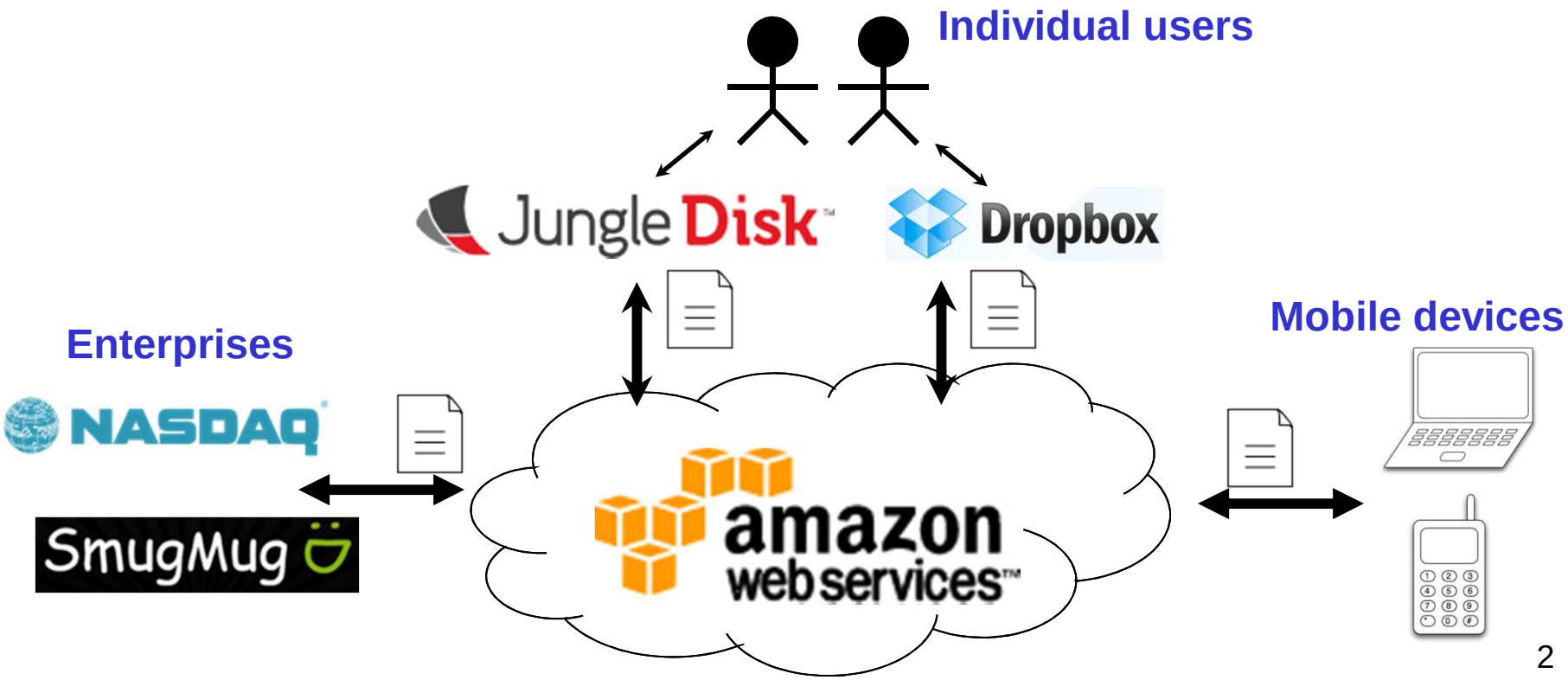
FADE: A Secure Overlay Cloud Storage System with Access Control and Assured Deletion

Patrick P. C. Lee

Cloud Storage is Emerging

- Cloud storage is now an emerging business model for data outsourcing



Case Studies

- Smugmug: hosting terabytes of photos since 2006
  - Savings: USD 500K per year as in 2006
  - More savings are expected with more photos
- NASDAQ: hosting historical market data since 2008
- More clients are found on: http://aws.amazon.com/solutions/case-studies/

References:
•http://don.blogs.smugmug.com/2006/11/10/amazon-s3-show-me-the-money/
•http://www.infoq.com/articles/nasdaq-case-study-air-and-s3?

Implications of Cloud Storage

- Cloud storage will be a cost-saving business solution:
  - Save cost for unused storage
  - Save technical support for data backups
  - Save electric power and maintenance costs for data centers
- Yet, as a cloud client, how do we provide security guarantees for our outsourced data?

Security Challenges

- Can we protect outsourced data from improperly accessed?
  - Unauthorized users must not access our data
  - We don’t want cloud providers to mine our data for their marketing purposes
- We need **access control**:
  - Only authorized parties can access outsourced data

Security Challenges

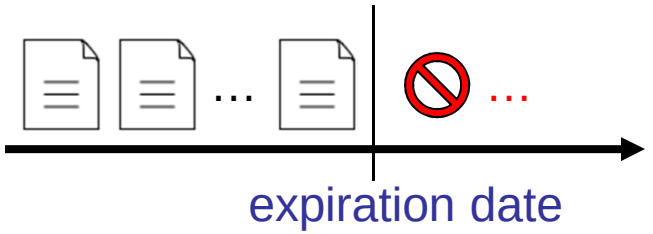
- Can we reliably remove data from cloud?
  - We don't want backups to exist after pre-defined time
    - e.g., to avoid future exposure due to data breach or error management of operators
  - If an employee quits, we want to remove his/her data
    - e.g., to avoid legal liability
- Cloud makes backup copies. We don't know if all backup copies are reliably removed.
- We need **assured deletion**:
  - Data becomes inaccessible upon requests of deletion

Previous Work

- Cryptographic protection on outsourced data storage
  [Ateniese et al., SecureComm'08; Wang et al., CCSW'09]
  - Require new protocol support on the cloud infrastructure
- Security solutions compatible with existing cloud (e.g., Cumulus, JungleDisk)
  [Yun et al., CCSW'09; Vrable et al., ToS'09]
  - No guarantees of reliable deletion of data

Previous Work

- Perlman's Ephemerizer [NDSS'07]



  - A file is encrypted with a data key
  - The data key is further encrypted with a time-based control key
  - The control key is deleted when expiration time is reached
  - The control key is maintained by a separate key manager (aka Ephemerizer)
- Weaknesses:
  - Target only time-based assured deletion
    - No fine-grained control of different file access policies
  - No implementation

Previous Work

- Vanish [USENIX'09]
  - Divide the data key into many key shares
  - Store key shares in nodes of a deployed P2P network
  - Nodes remove key shares that reside in cache for 8 hours
- Weaknesses:
  - Time-based, no fine-grained control

Our Work

FADE: a secure overlay cloud storage system with **f**ile **a**ssured **de**letion

- Design feature of FADE:
  - work atop today's cloud as an overlay
- Security features of FADE:
  - Data confidentiality and integrity
  - Fine-grained access control: files are accessible only when authorized
  - Fine-grained file assured deletion: files are permanently inaccessible and unrecoverable based on policies

Yang Tang, Patrick P. C. Lee, John C. S. Lui, Radia Perlman,
"Secure Overlay Cloud Storage with File Assured Deletion", SecureComm 2010.

