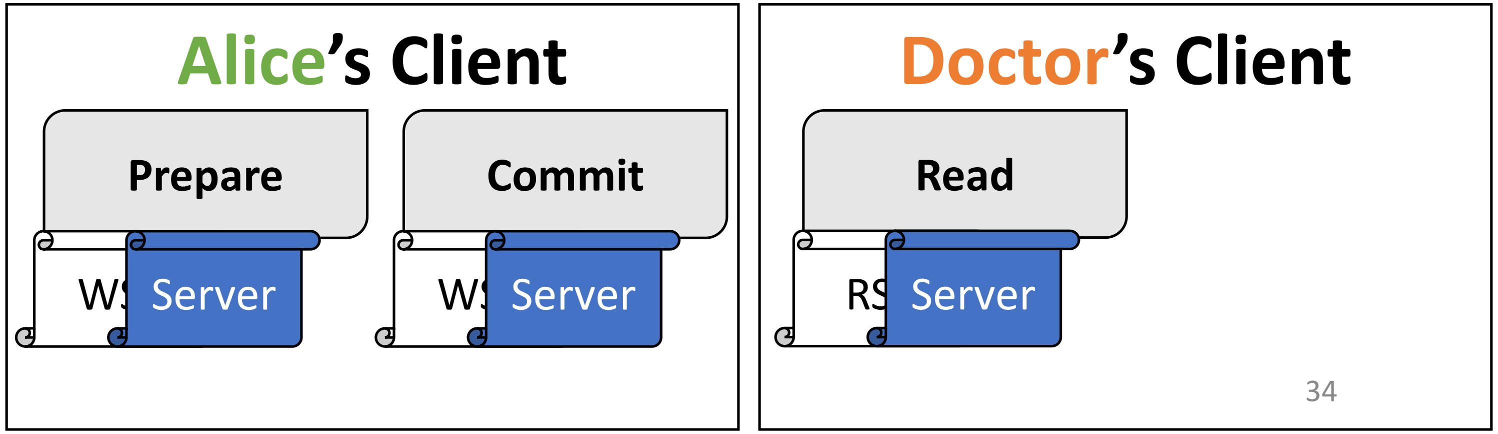
Write

WS Server

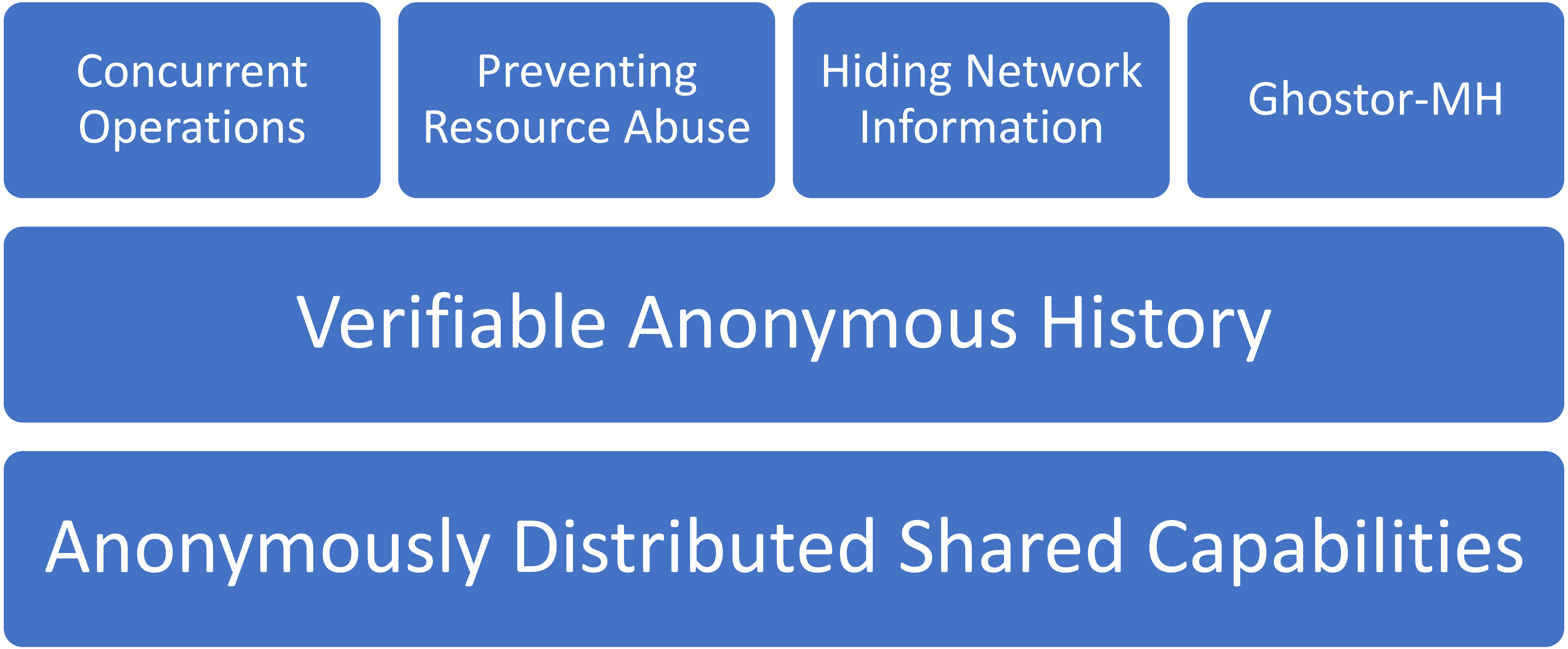
Concurrent Writes in Ghostor



- Suppose Alice writes the file



Ghostor Stack



Concurrent Operations

