SafeCloud

Storing Critical Data in the Cloud: Challenges and Solutions

Miguel Pupo Correia

ICSOFT 2018 – July 2018

U LISBOA | UNIVERSIDADE DE LISBOA

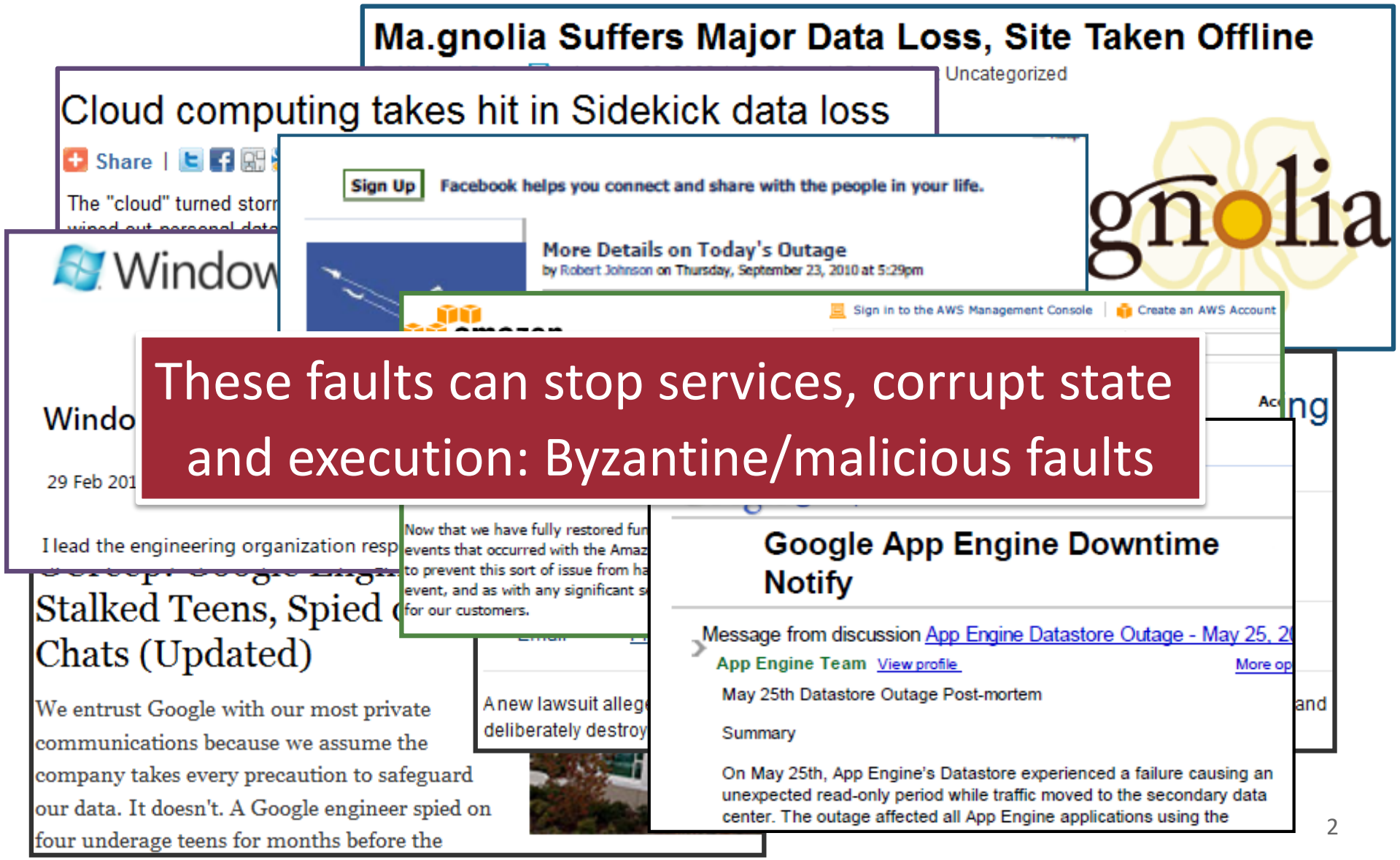
TÉCNICO LISBOA

European Commission | Horizon 2020 European Union funding for Research & Innovation

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

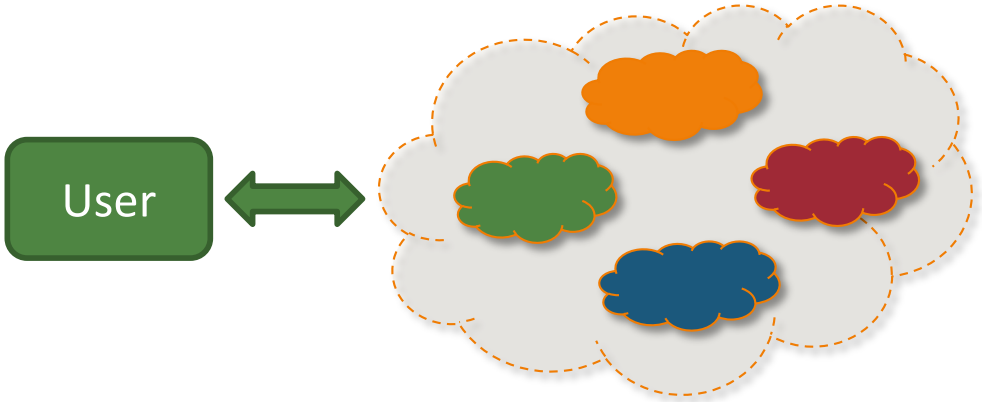
inesc id lisboa 10 anos

Clouds are complex so they fail



Cloud-of-Clouds

- Consumer runs service on a set of clouds forming a virtual cloud, what we call a cloud-of-clouds
- Related to the notion of federation of clouds
  - Federation of clouds – a virtual cloud created by cloud providers; requires cooperation between providers
  - Cloud-of-clouds – an ad-hoc virtual cloud created by consumers; no cooperation between clouds needed



Cloud-of-Clouds dependability+security

- There is redundancy and diversity between clouds
- so even if some clouds fail a cloud-of-clouds that implements replication can still guarantee:
  - Availability – if some stop, the others are still there
  - Integrity – if some corrupt data, data is still at the others
  - Disaster-tolerance – clouds can be geographically far
  - No vendor lock-in – several clouds anyway
- plus, although, not specific to cloud-of-clouds:
  - Confidentiality (from clouds) – encryption
  - Confidentiality/integrity (from users) – access control

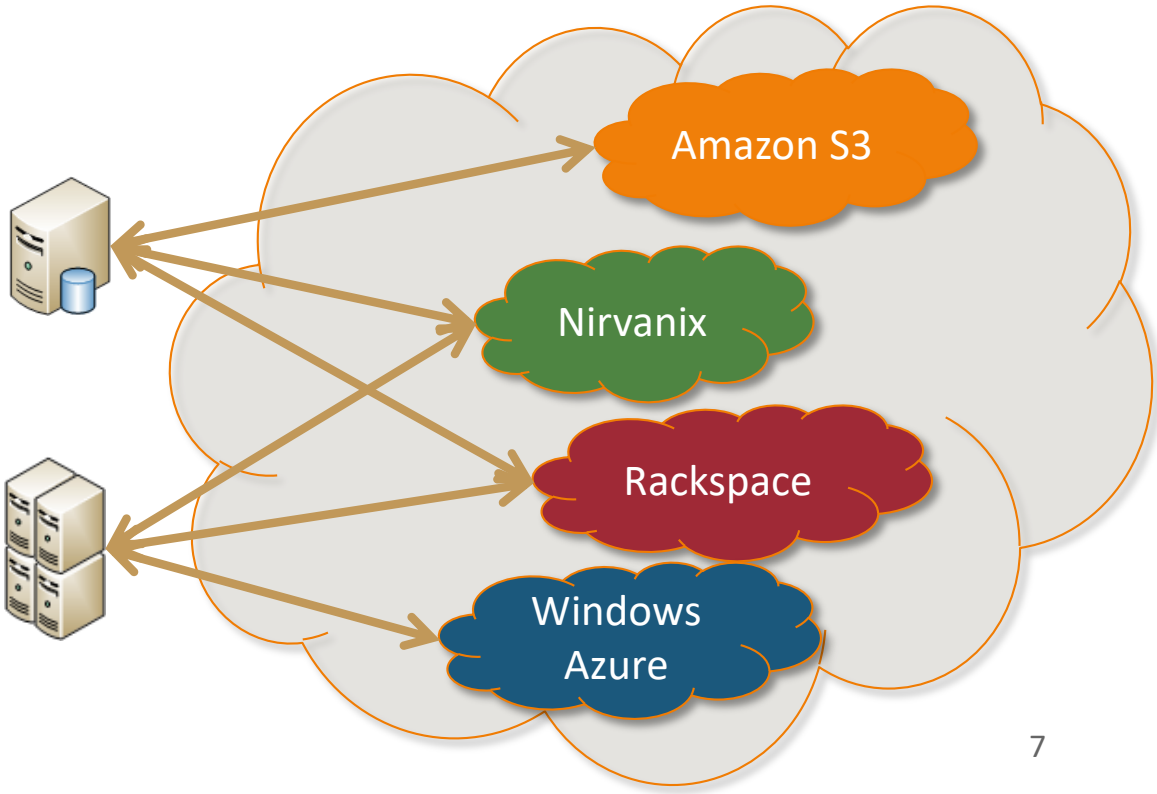
Outline

- DepSky – file storage in clouds-of-clouds
- SCFS – file system in clouds-of-clouds
- S-Audit – file integrity verifier
- SafeCloud-FS – file system in clouds-of-clouds