Our Work

- We propose a new policy-based file assured deletion scheme that reliably deletes files of revoked file access policies
  - A generalized version of time-based delete

- We implement and evaluate a working prototype of FADE atop Amazon S3
  - FADE respects REST interface for cloud
  - FADE works feasibly in practice

Scenarios: Defining Policies

- Scenario 1: storing files for permanent employees
  - For each employee (e.g., Alice), define a user-based policy

P: Alice is an employee

User-based policy

  - If Alice quits her job, the key manager will remove the control key of policy P

Scenarios: Defining Policies

- Scenario 2: storing files for contract-based employees
  - e.g., Bob's contract expires on 2010-01-01. Define two policies

P_1: *Bob is an employee*	P_2: *valid before 2010-01-01*
User-based policy	Time-based policy

  - Files of Bob are associated with policy combination $P_1 \wedge P_2$

Scenarios: Defining Policies

- Scenario 3: storing files for a team of N employees
  - Each employee *i* is assigned a policy combination $P_{i1} \wedge P_{i2}$
    - P_{i1} = policy for employment status
    - P_{i2} = policy for valid time for access
  - Associate team's files with disjunctive combination

$$(P_{11} \wedge P_{12}) \vee (P_{21} \wedge P_{22}) \vee \ldots \vee (P_{N1} \wedge P_{N2})$$

Scenarios: Defining Policies

- Scenario 4: switching a cloud provider
  - Define a customer-based policy

P: customer of cloud provider X

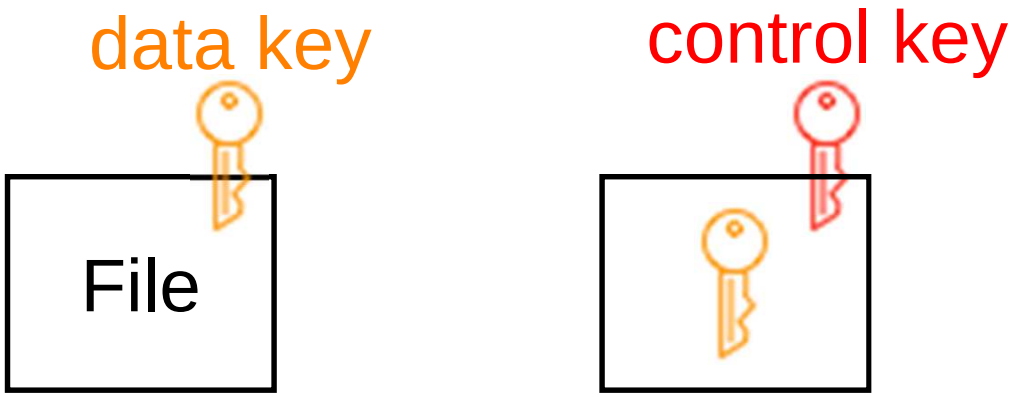
  - All files outsourced on X are tied with policy P
  - If the company switches to a new cloud provider, it simply revokes policy P

Policy-based File Assured Deletion

- Each file is associated with a data key and a file access policy
- Each policy is associated with a control key
- All control keys are maintained by a key manager
- When a policy is revoked, its respective control key will be removed from the key manager

Policy-based File Assured Deletion

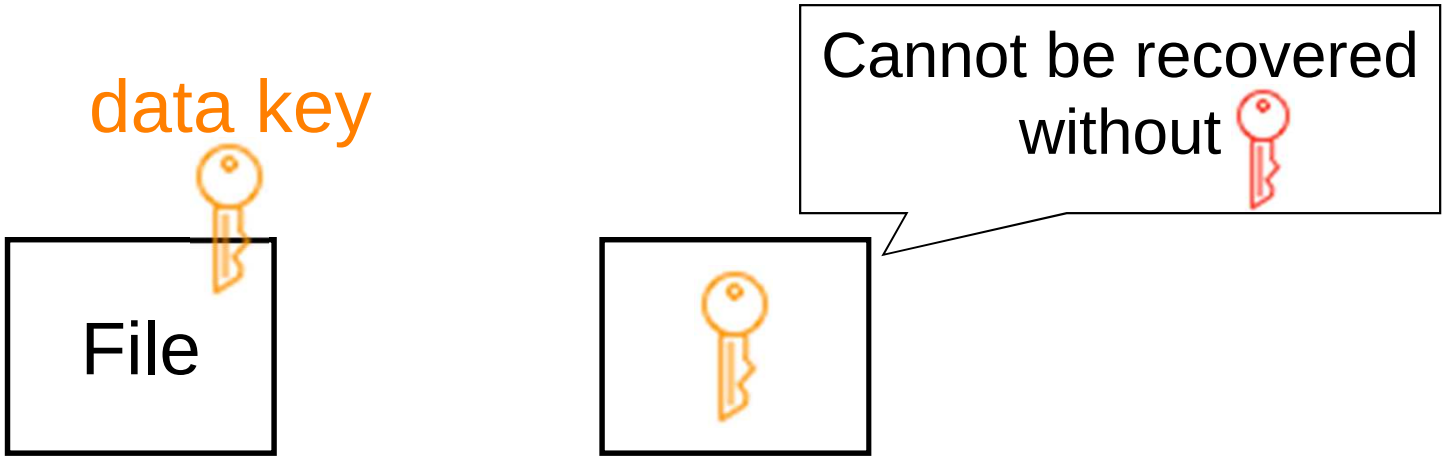
- Main idea:
  - File protected with data key
  - Data key protected with control key



 is maintained by the key manager

Policy-based File Assured Deletion

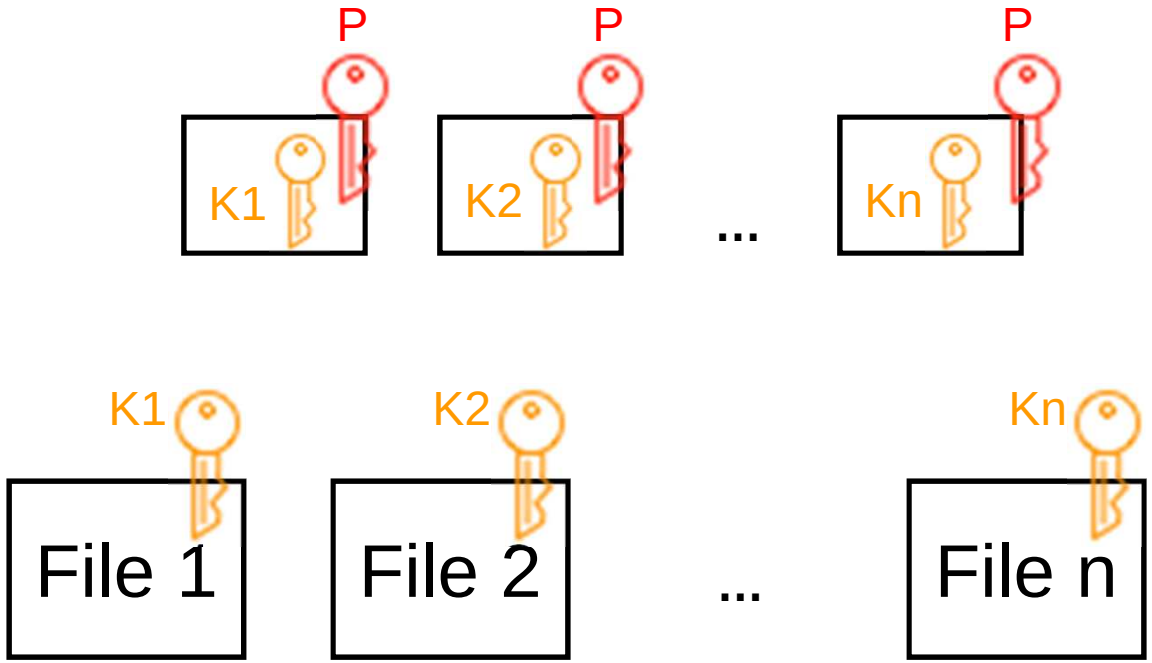
- When a policy is revoked, the control key is removed. The encrypted data key and hence the encrypted file cannot be recovered



