A Case of Distributed Data Secure Storage and Retrieval in Cloud Computing

Dr. Quan Qian

Shanghai University 12-Oct-11
Outlines

- Security in Cloud Computing
- Distributed Secure Data Storage based on *Hadoop*
- Ciphertext Retrieval with Keywords
- Demo of the Prototype System
Security in Cloud Computing

- Cloud computing is the fundamental change happening in the field of IT. It is a movement towards the intensive and large scale specialization.
- Cloud Computing brings about not only convenience and efficiency, but also great challenges in the field of data security and privacy protection.
- Introduce our recent work on distributed secure data storage, including data encryption and ciphertext retrieval.
Distributed Secure Data Storage based on Hadoop

• **Hadoop** – as a platform for massive data storage, and integrate Google’s *Map-Reduce* programming framework, being widely used in cloud computing.

• **Hadoop** cannot provide complete access control, identity authentication, and *Datanodes’s* data block access control.

• So, how to use this kind of platform to ensure the data security and integrity?
Typical HDFS Architecture
Main Work

• Provide the permission model of r-w-x POSIX standard, not executable, only readable and writable. 3 kind of operations: R(read-only), W(write-only), RW(read-write).

• Key management and distribution

• Encryption and Decryption of Data files

• Performance analysis
Key management and distribution

• Combine symmetric encryption and asymmetric encryption.
Encryption and Decryption of Data Files

- (1) Logical Storage Format of Files
- (2) Physical Storage Format of Files
- (3) File Encryption and Upload
- (4) File Decryption and Download
- (5) File Update
( 1 ) Logical Storage Format of Files

• Some notations
  – $H[\cdot]$ — to compute a hash value as message abstract, hash function usually used such as MD5, SHA-1 et al.
  – $E[\cdot]$ — to encrypt with symmetric ciphers
  – $E_{PUN}[\cdot]$ — to encrypt using public key of user N
  – $E_{PRN}[\cdot]$ — to encrypt using private key of user N

• File Storage Format

| File_Header | $H[\text{File_Header}]$ | $E[\text{File_Content}]$ |
(1) Logical Storage Format of Files

- File Header Format (*fingerprint encrypted by the private key of the file owner*)

<table>
<thead>
<tr>
<th>File_ID</th>
<th>File_Name</th>
<th>File_Owner</th>
<th>File_Description</th>
<th>Creation_Time</th>
<th>PR[Fingerprint]</th>
</tr>
</thead>
</table>

- File Fingerprint Format

<table>
<thead>
<tr>
<th>Items</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Encryption Algorithm</td>
<td>Such as: DES, 3DES, Blowfish …</td>
</tr>
<tr>
<td>(Symmetric Encryption)</td>
<td></td>
</tr>
<tr>
<td>User Information</td>
<td>[File_Owner_A, Authority, PUA(E)]</td>
</tr>
<tr>
<td>(Shared &amp; Unshared)</td>
<td>Optional: [File_Shared_1, Authority, PU1(E)]</td>
</tr>
<tr>
<td></td>
<td>……</td>
</tr>
<tr>
<td>Optional: [File_Shared_N, Authority, PUN(E)]</td>
<td></td>
</tr>
<tr>
<td>File Content Abstract</td>
<td>H[File_Content]</td>
</tr>
</tbody>
</table>
( 2 ) Physical Storage Format of Files

- In order to conveniently read, write and manage file contents, adopt the XML format.
Physical Storage Format of Files

- Some notations of XML storage:
  - **PART A**: ① ⑧ ⑨ respectively stand for file header, abstract of file header, encrypted file content; ②~⑦ respectively represent file id, file name, file owner, file description, file creation time and fingerprint.
  - **PART B**: (a)~(e)respectively stand for encryption algorithm, abstract of file content, user information, file owner and shared users.
  - **PART C**: (1)~ (3) respectively stand for user name, authority and encrypted symmetric key.
(3) File Encryption and Upload

1. Generate a symmetric key PK;

2. Record file owner information and his authority (RW), and then encrypt the symmetric key PK with public key of file owner;

3. Encrypt their related information (user names, assigned authorities, et al.) with public key of file owner to form user information items together with file owner information;

4. Exists shared users?
   - Yes
     - Record encryption algorithm, user information and abstract of file content to generate fingerprint information;
     - Record file ID, file name, file owner, file description, creation time and information encrypted by private key of file owner;
     - Combine file header, the hash value of file header and file content encrypted by symmetric key PK together to form the file storage format, then outsource to the hard disk.
   - No
(4) File Decryption and Download

- Only file owner and shared user assigned \( R \) or \( RW \) can perform download.