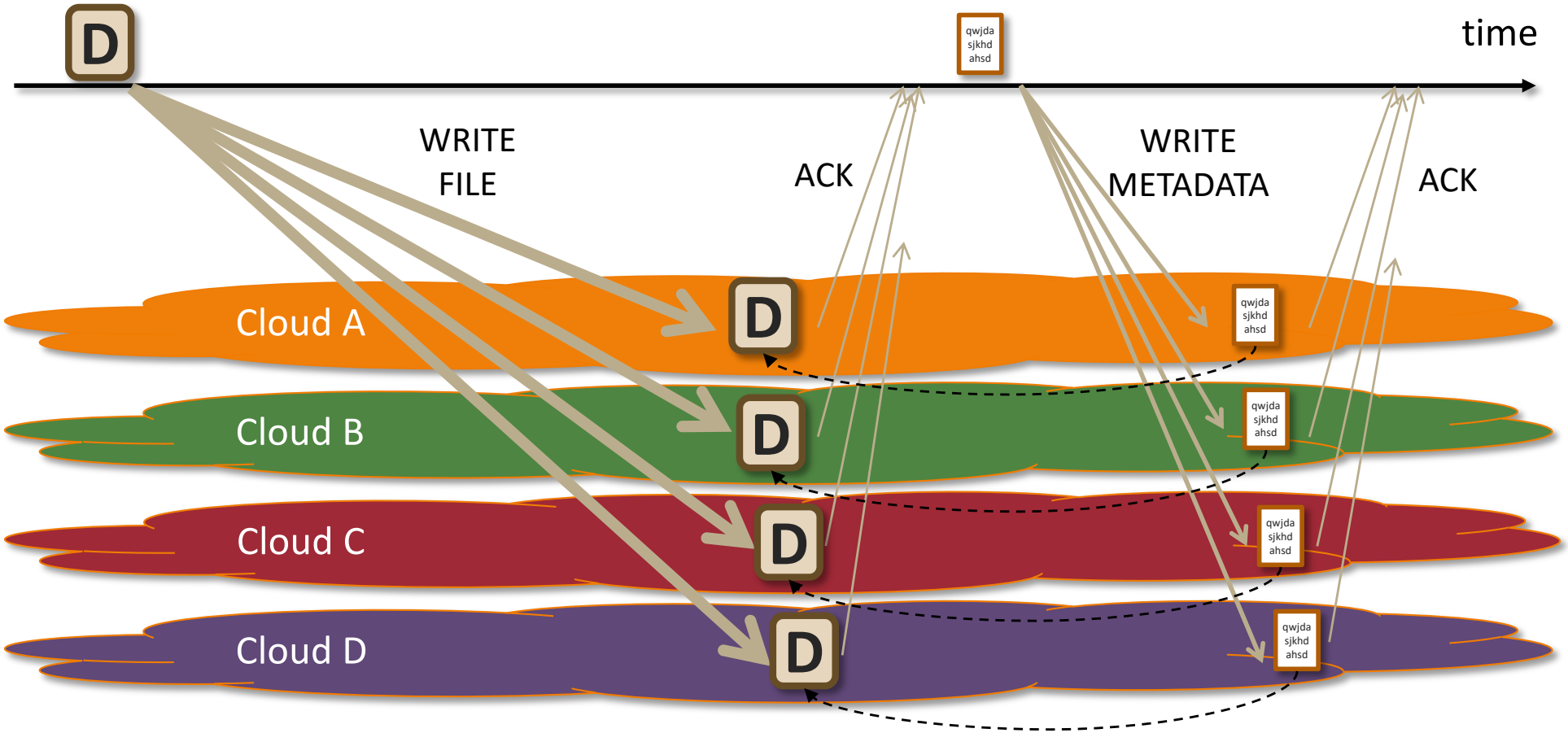
DEPSKY – FILE STORAGE IN CLOUDS-OF-CLOUDS

DepSky

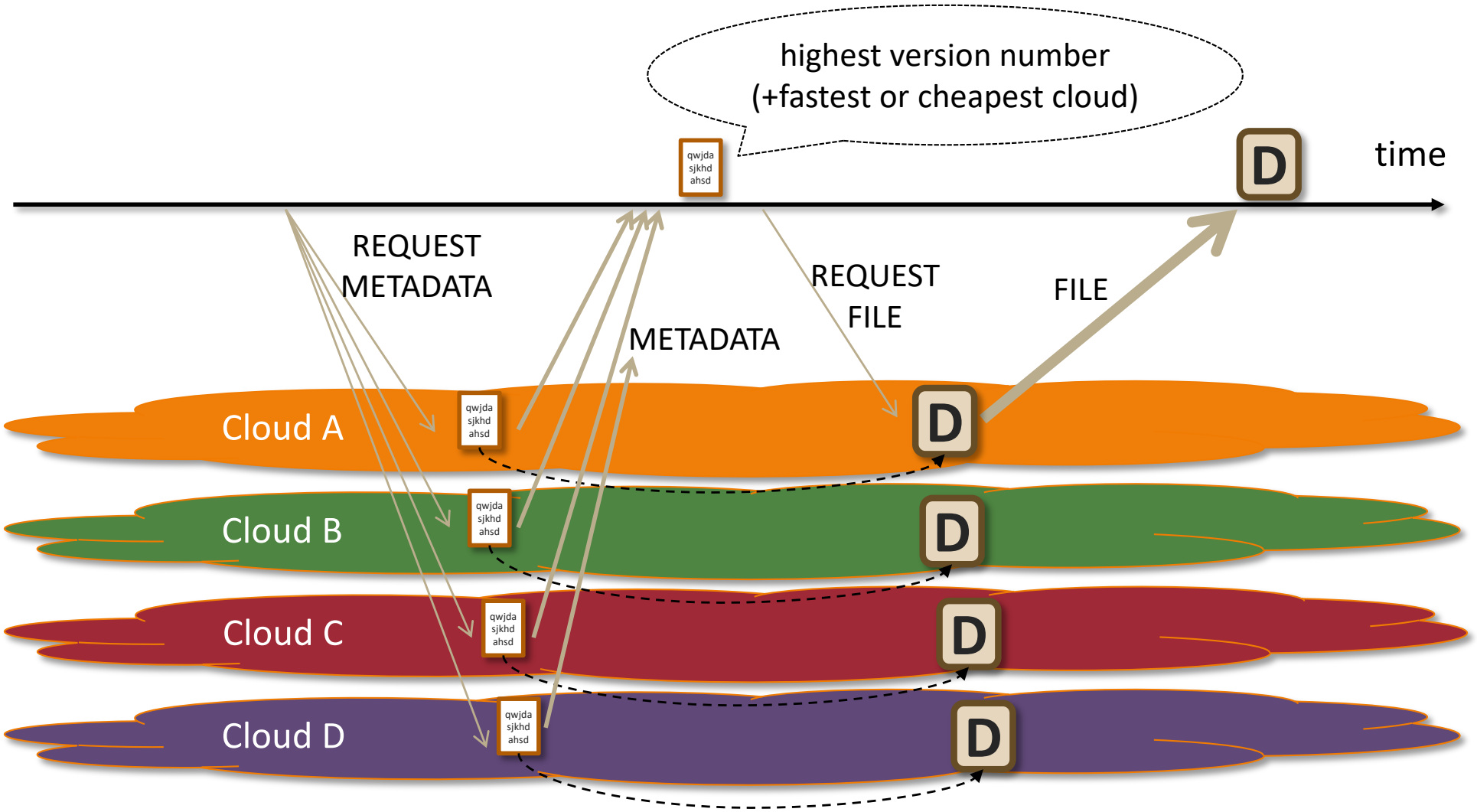
- Client-side library for cloud-of-clouds storage
  - File storage, similar to Amazon S3: read/write files, etc.
- Uses storage cloud services (S3, etc.) as they are:
  - All code at the client
- Data is updatable
  - Requires Byzantine quorum replication protocols for consistency