- The file is deleted, i.e., even a copy exists, it is encrypted and **inaccessible by everyone**

Policy-based File Assured Deletion

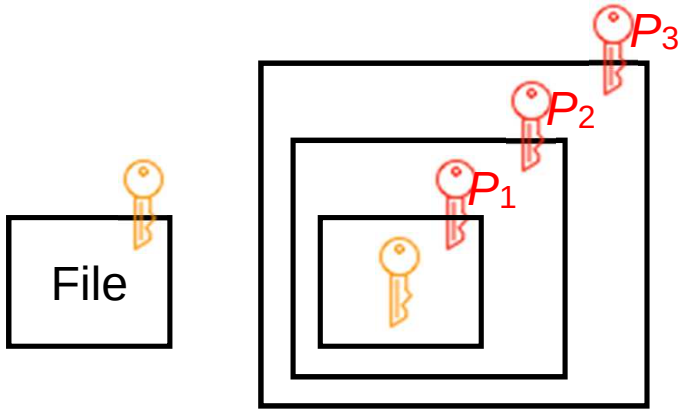
- *One policy associated with many files*



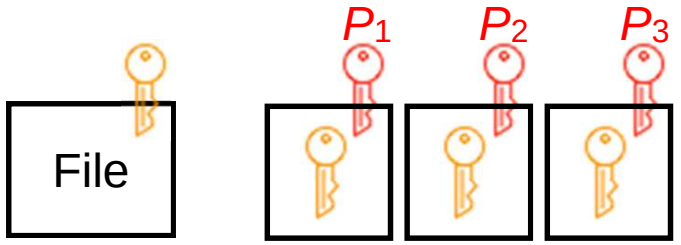
Policy-based File Assured Deletion

➢ *One file associated with multiple policies*

➢ Conjunctive policies
  - Satisfy all policies to recover file



➢ Disjunctive policies
  - Satisfy only one policy to recover file



How to Delete Data in Practice?

- With FADE, we delete data in two steps:
  - Normal deletion:
    - Request the cloud to delete a data copy via a regular DELETE command
    - Deregister the cloud copy to save storage cost
  - Assured deletion:
    - Delete the associated control key
    - Guarantee that all backup copies are unrecoverable
- In summary, we extend existing DELETE operation with assured deletion

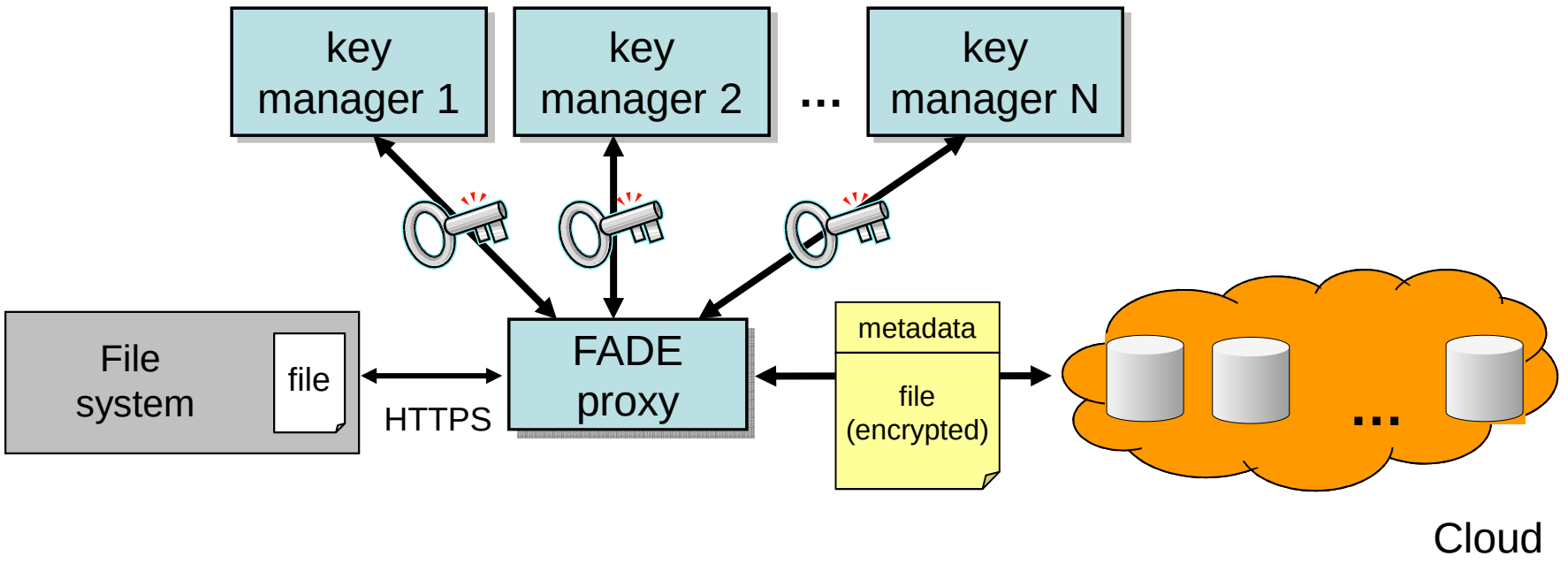
Lessons Learned

- Policy-based file-assured deletion allows a fine-grained control of how to delete files
- Similar to Attribute-Based Encryption (ABE)
  - ABE focuses on accessing data and distribute keys to users that satisfy attributes (policies)
  - We focus on deleting data, and need to manage/delete keys in a centralized manner

System Entities

- Data owner: the entity that originates data to be stored on cloud
- Key manager: maintains policy-based control keys for encrypting data keys
- Cloud: third-party cloud provider (e.g., Amazon S3) that stores data

Architecture of FADE



- FADE decouples key management and data management
- Key manager can be flexibly deployed in another trusted third party, or deployed within data owner
- No implementation changes on cloud