- Flow chart of file decryption and download:

<table>
<thead>
<tr>
<th>Tokens</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>XML parsing error; file has been tampered</td>
</tr>
<tr>
<td>R2</td>
<td>file header has been tampered</td>
</tr>
<tr>
<td>R3</td>
<td>file owner information is error; the origin of the file is not trusted</td>
</tr>
<tr>
<td>R4</td>
<td>authority shortage (lack of readable authority)</td>
</tr>
<tr>
<td>R5</td>
<td>asymmetric key pairing is not matched</td>
</tr>
<tr>
<td>R6</td>
<td>file decryption error</td>
</tr>
<tr>
<td>R7</td>
<td>file content has been tampered</td>
</tr>
</tbody>
</table>
(5) File Update

• For File owner:
  – Re-encrypting file, rebuilding the abstract of file content, re-encrypting the fingerprint with the private key.

• For shared user with writable authority:
  – Re-encrypting the fingerprint after updating the file with his own private key;

• Re-encryption operation should coordinated by the **KDC** (*key distribution center)*.
The Sequence Diagram of File Update

1. File Shared User
2. File Update Request
3. Update Agreement
4. Update Rejection
5. Update Agreement
6. Transmit Content
7. Transmit Content
8. File Content
9. End of Writing
Model Realization

- Some software tools:
  - dom4j.jar
  - hadoop-0.20.2-core.jar
  - jdk1.6
  - My Eclipse 7.5
  - bcprov-jdk16-145.jar

Fig: Model Realization Architecture
Performance Analysis

- Bytes increment compared with the original data
- The additional parts includes: XML tags, file attribute information, fingerprint, message abstract, data encryption bytes, etc.

Bytes increment increases almost linearly with the original file size.
Performance Analysis

• Total time overhead of upload and download

\[ T_{\text{total}_\text{upload}} = T_{\text{data}_\text{encryption}} + T_{\text{data}_\text{encapsulation}} + T_{\text{data}_\text{upload}} \]  
\[ T_{\text{total}_\text{download}} = T_{\text{data}_\text{decapsulation}} + T_{\text{data}_\text{decryption}} + T_{\text{data}_\text{check}} + T_{\text{data}_\text{download}} \]

• File upload and download dominate most of the total time overhead.
• With the increase of initial file size, the total time overhead increases steadily and linearly.
Why do we need ciphertext retrieval?

1. Ciphertext Retrieval enables the server to locate all encrypted messages containing the keyword sent by user, but learn nothing else.

2. Server has powerful computing ability, compared with low processing ability of PC.
Recent Research

• Proposing the concept of public key encryption with keyword search scheme ——Boneh et al.

• Proposing the concept of secure channel free public key encryption with keyword search ——Cong Wang et al.

• Proposing a secure free public key encryption with keyword search ——Chunxiang Gu et al.

• Proposing secure ranked keyword search over encrypted cloud data ——Cong Wang et al.
Adv. & Disadv.

- **Advantages** — *the bilinear pairings are effective way to keep the security of data stored in the cloud environment.*

- **Disadvantages** — *time wasting, low performance, low scalability.*
In cloud computing environment, we can choose to:

1) **Construct distributed file system to store encrypted data, promising safety and confidentiality.**

2) **Use Map-reduce framework to run distributed program.**

3) **Optimize the data structure, enabling quick store and search.**
Framework of Data Store and Retrieval

User 1
User 2
Public Key Storage Server
Storage Server 1
Storage Server 2
Storage Server N
Authenticator Server
MapReduce Server
Tracker Server
Public Key Storage Server
Storage Server 1
Storage Server 2
Storage Server N
Elements of the Architecture

• **User**
  – File owner upload the encrypted data to the Storage Server; Shared user retrieve data from the encrypted files.