Write protocol



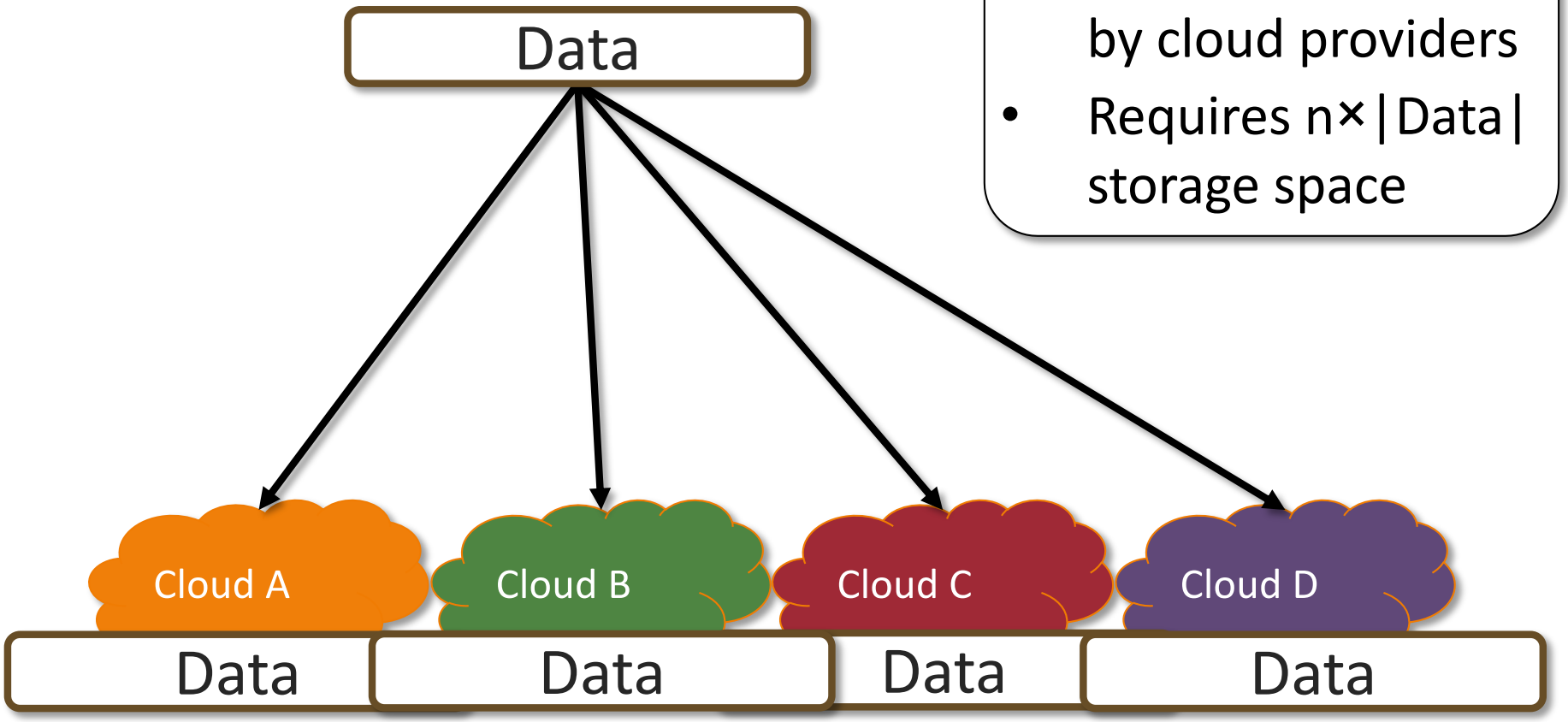
Read protocol



File is fetched from other clouds if signature doesn't match the file

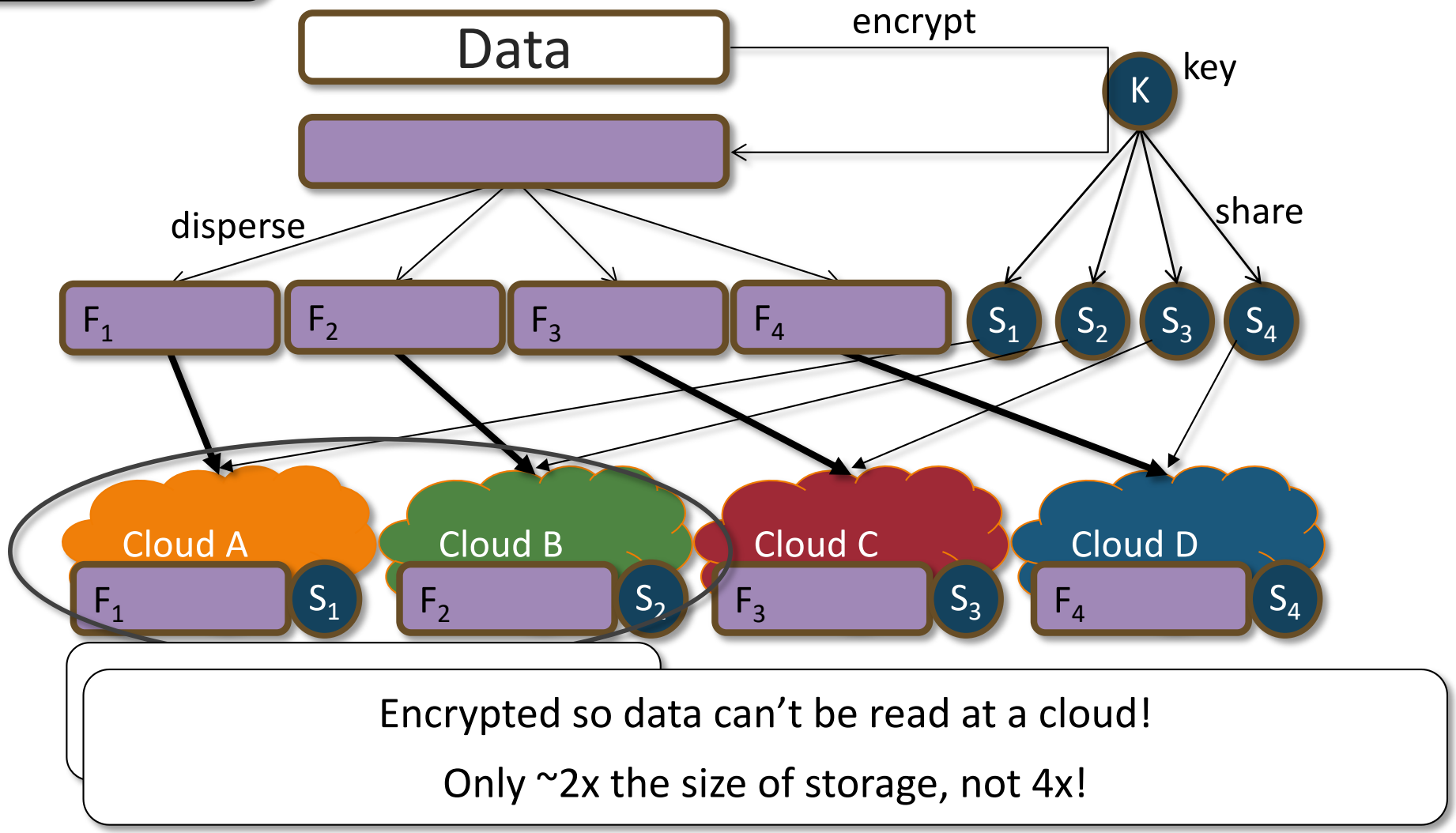
DepSky-A: limitations

- Data is accessible by cloud providers
- Requires n×|Data| storage space



DepSky-CA: combining erasure codes and secret sharing

Only for data, not metadata



Encrypted so data can't be read at a cloud!

Only ~2x the size of storage, not 4x!