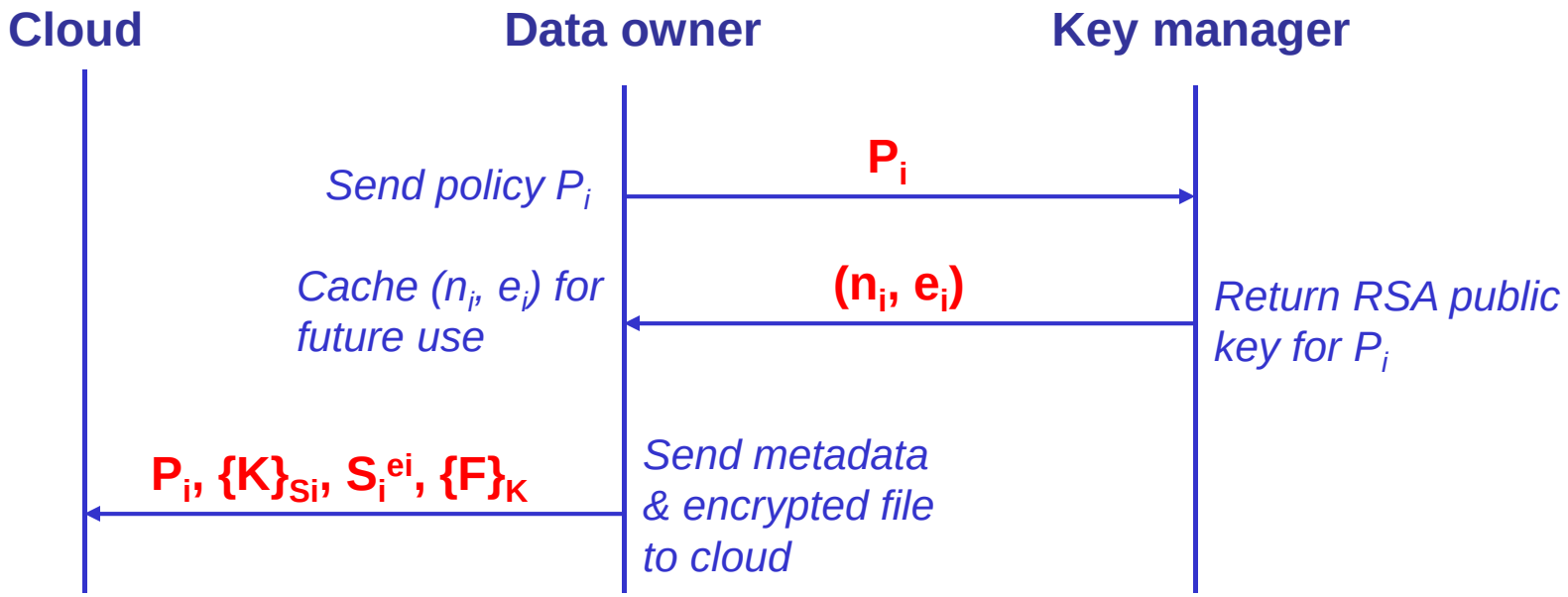
Threat Models and Assumptions

- File assured deletion is achieved
  - If we request to delete a file, it is inaccessible
- Key manager is minimally trusted
  - can reliably remove keys of revoked policies
  - can be compromised, but only files with active policies can be recovered
  - only knows the control keys, but should never know the data key used to encrypt a file
- Data owner forms an authenticated channel with key manager for key management operations

Key Management Operations

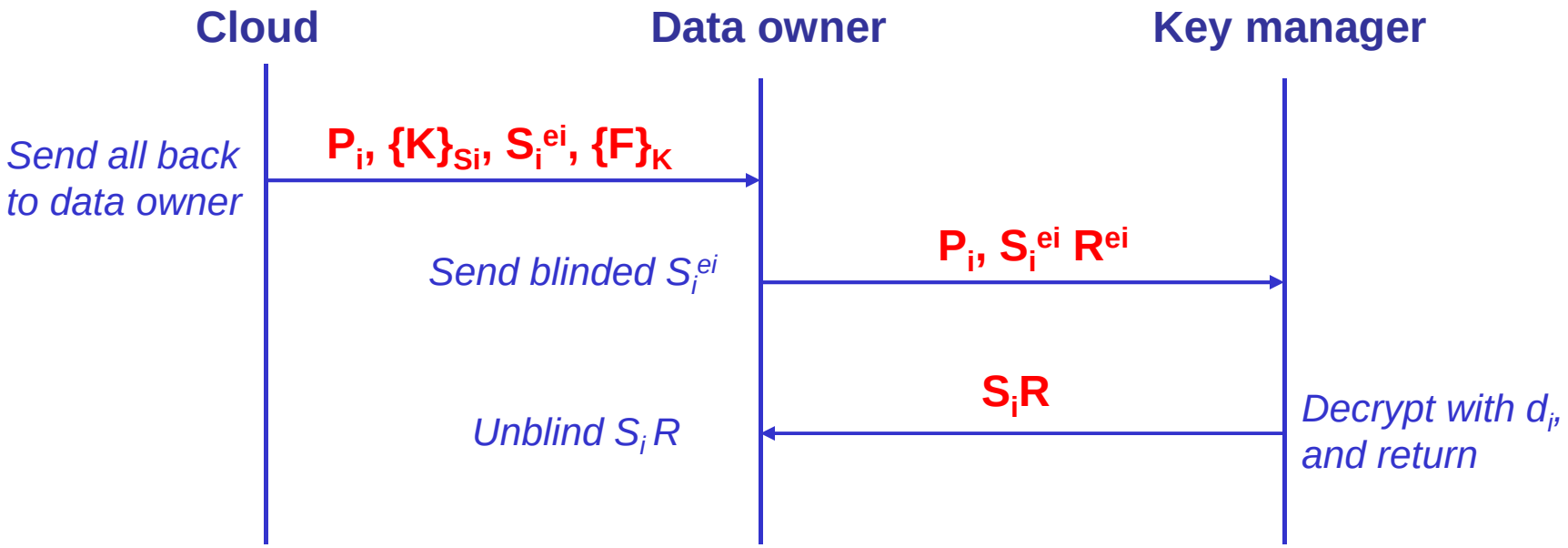
- Idea: use key management operations to decide how files are accessed while achieving file assured deletion
- Operations:
  - File upload
  - File download
  - Policy revocation
  - Policy renewal
- Built on blinded RSA

