On Deletion of Outsourced Data in Cloud Computing

Authors: Zhen Mo, Qingjun Xiao, Yian Zhou, Shigang Chen
Outline

- Background
- Security Concerns
- Our Concern
- Previous Work
- Problem Definition
- Our Approach
- Experimental Results
System Model

Clients

Third party

Servers
Security Concerns

- Security is one of the major concerns in the cloud storage systems.
- After the clients outsource their data, they will lose physical control of their data.
- Integrity and Confidentiality of outsourced data are major security concerns in existing work.
- They are trying to verify the accessibility or existence of the outsourced data on the cloud servers.
Our Concern

- A different direction.
- Inaccessibility or Nonexistence of the outsourced data on the cloud servers.
- Assured deletion problem in cloud storage systems.
- When clients delete data in the cloud storage system, the deleted data will never resurface in the future even if the clients do not perform the actual data removal themselves.
- Any idea?
Basic Idea

- Encryption
- How many keys to use?
  - One key for all files
  - One key for each file
  - One key for each data item
- Who will maintain the keys?
  - Client
  - A trusted third party
  - Cloud server
Previous Work

- Ephemerizers:
  - Shift key management burden to third-party servers
  - The third party will delete the keys.
- Simultaneous deletion:
  - Group multiple files which should be deleted at the same future time
  - Use one key to protect these files
  - Reduce the number of keys
  - Still need the third party
Problem Definition

- Design an assured deletion solution in cloud storage systems:
  - No third party: The solution does not rely on any third-party server.
  - Small client storage: outsource both data and keys that encrypt the data to the cloud.
  - Fine-grained assured deletion: The client is able to delete any individual data block in any file without causing significant overhead, and the deletion is permanent.
Basic Solutions

- **Master-key Solution:**

  ![Diagram of Master-key Solution]

- **Individual-key Solution:**

  ![Diagram of Individual-key Solution]
Recursively Encrypted Key Tree

- Can we outsource the data keys to the cloud server? But to keep:
  - Confidentiality
  - Integrity and correctness
  - Efficiency
  - Key assured deletion
To store the data keys, the client first constructs a red-black tree with n leaves.
Confidentiality
• Each leaf $w_i$ represents a key $k_i$, where $i$ is the index number.
• For each internal node, the client randomly chooses an auxiliary key $k_x$, where $x$ is a letter.
• Pick a metakey $k^\star$.

We define a value called Encrypted Key (EK) for node $w_x$ as follows:
• $EK(w_x) = \{k_x\}k^\star$ if $w_x$ is the root
• $EK(w_x) = \{k_x\}p_x$ otherwise
• where $p_x$ be the key of $w_x$’s parent node
Recursively Encrypted Key Tree
Recursively Encrypted Key Tree

- **Integrity and Correctness**
  - We make use of the idea of rank based merkle hash tree.
  - We define the rank of a node as the number of leaf nodes in the sub-tree rooted at the node.
  - We define a value called tag for node \( w_x \) as follows:
    - \( t(w_x) = h( EK(w_x) \ || \ r(w_x) \ || \ t_(w_x) ) \);
    - where \( \| \) is the concatenation operator and
    - \( t_(w_x) = NULL \) if \( w \) is a leaf node
    - \( t_(w_x) = h( t(w_l) \ || \ t(w_r) ) \) otherwise
Recursively Encrypted Key Tree

- $w_1: 5 - 2 - 1 - 1 = 1$
Recursively Encrypted Key Tree

- Efficiency
- Self-balancing data structure
- Two new values are defined for each node $w_x$ in the RERK: a color $\text{col}(w_x)$ and a red children $r(w_x)$
- We give the new tag definition as follows:
  \[
t(w_x) = h( EK(w_x) \ || \ r(w_x) \ || \ t_-(w_x) \ || \ \text{col}(w_x) \ || \ \text{red}(w_x) )
  \]
- $t_-(w_x) = \text{NULL}$ if $w_x$ is a leaf node
- $t_-(w_x) = h(t(w_l) \ || \ t(w_r))$ otherwise
Recursively Encrypted Key Tree

- Assured Key Deletion:

Delete $w_4$

Fetch $w_c$ and $w_5$

Re-balancing
Recursively Encrypted Key Tree
Experimental Results

- We implement cloud storage servers on Amazon EC2.
- We use an ordinary desktop computer as a client.
- We use Secure Hash Algorithm-1 (SHA-1) in the modulated hash chain. SHA-1 produces a 160-bit message digest.
- We choose Advanced Encryption Standard (AES) to encrypt each data item and each key. AES has a key size of 128, 192, or 256 bits. In our implementation, we use 128-bit keys.
Communication Overhead

![Graph showing communication overhead for different operations.]

- Delete a data key
- Insert a data key
- Look up a data key

Vertical axis: Average comm. overhead (KB)
Horizontal axis: Total number of data blocks

The graph illustrates the relationship between communication overhead and the total number of data blocks for each operation.
Computational Overhead

- **Client Computational Overhead**

![Graph showing computational overhead vs total number of data blocks]

- Average comp. overhead (ms)
- Total number of data blocks

Lines representing:
- Delete a data key
- Insert a data key
- Look up a data key
Computational Overhead

- Server Computational Overhead

![Graph showing the relationship between computational time (ms) and total number of data blocks. The computational time increases exponentially with the total number of data blocks.]
Thanks you!

Authors: Zhen Mo, Qingjun Xiao, Yian Zhou, Shigang Chen