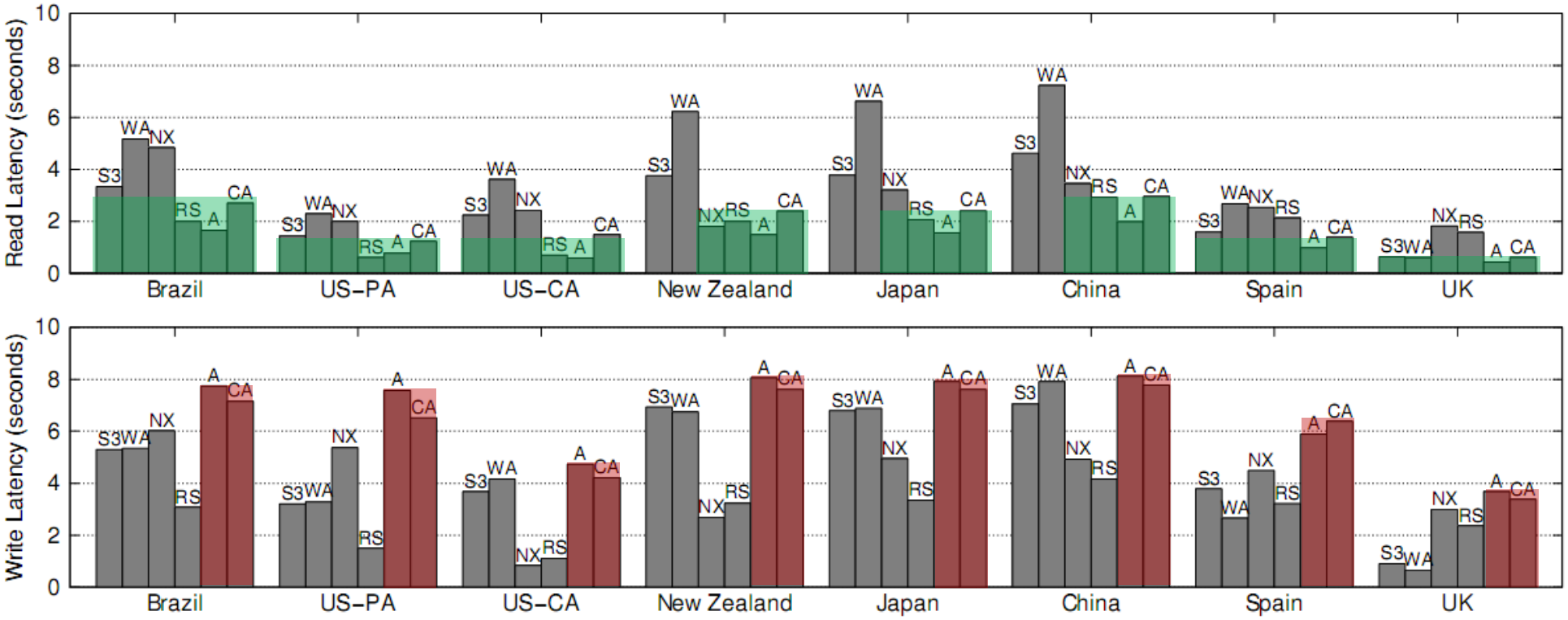
Consistency proportionality

- The consistency provided by DepSky is the same as the base storage clouds
  - If the weakest consistency cloud provides eventual consistency, DepSky provides eventual consistency
  - If the weakest consistency cloud provides regular storage, DepSky provides regular storage
  - …

DepSky latency

100KB files, clients in PlanetLab nodes

DepSky's **read** latency is close to the cloud with the **best** latency



DepSky's **write** latency is close to the cloud with the **worst** latency

DepSky perceived availability

- perceived availability = n. of files read / n. of read attempts
- impacted by the cloud and Internet availability

Location	Reads Tried	DepSky-A	DepSky-CA	Amazon S3	Rackspace	Azure	Nirvanix
Brazil	8428	1.0000	0.9998	1.0000	0.9997	0.9793	0.9986
US-PA	5113	1.0000	1.0000	0.9998	1.0000	1.0000	0.9880
US-CA	8084	1.0000	1.0000	0.9998	1.0000	1.0000	0.9996
New Zealand	8545	1.0000	1.0000	0.9998	1.0000	0.9542	0.9996
Japan	8392	1.0000	1.0000	0.9997	0.9998	0.9996	0.9997
China	8594	1.0000	1.0000	0.9997	1.0000	0.9994	1.0000
Spain	6550	1.0000	1.0000	1.0000	1.0000	0.9796	0.9995
UK	7069	1.0000	1.0000	0.9998	1.0000	1.0000	1.0000

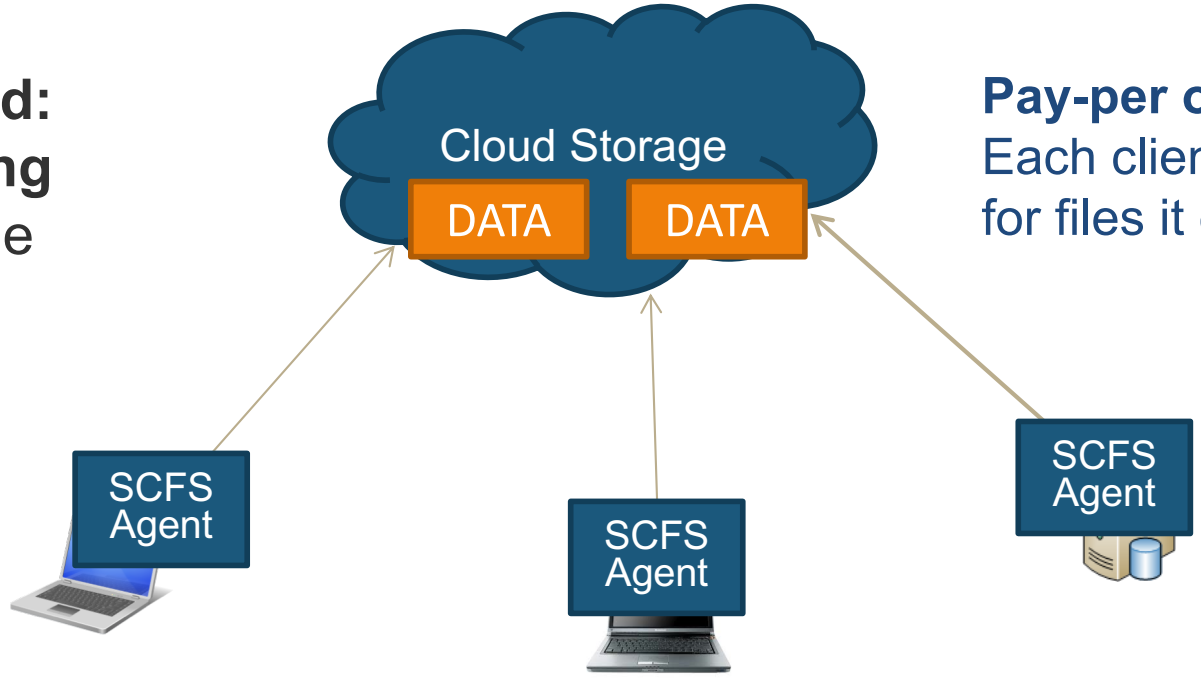
SCFS – FILE SYSTEM IN CLOUDS-OF-CLOUDS

Storage vs. File System (DepSky vs. SCFS)

- Storage (DepSky)
  - API: simple operations over data blocks
  - same consistency as clouds
  - **create**(id)
  - **read**(fd)
  - **write**(fd,block)
  - **delete**(fd)
  - **lock**(fd)
  - **unlock**(fd)
  - **setACL**(fd)
- File system (SCFS)
  - API: ~POSIX, so it's mounted and unmodified apps can use it (uses FUSE)
  - strong consistency
  - **open**(path,flags)
  - **read**(fd,buffer,length,offset)
  - **write**(fd,buffer,length,offset)
  - **chmod**(path,mode)
  - **mkdir**(path,mode)
  - **flush, fsync, link, rmdir, symlink, chown,...**

Shared Cloud-backed File System-SCFS

Client-based:
Uses **existing** cloud storage services



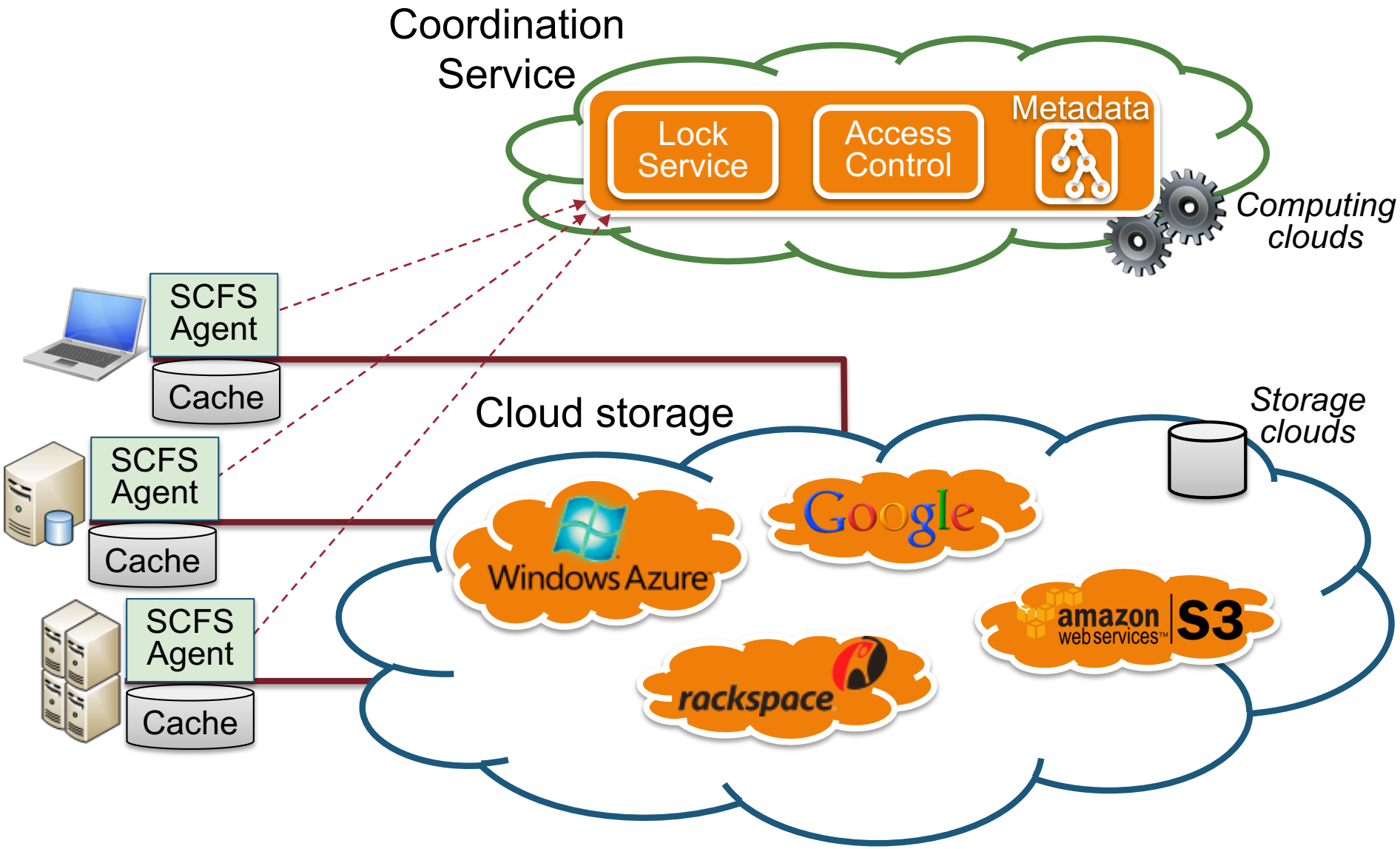
Pay-per ownership:
Each client **pays** for files it creates