File Upload



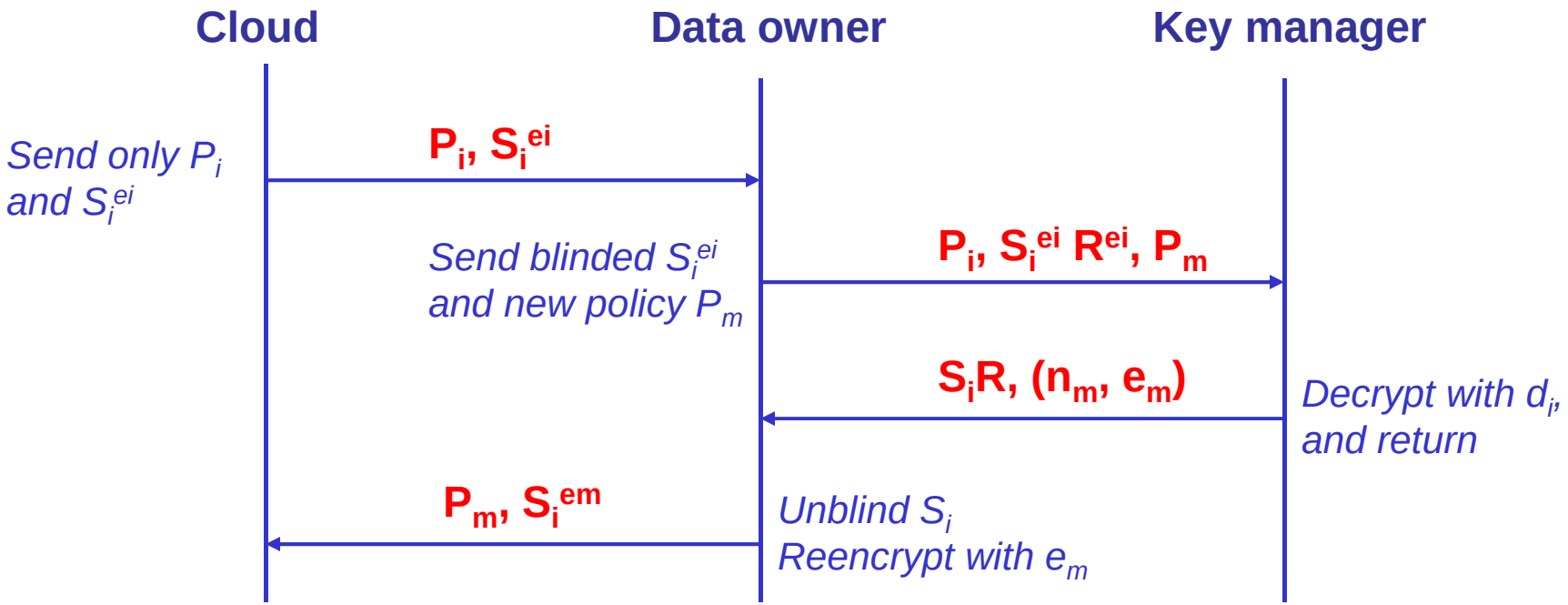
- Data owner randomly chooses (i) K for file F and (ii) S_i for policy P_i.
- Things sent to cloud
  - P_i = policy P_i
  - $\{K\}_{Si}$ = data key K encrypted with S_i using symmetric key crypto
  - S_i^{ej} = secret key S_i encrypted with e_i using public key crypto
    - S_i is used for policy renewal
  - $\{F\}_K$ = file encrypted with data key K using symmetric key crypto

File Download



- Data owner randomly picks a number R, and blinds S_i^{ei} with R^{ei}
- It unblinds $S_i R$, and recovers K and F

Policy Renewal



- Main idea: S_i re-encrypted into S_i^{em}
- $\{K\}_{Si}$ and $\{F\}_K$ remain unchanged on cloud

Policy Revocation

- Revoke policy P_i
  - Key manager removes all keys (n_i, e_i, d_i)
  - All files tied with policy P_i become inaccessible

FADE Implementation

- Use Amazon S3 as our backend (but can use other clouds)
- Use C++ with OpenSSL and libAWS++
- Each file has its own metadata:
  - File metadata: file size and HMAC
  - Policy metadata: policy information and encrypted keys

Interfaces of Data Owner

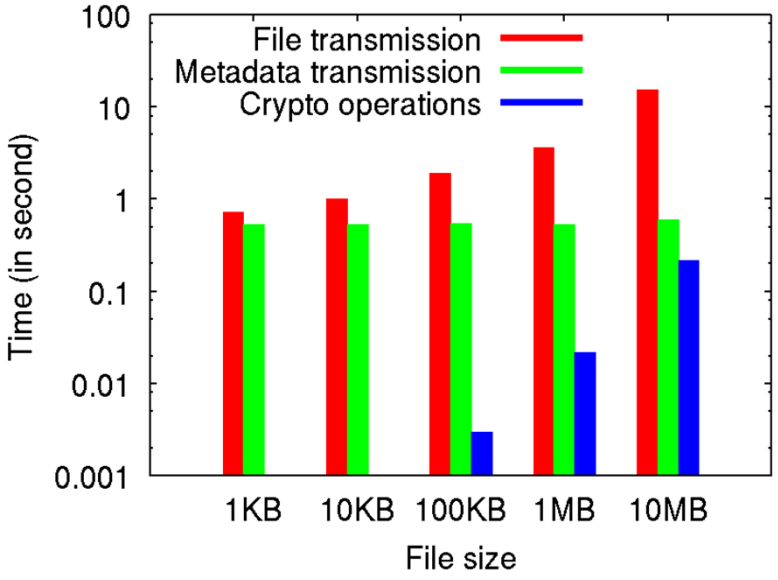
- Interfaces to interact with cloud:
  - **`Upload(file, policy)`**
  - **`Download(file)`**
  - **`Delete(file)`**
  - **`Delete(policy)`**
  - **`Renew(file, new_policy)`**
- Can be exported as library APIs for other implementations of data owner

Experiments

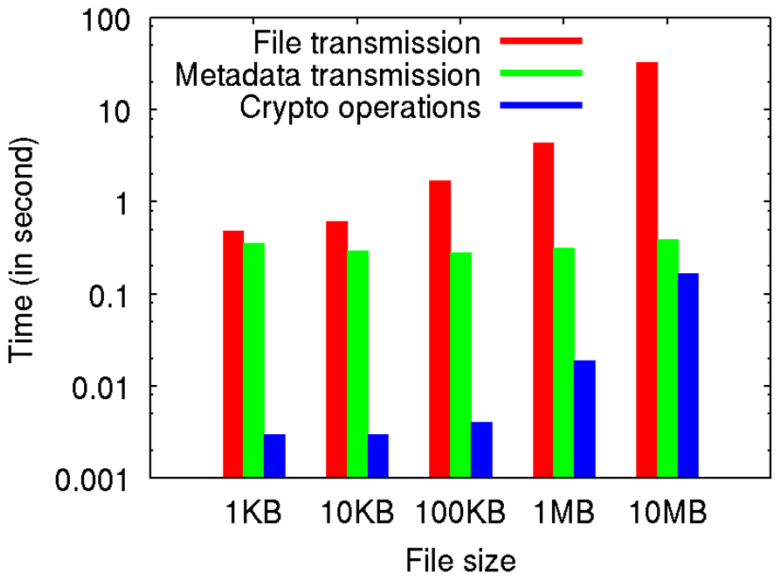
- What is the performance overhead of FADE?
  - e.g., metadata, cryptographic operations
- Performance overhead:
  - Time
    - File transmission time
    - Metadata transmission time
    - Time for cryptographic operations (e.g., AES, HMAC, key exchanges)
  - Space
    - Metadata