• **Public Key Storage Server**
  – Store the public keys of all users;

• **Tracker Server**
  – Manage the metadata and task coordination;

• **Storage Server**
  – Distributed storage of the encrypted data;

• **Authentication Server**
  – User registration and authentication;

• **Map-Reduce Server**
  – Build and execute the Map-Reduce task;
Our Work

Bloom Filter
+ Map-Reduce

= Quick Search?
= High Security?
= Lowest Redundancy?
Bloom Filter

• Define a array with $m$ bits and number of $k$ unrelated hash functions;

• For a keyword $W$, compute the hash value of $W$, get $b_1, b_2, \ldots, b_k$ respectively and use them as the array index; meanwhile set the array element to “1”;

• When want to retrieve the keyword, compute the number of $k$ hash value, if the corresponding the element in the array is “1”, then we can say the text contains the keyword.
Bloom Filter schematic diagram

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \ldots & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
\end{array}
\]

\[b_1, 1 = H_1(W_1)\]

\[b_2, 1 = H_2(W_1)\]

\[b_1, 2 = H_1(W_2)\]

\[b_2, 2 = H_2(W_2)\]
Map-reduce schematic diagram
Procedure of the Data Storage

IF user A want to share file with user B, C, D,..., he/she should:

- **Step1** --- User A generate core keywords $kw_1, kw_2, ..., kw_n$ and then encrypt the keywords with A’s public key $\{ E_{pk_A}(kw_j) \}$;
- **Step2** --- Generate the Bloom Filter Vector of the encrypted keywords:
  \[
  \vec{BF} = \{ b_{i,j} \left( E_{pk_A}(kw_j) \right) \} = \{ b_{1,1}b_{1,2}b_{1,3}b_{2,1}b_{2,2}b_{2,3}b_{3,1}b_{3,2}b_{3,3} \}, \ i \in (1,k), \ j \in (1,n)
  \]
- **Step3** --- Encrypt the file with a random key, $E_{key}(msg)$;
- **Step4** --- Require the Key Storage Server to get the shared user’s public key, and use them to encrypt the encryption key, $E_{PK_B}(key), E_{PK_C}(key), ...$
- **Step5** --- User A send the whole data to the storage server.
Procedure of the Data Retrieval

IF user $B$ want to retrieve the file that user $A$ send with keywords $\text{kw}_1, \text{kw}_2, \ldots, \text{kw}_n$, he/she should:

- **Step 1**--- Require the *key Storage Server* to get the user A’s public key $PK_A$;
- **Step 2**--- Compute the keywords set $\{E_{\text{dk}_A}(\text{kw}_j)\}$ and the trapdoor information $\vec{BF}' = \{b_{1,1}b_{2,1} \ldots b_{k,1}b_{1,2}b_{2,2} \ldots b_{k,2} \ldots b_{1,n}b_{2,n} \ldots b_{k,n}\}$, then send it to the server;
- **Step 3**--- Execute Map-Reduce computation and compare, if $\vec{BF}' = \vec{BF}$, then send $\{E_{\text{key}}(\text{msg}), B \| E_{PK_B}(\text{key})\}$ to user $B$;
- **Step 4**--- user B use his own private key to decrypt the symmetric key and then get the original plaintext;
Data Storage Architecture Model
Experimental Results

- Use 4 machines, 1 as *Name node* and 3 as *Data nodes* (also serve as the map-reduce servers);
- Bloom filter use 100 bits array and 2 different hash functions;

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Datanode 1</th>
<th>Datanode 2</th>
<th>Datanode 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory Size</strong></td>
<td>2G</td>
<td>2G</td>
<td>2G</td>
</tr>
<tr>
<td><strong>Max Map Num.</strong></td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td><strong>Max Reduce Num.</strong></td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Execution Time of Map-Reduce

![Graph showing execution time vs. lines of index table for MapReduce and Sequence methods. The x-axis represents the number of lines in the index table (in millions), ranging from 0 to 1000. The y-axis represents time, ranging from 0 to 800. Two lines are plotted: one for MapReduce and one for Sequence. The MapReduce line is consistently below the Sequence line, indicating shorter execution times.]
Conclusions

• Optimized data storing structure and algorithm will enable quick search.
• Distributed computing is a trend in cloud computing environment.
• Data security is a must or a preliminary of the above.
• Further research include how to deploy the prototype system in the real cloud computing environment, key management for massive users, hot nodes identification and optimization, …, etc.

• Demo
• Thank you!

Welcome to Shanghai University