Strong Consistency

Controlled sharing:
Access control for security and concurrency

Redundant Cloud Services

SCFS architecture



Features

- Data layout/access pattern
  - Each file is an object (single-block file)
  - Multiple versions of the files are maintained
  - Always write, avoid reading (exploiting free writes)
- Caching
  - File cache: persistent (to avoid reading)
    - Local storage is used to hold copies of all client files (that fit)
    - Opened files are also maintained in main-memory
  - Metadata cache: short-lived, main-memory
    - To deal with bursts of *metadata* requests

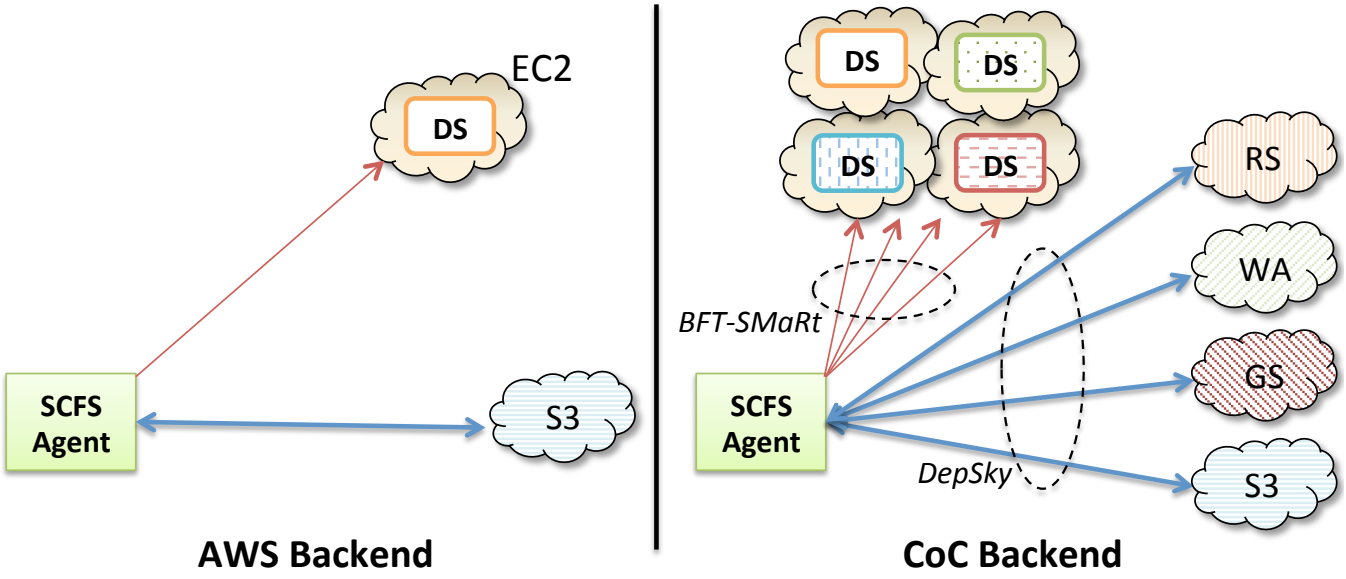
Features

- Consistency
  - Consistency-on-close semantics
    - when user closes a file, all updates he did become observable by the rest of the users
  - Locks to avoid write-write conflicts
- Modular coordination
  - Metadata is stored in a coordination service
    - e.g., Apache Zookeeper (crash fault-tolerant), our own DepSpace (Byzantine/intrusion-tolerant)
  - Also used for managing file locks
  - Separate data from metadata

SCFS configurations

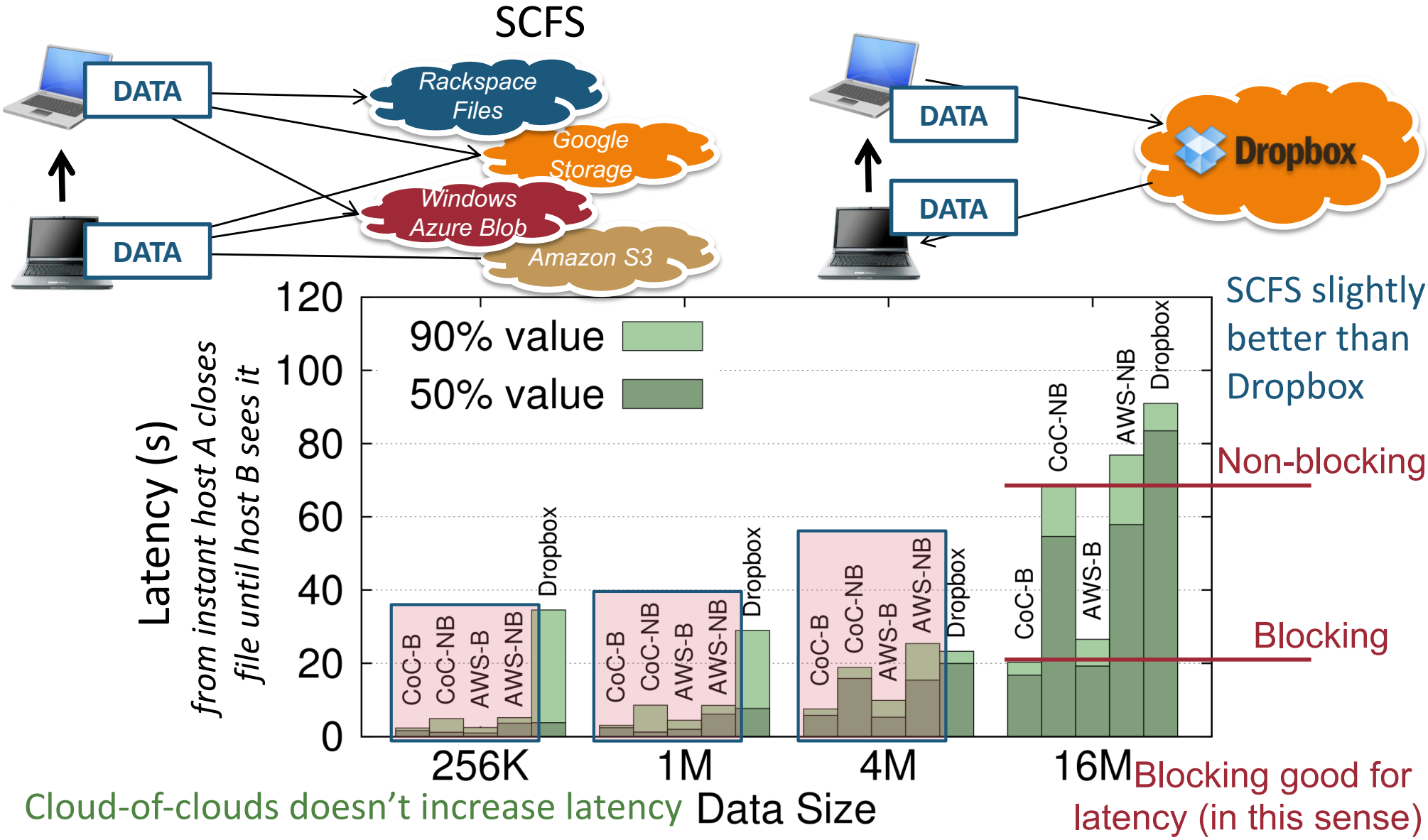
- SCFS can use different configurations/backends

Intrusion-tolerant configuration
(uses DepSky)