File Upload/Download

File upload



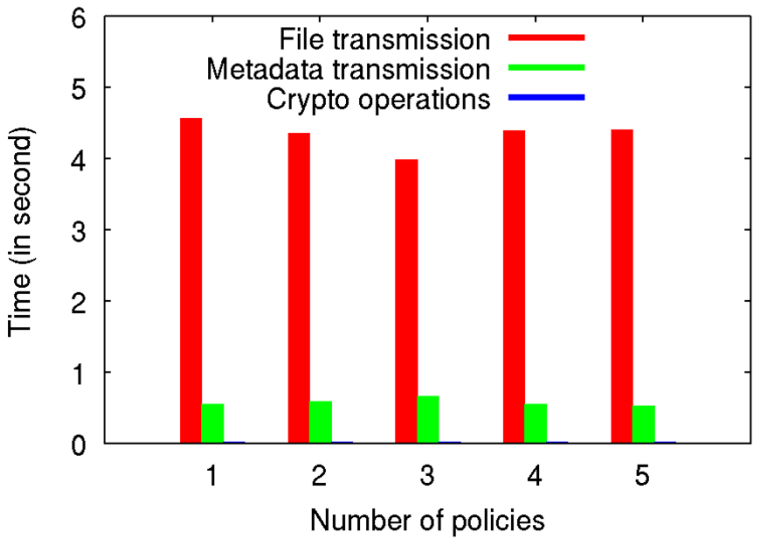
File download



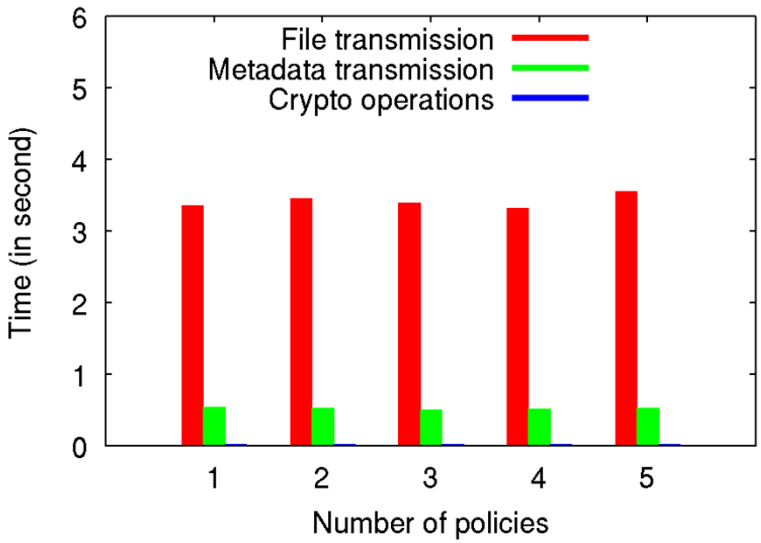
- Overhead of metadata is less if file size is large
- Time for cryptographic operations is small

Multiple Policies

Conjunctive Policies



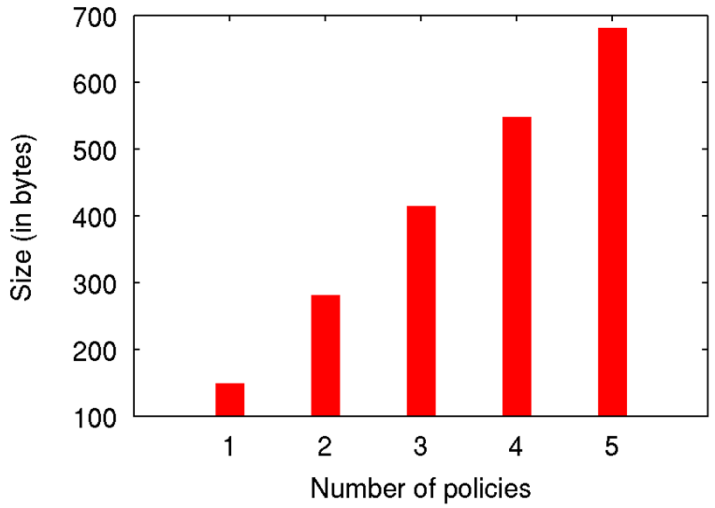
Disjunctive Policies



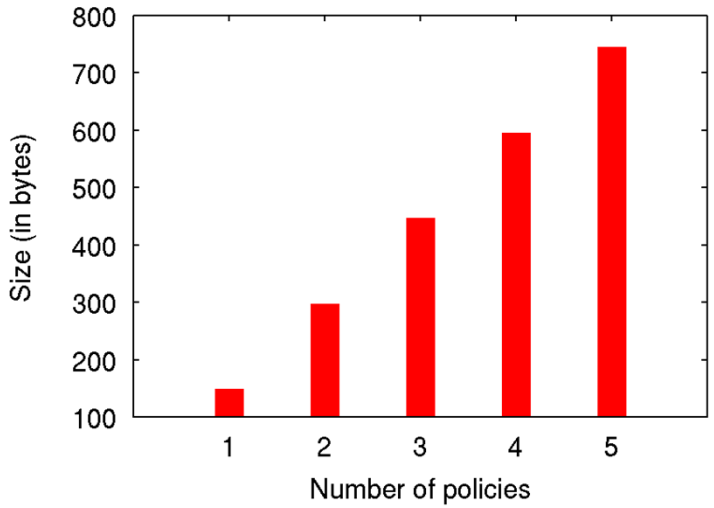
- File size is fixed at 1MB
- Time for cryptographic operations remain low (order of milliseconds) where there are more policies

Space Usage of Metadata

Conjunctive Policies



Disjunctive Policies



- Metadata overhead is less than 1KB for no more than 5 policies

Conclusions

- FADE, an overlay cloud storage system with access control and assured deletion
- Cryptographic operations for policy-based file assured deletion
- Implement a FADE prototype atop Amazon S3
- FADE is feasible in practice

Extensions

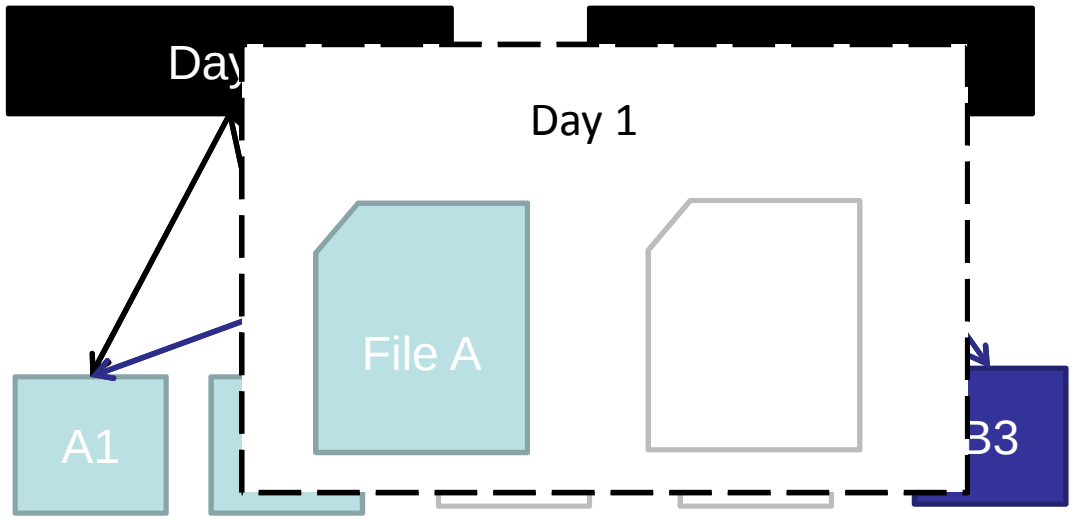
- Quorum scheme of multiple key managers
  - Threshold secret sharing
  - k out of n key shares to recover keys
- Integration with ABE for communication between data owner and key managers

Backup on the Cloud

- How to build a secure cloud backup system?
- Desirable features:
  - Security
  - Version control
- Design considerations:
  - Connectivity performance
  - Monetary cost