Preventing Resource Abuse

Hiding Network Information

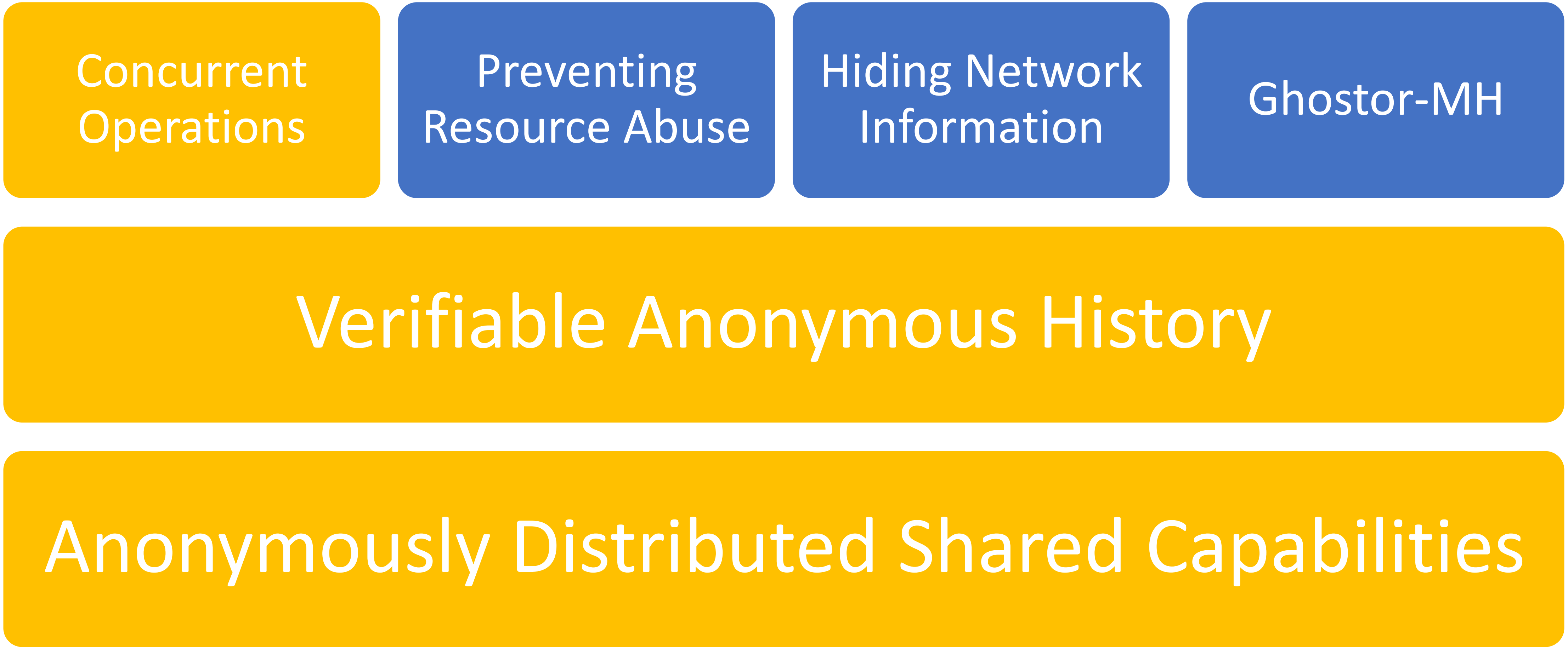
Ghostor-MH

Verifiable Anonymous History

Anonymously Distributed Shared Capabilities

Ghostor Stack

Described in our paper