- Operation: **blocking**, non-blocking and non-sharing

Sharing latency: SCFS vs DropBox



Benchmarking unmodified desktop applications

1.2 MB file

OpenOffice Writer

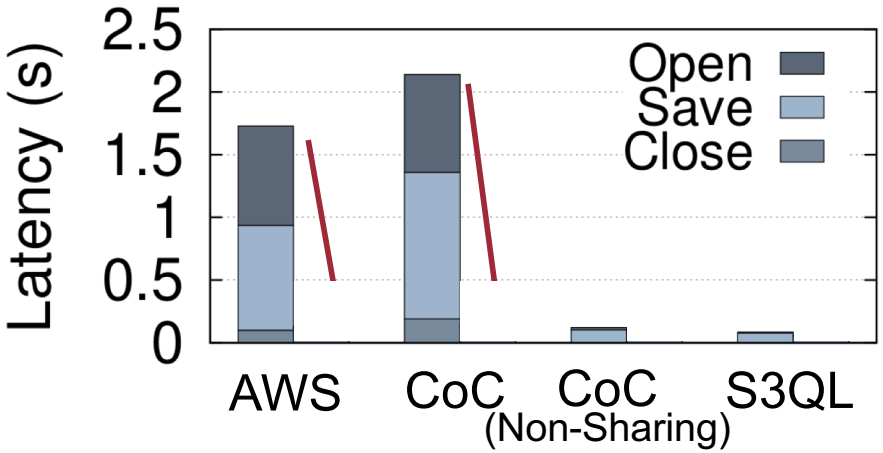
Open Action: 1 open(f,rw), 2 read(f), 3-5 open-write-close(lf1), 6-8 open-read-close(f), 9-11 open-read-close(lf1)

Save Action: 1-3 open-read-close(f), 4 close(f), 5-7 open-read-close(lf1), 8 delete(lf1), 9-11 open-write-close(lf2), 12-14 open-read-close(lf2), 15 truncate(f,0), 16-18 open-write-close(f), 19-21 open-fsync-close(f), 22-24 open-read-close(f), 25 open(f,rw)

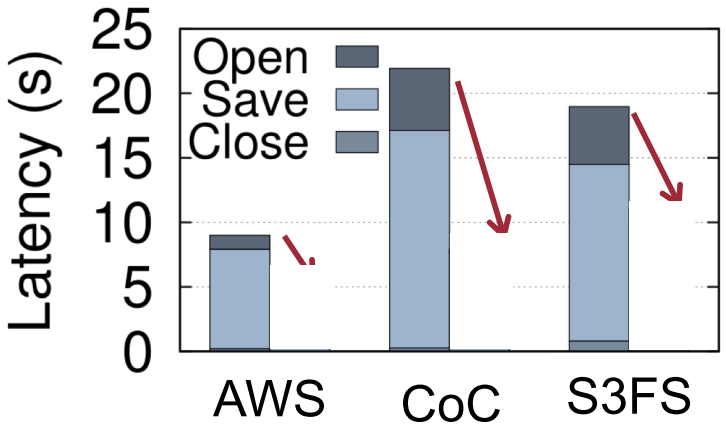
Close Action: 1 close(f), 2-4 open-read-close(lf2), 5 delete(lf2)

55% 40% 80% — lock file ops; may be done locally

Lots of operations; doing this remotely...



Non-blocking



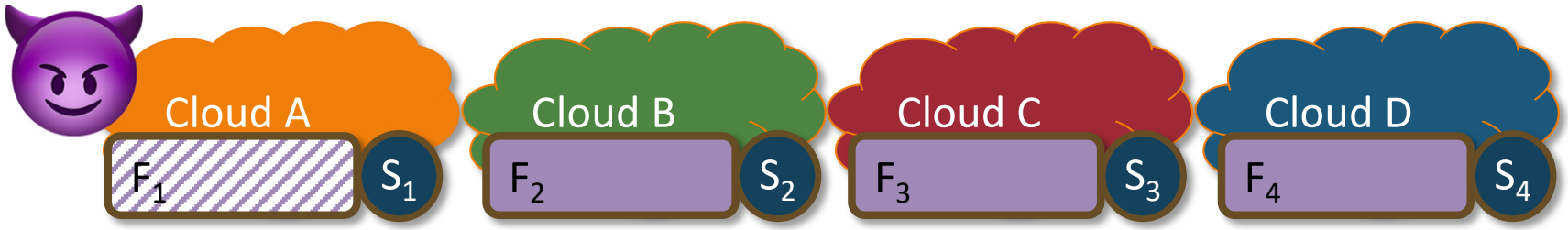
Blocking

Cloud-of-clouds per se doesn't increase latency much

Doing locks locally reduces much the latency

S-AUDIT – FILE INTEGRITY VERIFIER

Challenge: compromised cloud



- DepSky/SCFS: file compromise detected only they are downloaded → signatures don't match
- Is it ok to leave files unchecked for long periods?
- What is the cost of downloading all our files?

S-Audit

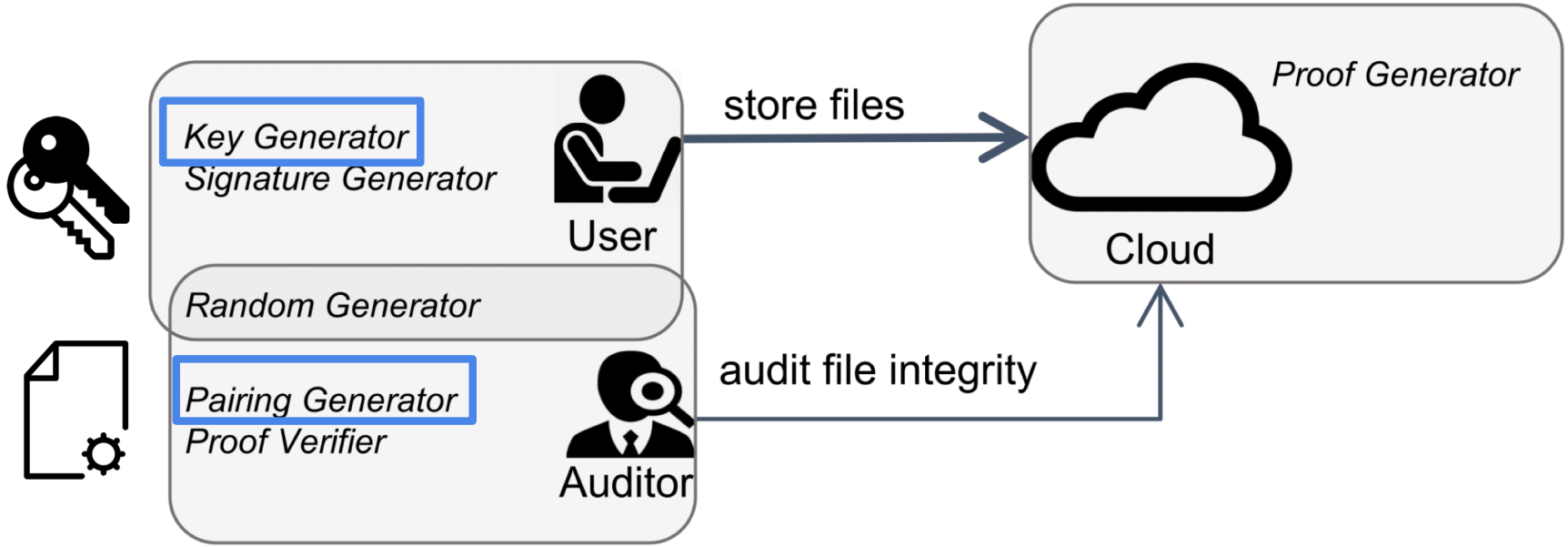
- Challenge: how to check integrity without adversary being able to provide a fake proof?
  - Signatures don't work: cloud might store only the signatures, not the files
- Solution: homomorphic digital signatures
  - Computed in runtime
  - Adversary can't generate them without the files

S-Audit

- First practical library to implement homomorphic digital signatures
  - Improves the Shacham Waters (SW) scheme
  - Smaller signatures by choosing a class of elliptic curves
- Actions: Setup, Sign, Verify