Version Control with Deduplication

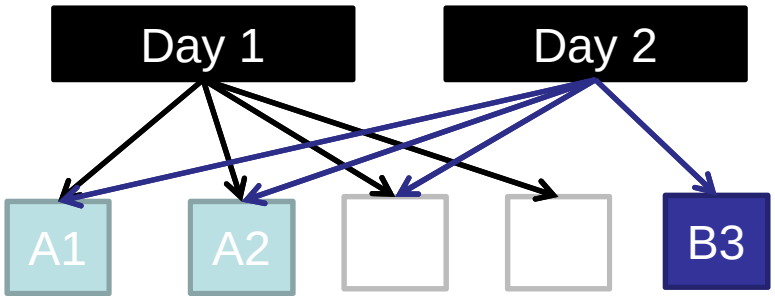
- Cumulus [Vrable et al., ToS'09]:
  - Split the file into blocks
  - Stores the hash of uploaded block in local database
  - Identify new blocks by finding hash in database
  - Upload the parts that have changed.




```
name: fileA
checksum: sha1=...
ctime: 1300000000
data: A/1
    A/2
group: 1 (root)
inode: ...
mode: 0755
mtime: 1300000000
size: 3000000
type: f
user: 1 (root)
```

Version Control + Assured Deletion

- Combining version control and assured deletion is non-trivial
  - Data → Version Control → Assured Deletion
    - Cannot just assure delete all data in day 1 if we want to access data in day 2
  - Data → Assured Deletion → Version Control
    - Same data encrypted by different keys will look different, preventing de-duplication.



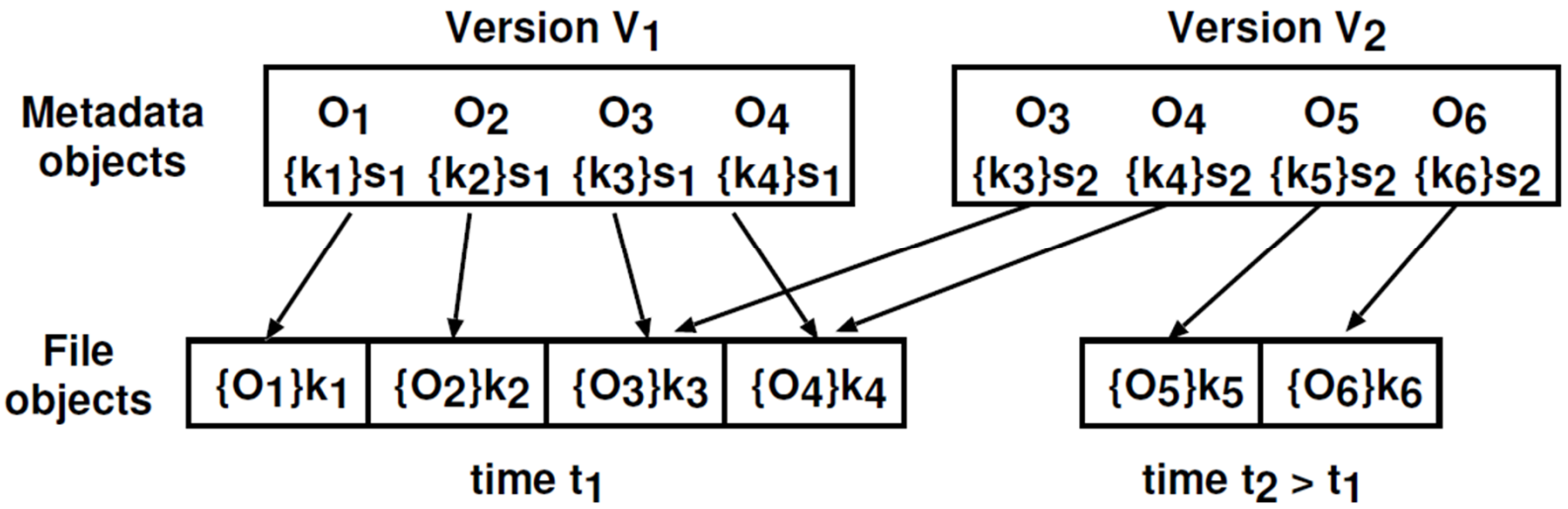
FadeVersion

- Propose **FadeVersion**, a cloud backup system that integrates version control and assured deletion
  - Main idea – using a layered encryption approach

A. Rahumed, H. Chen, Y. Tang, P. Lee, J. Lui,
"A Secure Cloud Backup System with Assured Deletion and Version Control",
CloudSec'11 (ICPP Workshops).

Layered Encryption

- Encrypt each data block with data key.
- Put a copy of the data key of each block in the metadata of each snapshot (version).
- Protect snapshot metadata with policy-based keys.

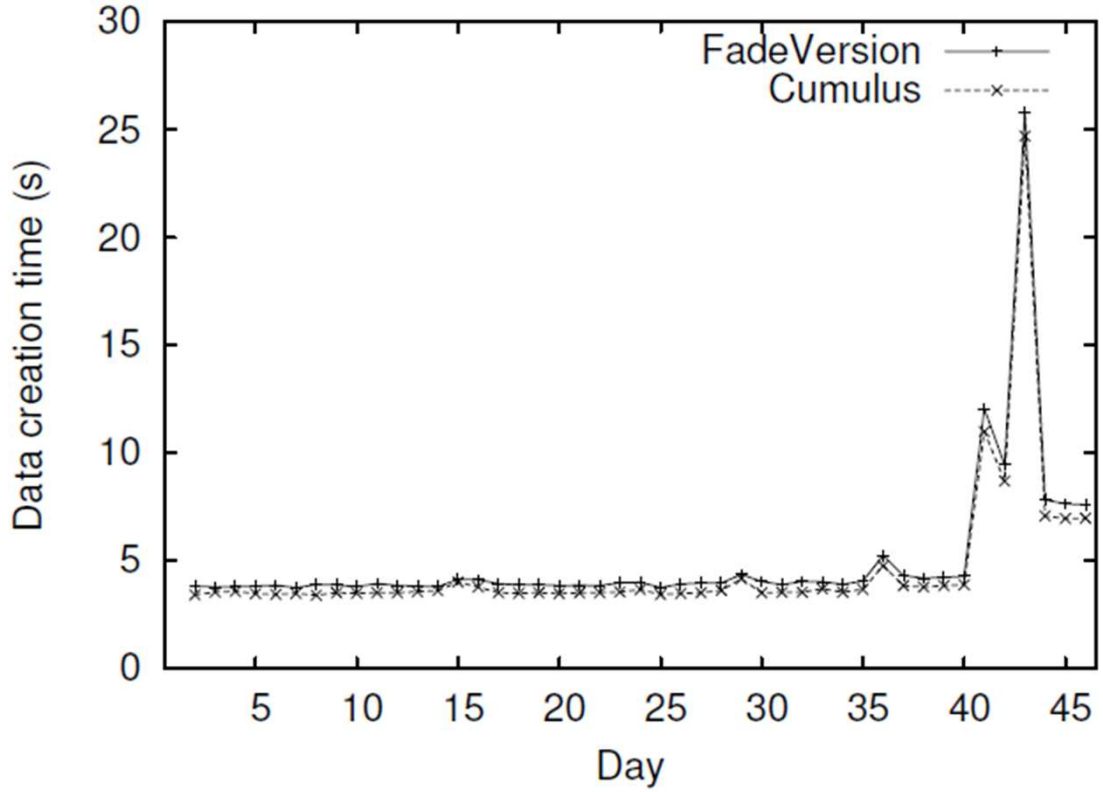


Experiments

- Prototype FadeVersion and run it atop Amazon S3
- Compare with Cumulus
- **Dataset:** 46 days of snapshots of a user's home directory