Concurrent Operations

Preventing Resource Abuse

Hiding Network Information

Ghostor-MH

Verifiable Anonymous History

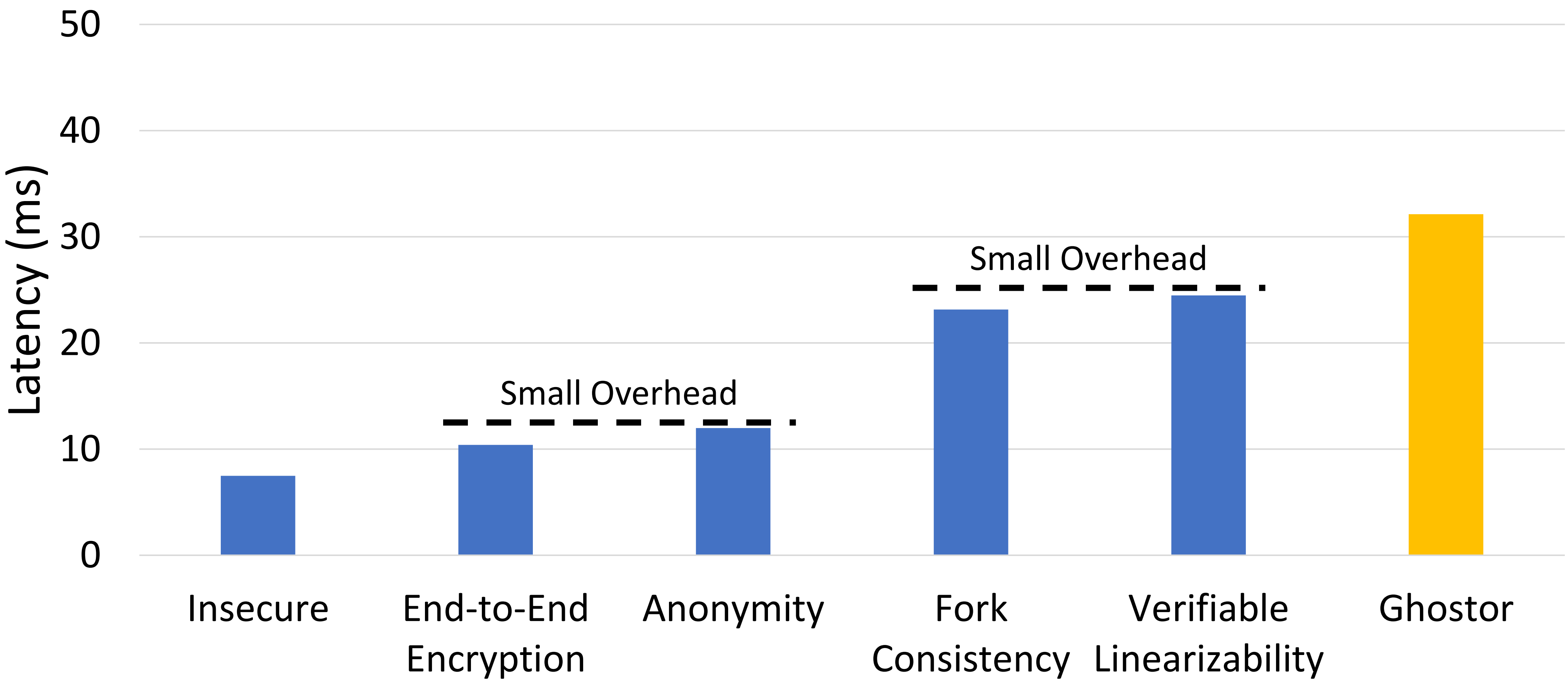
Anonymously Distributed Shared Capabilities

