Rekeying for
Encrypted Deduplication Storage

Jingwei Li*, Chuan Qin*, Patrick P. C. Lee*, Jin Li#
*The Chinese University of Hong Kong
#Guangzhou University

DSN 2016
Cloud Storage

- Outsourcing data management to public cloud storage is common today

- Challenges:
  - Security in outsourced storage
  - Storage efficiency

- Solutions:
  - **Encryption**: encrypts data before outsourcing
  - **Deduplication**: stores one message for redundant messages with same content
    - Compression can be further applied
Traditional encryption prohibits deduplication

- Same messages encrypted with different keys
  -> different ciphers
Encrypted Deduplication

- **Message-locked encryption** [Bellare, EUROCRYPT’13]
  - Derive encryption key from message itself
  - Same message → Identical cipher text
  - e.g., **convergent encryption**: key = message hash

- **DupLESS** [Bellare, USENIX Security’13]
  - Realizes message-locked encryption
  - A dedicated **key manager** for key generation
  - **MLE key** generated by a **derivation function**
    - Same messages → same ciphers
    - Ciphers appear random
Rekeying renews security protection

- Replaces an existing key with a new encryption key

Benefits:
- Protects against key compromise
- Revokes unauthorized users from accessing data

Challenges:
- Renewing derivation function makes new data fail to be dedup’ed with old data
- Cipher re-encryption is expensive
Rekeying Challenges

- Renewing derivation function (e.g., $K \rightarrow K'$): new data can’t be dedup’ed with old data:

- Cipher re-encryption with new key $K'$:
Our Contributions

- Design REED, a Rekeying-aware Encrypted Deduplication storage system
  - Provides secure and lightweight rekeying
  - Preserves content similarity for deduplication

- Two encryption schemes for REED
  - Basic: high performance (203MB/s)
  - Enhanced: resilient against key leakage (155MB/s)

- Enabling dynamic access control

- Testbed experiments
Threat Model

- **Honest-but-curious** adversary, who can:
  - Compromise storage backend
  - Collude with any revoked client
  - Attempt to learn files beyond access scope
  - Monitor keys returned by key manager

- **Assumptions:**
  - Encrypted and authenticated communication between client and key manager (e.g., by SSL/TLS)
  - Key manager cannot infer message content (OPRF)
  - Key manager is deployed in protected zone
Main Idea

Build security on two symmetric keys:

- **File key**: renewable, controlling access for files
- **MLE key**: unchanged, preserving deduplication

Extends convergent all-or-nothing transform (CAONT) [Li, USENIX ATC’15]

- Encrypts entire message using MLE key; and further encrypts a small part (**stub**) using file key
- Performs deduplication on large part; yet message is unrecoverable with any small part unavailable
- Rekeying on stub (64 bytes, 0.78% for 8KB chunks)
Target workload: backup and archival storage
CAONT

Limitation:

- Secure for unpredictable messages only (otherwise, vulnerable to brute-force dictionary attacks)
Two modifications to CAONT

- Replaces hash key by MLE key from key manager
- Add a canary for integrity checking

Limitation: vulnerable to MLE key compromise
Resilient against MLE key leakage:
- First applies MLE to form a ciphertext
- Then applies CAONT to the MLE ciphertext

Rationale: MLE ciphertext is further protected by CAONT
Comparison

Basic encryption:
- Vulnerable to MLE key compromise
  - Adversary can recover large part (trimmed package) of the original message with MLE key obtained
- Faster encryption

Enhanced encryption:
- Higher security level
  - Adversary needs both MLE key and file key to recover a message
  - Even if MLE key is disclosed, remains secure for unpredictable messages
- Slower encryption
Dynamic Access Control

- Uses CP-ABE for access control [Bethencourt, S&P’07]
- Uses key regression for lazy revocation [Kamara, NDSS’06]
Dynamic Access Control

- **Lazy revocation**
  - Current key state can derive previous states
  - Revoked user cannot access future key states
  - Allows user to access not-yet-updated files
  - Defers file re-encryption (e.g. midnight update)

- **Active revocation**
  - Re-encryption happens immediately with new key
Confidentiality

➢ Level 1: same as DupLESS
  • Adversary can access all trimmed packages, encrypted stubs, and encrypted key states

➢ Level 2: colluding with revoked users
  • Adversary can learn a set of private access keys from any revoked user

➢ Level 3: monitoring key generation
  • Adversary can monitor a subset of revoked users and identify MLE keys returned by key manager
Integrity

- Basic encryption
  - By checking the canary attached to recovered chunks

- Enhanced encryption
  - By comparing the hash of MLE ciphertext
Implementation

Entities:
- **Client**: chunking, encryption/decryption, upload/download
- **Key manager**: MLE key generation
- **Server**: deduplication, metadata storage
- **Cloud**: file recipe, stub, key states

Optimization:
- **Batch** key generation requests to mitigate I/O
- **Cache** previous MLE keys to reduce computation
- **Parallelize** key generation, encryption and upload via multi-threading
Evaluation

Datasets
- Synthetic dataset (2 GB files with unique chunks)
- FSL data trace (147 daily snapshots, 56.2 TB in total)

Testbed
- Servers connected over a Gigabit LAN
MLE Key Generation

- Key generation is expensive (for unique data)
  - REED achieves 17.64 MB/s
Encryption is not a bottleneck

- Basic: 203MB/s
- Enhanced: 155MB/s
Data Transfer

- Caching boosts overall performance
  - First upload bounded by key generation (17MB/s)
  - Second upload of identical data bounded by network (106MB/s)
Rekeying Performance

- Rekeying delays remain small
  - 3.4s for 8 GB data
Storage Efficiency

- **Storage overhead of FSL traces**
  - Storage saving of 98.6%
  - Reduces storage space for ~56 TB logical data to ~431 GB physical data plus ~380 GB stub data
Trace-driven Upload/Download

- 7 consecutive daily backups
  - First day around 13.1 MB/s due to key generation
  - Later transfers reach around 105 MB/s
Conclusions

**REED:**
- Enables rekeying for encrypted deduplication storage
- Proposes two encryption schemes
- Enables dynamic access control
- Implements a prototype
- Conducts extensive trace-driven evaluation

**Software:**
http://ansrlab.cse.cuhk.edu.hk/software/reed