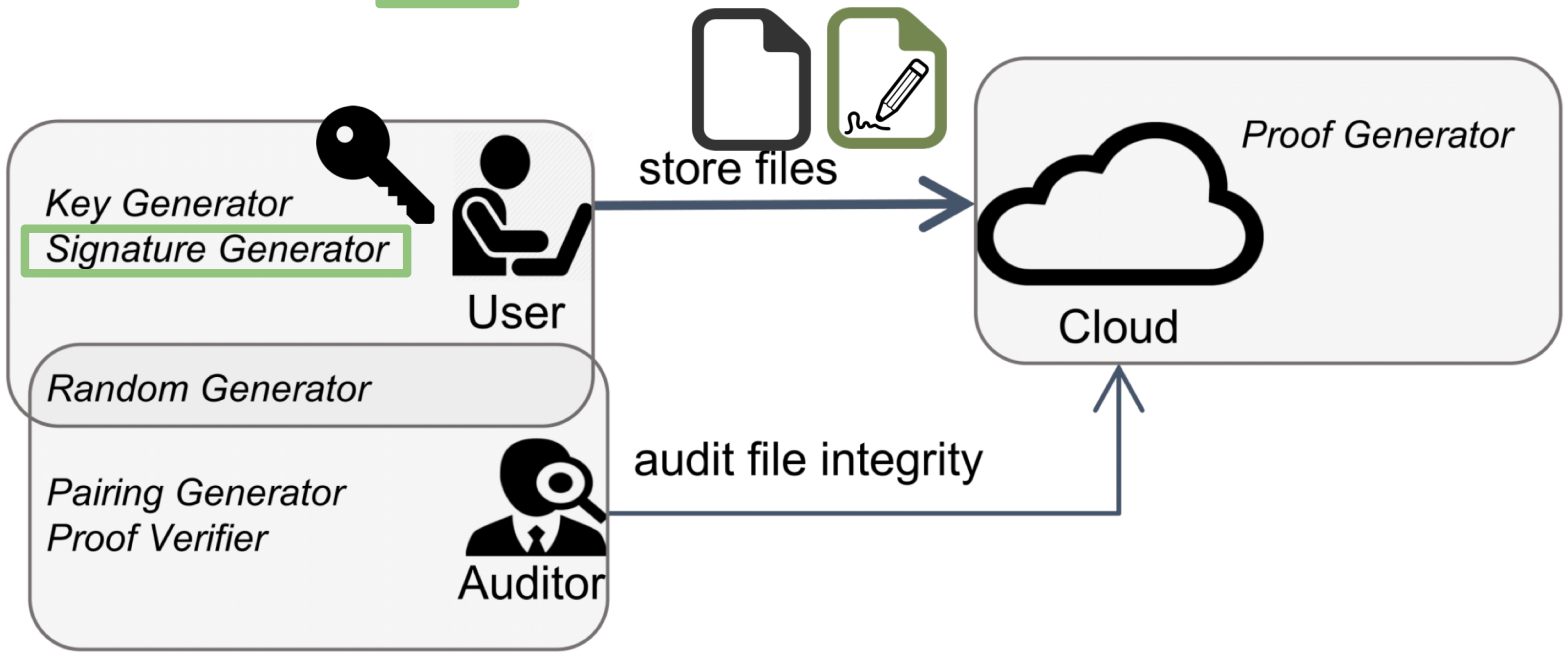
S-Audit

- First practical library to implement homomorphic digital signatures
  - Improves the Shacham Waters (SW) scheme
  - Smaller signatures by choosing a class of elliptic curves
- Actions: Setup, Sign, Verify



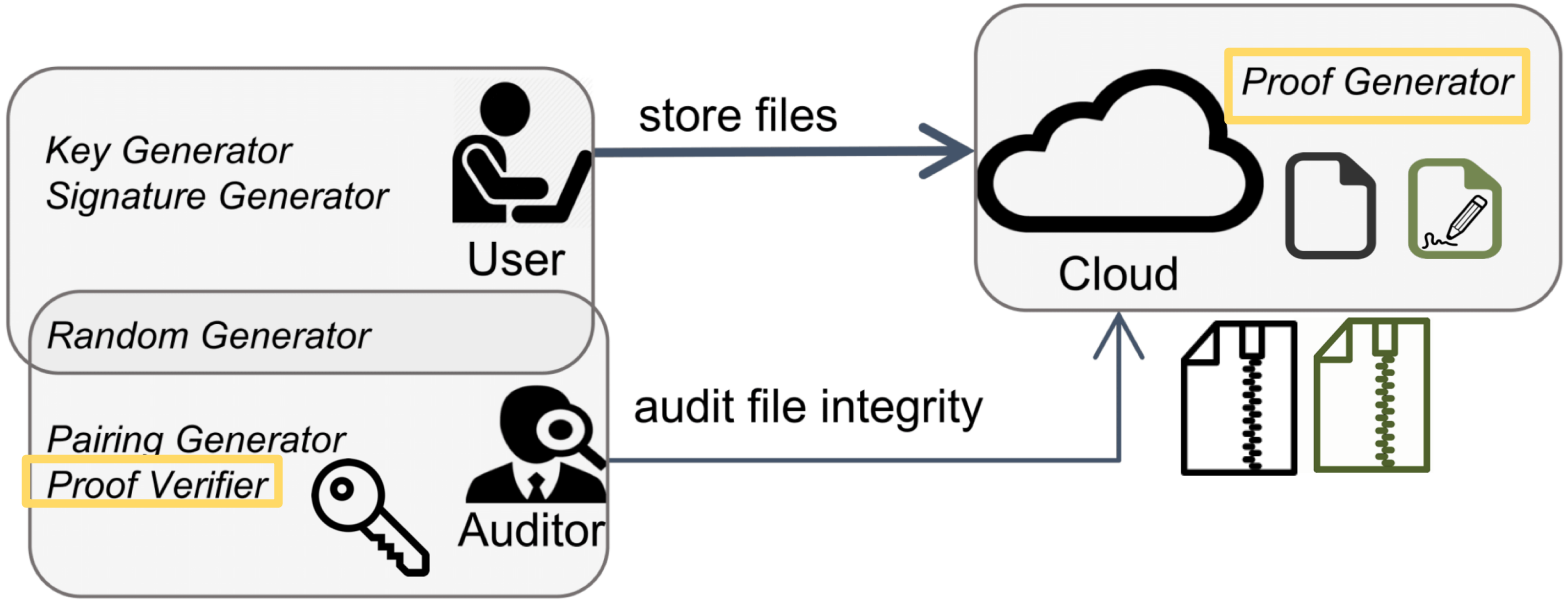
S-Audit

- First practical library to implement homomorphic digital signatures
  - Improves the Shacham Waters (SW) scheme
  - Smaller signatures by choosing a class of elliptic curves
- Actions: Setup, Sign, Verify



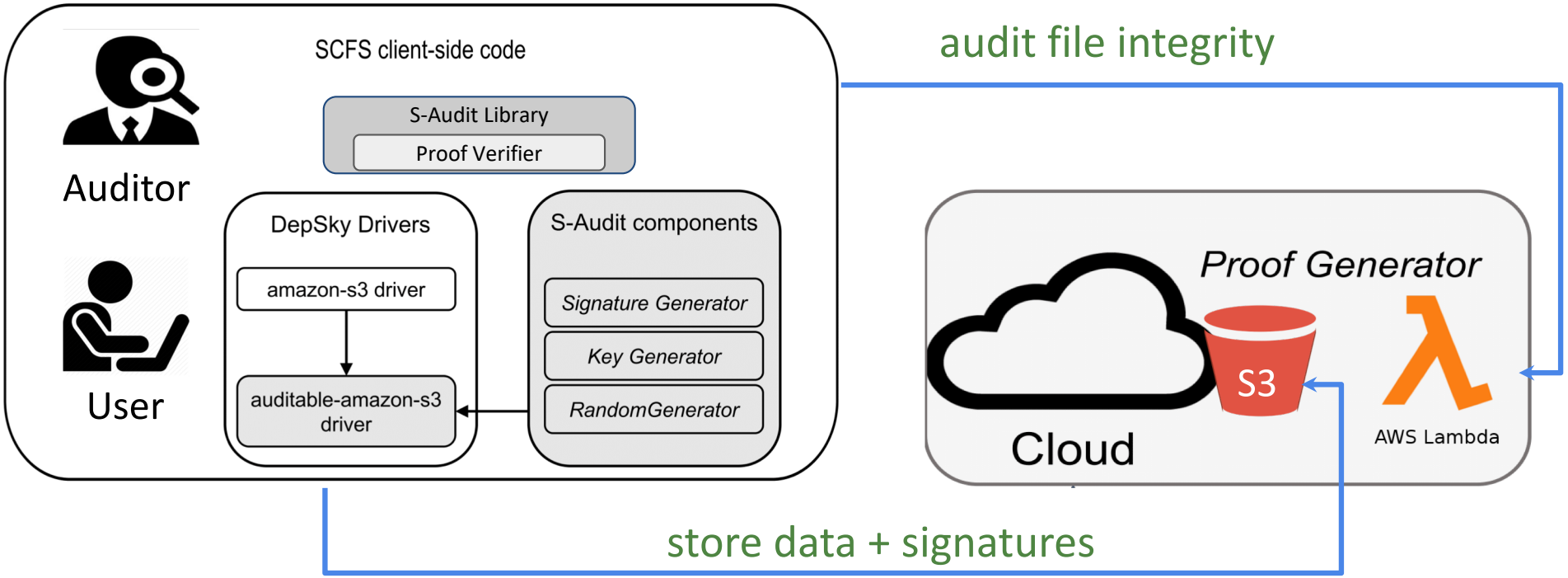
S-Audit

- First practical library to implement homomorphic digital signatures
  - Improves the Shacham Waters (SW) scheme
  - Smaller signatures by choosing a class of elliptic curves
- Actions: Setup, Sign, Verify