Implementation

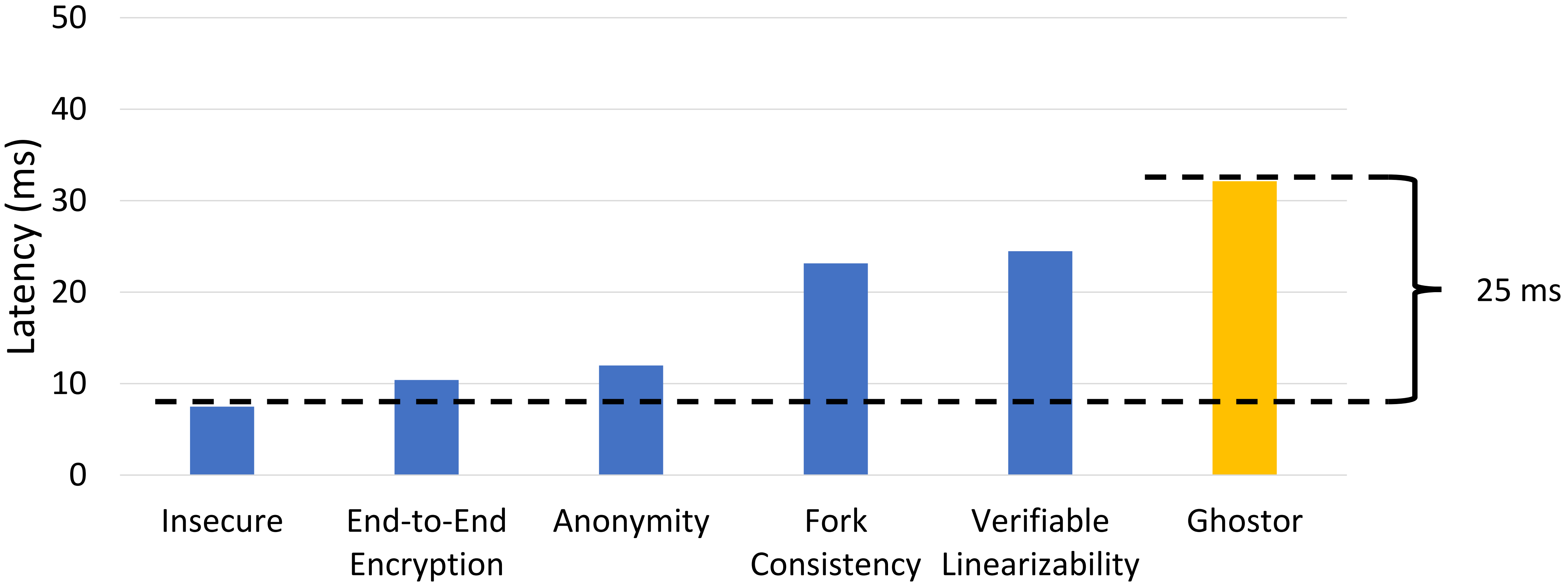
- Implemented Ghostor prototype in Go
- Built on top of Ceph RADOS
  - Linearizable, distributed, fault-tolerant object store

- Benchmarked on Amazon EC2 in multi-node, multi-SSD setup

Server-Side Latency to PUT a 1 MiB Object



Server-Side Latency to PUT a 1 MiB Object



Total Latency

- To hide network information, Ghostor clients use the Tor anonymity network to contact the server
- **With Tor, overall latency is several seconds**

Conclusion

Ghostor is a cryptographic data sharing system based on *decentralized trust*

It achieves:

- Anonymity: server cannot tell which user makes an access
- Verifiable Linearizability: users detect if they don't receive the latest data

Ghostor's techniques could significantly boost the security guarantees of:

WhatsApp Keybase tresorit CRYPHO virtru

Conclusion

Ghostor is a cryptographic data sharing system based on *decentralized trust*.

It achieves:

- Anonymity: server cannot tell which user makes an access
- Verifiable Linearizability: users detect if they don’t receive the latest data

Thank you!

This material is based on work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1752814. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.