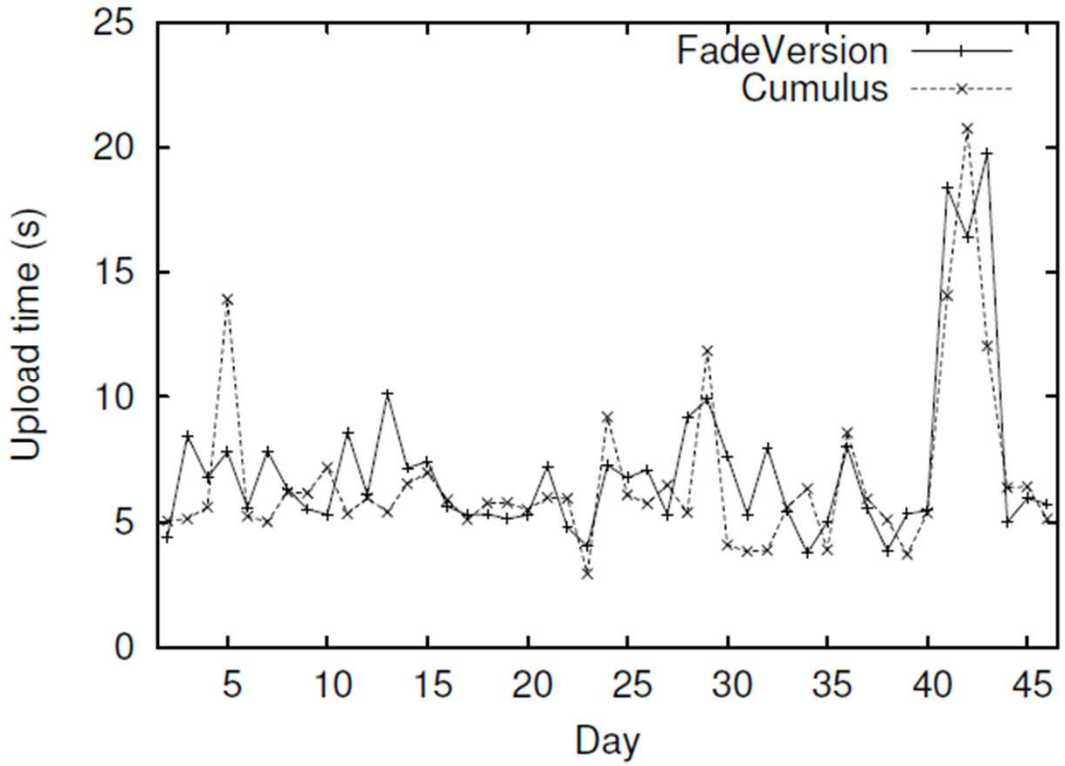
	Day 1	Day 46
Number of files	5590	11946
Median	2054B	1731B
Average	172KB	158KB
Maximum	56.7MB	100MB
Total	940MB	1.85GB

Backup Creation Time



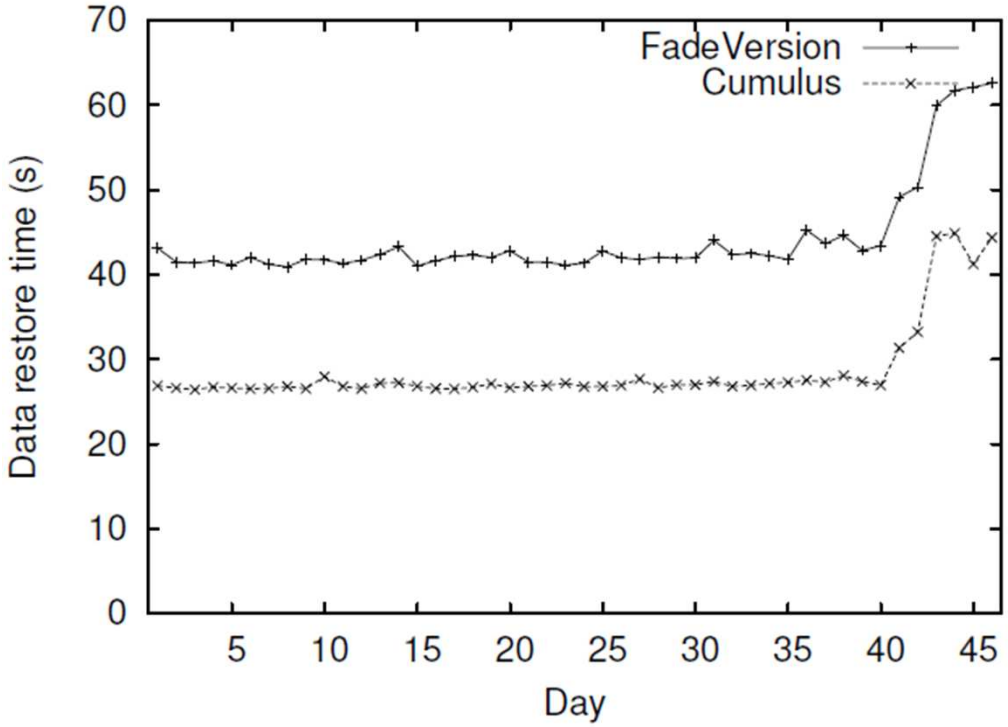
- FadeVersion uses 9.8% more time than Cumulus in creating incremental backups

Upload Time



- Both Cumulus and FadeVersion have very similar values of upload time, and the average values are 6.624 s and 7.106 s, respectively.

Restore Time



- FadeVersion uses 55.1% more time than Cumulus in restoring.
- The overhead is mainly due to the cryptographic operations of decrypting *all* encrypted file objects.

Costs

- Storage costs per month and overall bandwidth cost for 46 days of backup

Providers	Storage Cost $/GB/month	Cumulus	FadeVersion	Bandwidth Cost for updates$/GB	Cumulus	FadeVersion
S3 (Singapore)	0.14	$0.103	$0.108	0.10	$0.0154	$0.0185
Rackspace	0.15	$0.111	$0.115	0.08	$0.0124	$0.0148
Nirvanix SDN	0.25	$0.184	$0.192	0.10	$0.0154	$0.0185
Windows Azure	0.15	$0.111	$0.115	0.10	$0.0154	$0.0185
Google Storage	0.17	$0.125	$0.131	0.10	$0.0154	$0.0185

- Additional storage cost of FadeVersion is no more than $0.08 per month than Cumulus

References

- Publications:
  - Yang Tang, Patrick P. C. Lee, John C. S. Lui, and Radia Perlman, "FADE: Secure Overlay Cloud Storage with File Assured Deletion.", SecureComm 2010, September 2010.
  - Arthur Rahumed, Henry C. H. Chen, Yang Tang, Patrick P. C. Lee, and John C. S. Lui, "A Secure Cloud Backup System with Assured Deletion and Version Control", CloudSec (in conjunction with ICPP'11), September 2011.
- Source code:
  - http://ansrlab.cse.cuhk.edu.hk/software/fade/