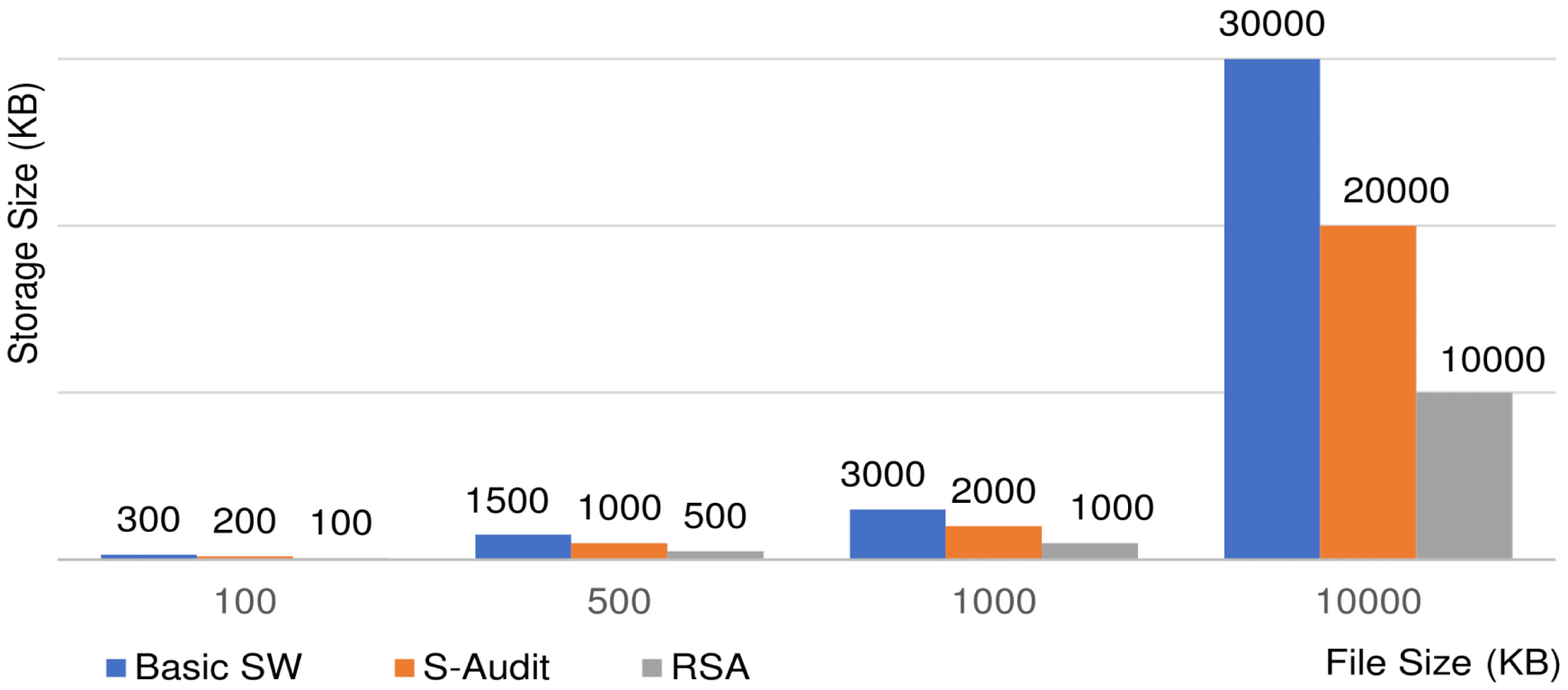
S-Audit implementation

- Integrated with Amazon AWS and SCFS with DepSky library



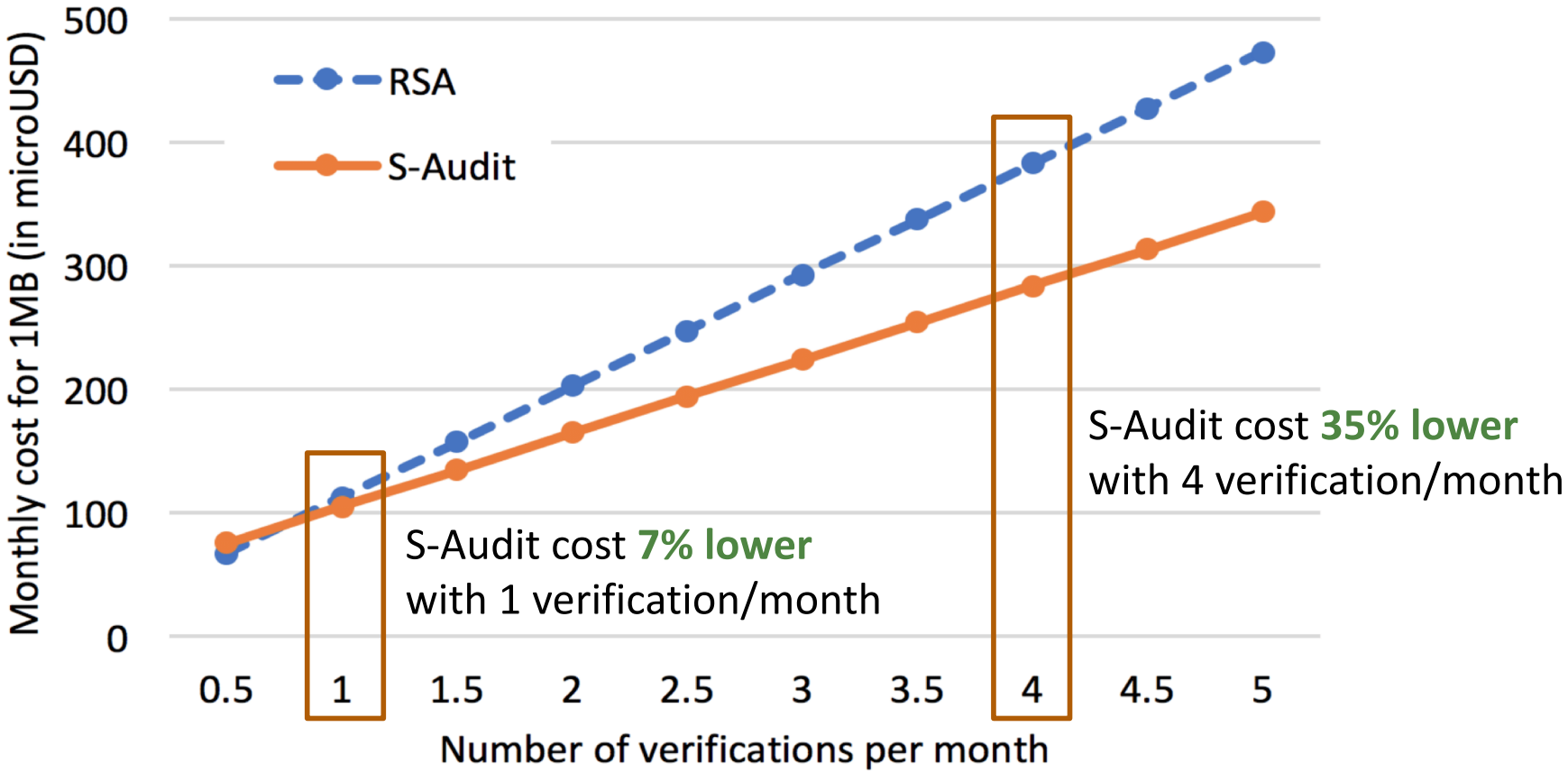
Storage costs

- What is the extra storage cost for storing a signature?
  - S-Audit: half than SW, but double than RSA signatures



Cost tradeoff

- Storage cost is double, verification cost is less ~30%, so benefit depends on number of verifications per month



SAFECLOUD-FS – AN ENHANCED CLOUD-OF-CLOUDS FILE SYSTEM

SafeCloud-FS

- New implementation of the SCFS architecture
- DepSky for cloud-of-clouds storage
- DepSpace coordination service
  - Although we started exploring HomomorphicSpace
- S-Audit for integrity verification
- Client-side security mechanisms
  - User credential protection
  - Intrusion recovery

WRAP-UP

Conclusions

- DepSky: storage clouds-of-clouds
  - Availability, integrity, disaster-tolerance, no vendor lock-in, confidentiality
  - Faults in clouds + versions, so Byzantine quorum system protocols
  - Same consistency as the storage clouds
  - Erasure codes to reduce the size of data stored
  - Secret sharing to store cryptographic keys in clouds

Conclusions

- SCFS: a cloud-backed file system
  - Similar guarantees to DepSky but near-POSIX API
  - Strong consistency provided by coordination service
  - Caching and careful design allows good performance
- S-Audit: file integrity verification
  - Uses an homomorphic digital signature scheme
- SafeCloud-FS: an enhanced cloud-backed file system

SafeCloud

Thank you

- Papers:
  - **DepSky: Dependable and Secure Storage in a Cloud-of-Clouds**. EuroSys 2010 / ACM Transactions on Storage, 2013
  - **SCFS: a Shared Cloud-backed File System**. Usenix Annual Technical Conference, 2014
  - **S-Audit: Efficient Data Integrity Verification for Cloud Storage**. IEEE TrustCom 2018
- Code:
  - https://www.safecloud-project.eu/results/platform/ss3 (new version in a few days)

TÉCNICO LISBOA

European Commission | Horizon 2020 European Union funding for Research & Innovation

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

inesc id lisboa 10 anos