Enabling Cloud Storage Auditing with Key-Exposure Resistance

- Cloud storage auditing is viewed as an important service to verify the integrity of the data in public cloud. Current auditing protocols are all based on the assumption that the client's secret key for auditing is absolutely secure. However, such assumption may not always be held, due to the possibly weak sense of security and/or low security settings at the client.

- If such a secret key for auditing is exposed, most of the current auditing protocols would inevitably become unable to work. In this paper, we focus on this new aspect of cloud storage auditing. We investigate how to reduce the damage of the client's key exposure in cloud storage auditing, and give the first practical solution for this new problem setting.

- We formalize the definition and the security model of auditing protocol with key-exposure resilience and propose such a protocol. In our design, we employ the binary tree structure and the pre-order traversal technique to update the secret keys for the client.

- We also develop a novel authenticator construction to support the forward security and the property of block less verifiability. The security proof and the performance analysis show that our proposed protocol is secure and efficient.

Existing System

- Auditing protocols can also support dynamic data operations. Other aspects, such as proxy auditing, user revocation and eliminating certificate management in cloud storage auditing have also been studied. Though many research works about cloud storage auditing have been done in recent years, a critical security problem exposure problem for cloud storage auditing, has remained unexplored in previous researches. While all existing protocols focus on the faults or dishonesty of the cloud, they have overlooked the possible weak sense of security and/or low security settings at the client.

- Unfortunately, previous auditing protocols did not consider this critical issue, and any exposure of the client's secret auditing key would make most of the existing auditing protocols unable to work correctly.

- we focus on how to reduce the damage of the client's key exposure in cloud storage auditing. Our goal is to design a cloud storage auditing protocol with built-in key-exposure resilience. How to do it efficiently under this new problem setting brings in many new challenges to be addressed below.

- First of all, applying the traditional solution of key revocation to cloud storage auditing is not practical. This is because, whenever the client's secret key for auditing is exposed, the client needs to produce a new pair of public key and secret key and regenerate the authenticators for the client's data previously stored in cloud. The process involves the downloading of whole data

from the cloud, producing new authenticators, and re-uploading everything back to the cloud, all of which can be tedious and cumbersome.

- Besides, it cannot always guarantee that the cloud provides real data when the client regenerates new authenticators. Secondly, directly adopting standard key-evolving technique is also not suitable for the new problem setting. It can lead to retrieving all of the actual files blocks when the verification is preceded. This is partly because the technique is incompatible with block less verification. The resulting authenticators cannot be aggregated, leading to unacceptably high computation and communication cost for the storage auditing.

PROPOSED SYSTEM

- We firstly show two basic solutions for the key-exposure problem of cloud storage auditing before we give our core protocol. The first is a naive solution, which in fact cannot fundamentally solve this problem. The second is a slightly better solution, which can solve this problem but has a large overhead. They are both impractical when applied in realistic settings. And then we give our core protocol that is much more efficient than both of the basic solutions.

Naive Solution

- In this solution, the client still uses the traditional key revocation method. Once the client knows his secret key for cloud storage auditing is exposed, he will revoke this secret key and the corresponding public key. Meanwhile, he generates one new pair of secret key and public key, and publishes the new public key by the certificate update. The authenticators of the data previously stored in cloud, however, all need to be updated because the old secret key is no longer secure. Thus, the client needs to download all his previously stored data from the cloud, produce new authenticators for them using the new secret key, and then upload these new authenticators to the cloud. Obviously, it is a complex procedure, and consumes a lot of time and resource. Furthermore, because the cloud has known the original secret key for cloud storage auditing, it may have already changed the data blocks and the corresponding authenticators. It would become very difficult for the client to even ensure the correctness of downloaded data and the authenticators from the cloud. Therefore, simply renewing secret key and public key cannot fundamentally solve this problem in full.

MODULE DESCRIPTION

Modules:

The system consists of modules and threat modules.

- Public Key and Secret Key
- File Storage
- Generate time period key

- Indexing of files
- View files and download files
- Auditor Public key

Module Explanations:

Public Key & Secret Key:

In this Module public key is generated for authentication for the user to provide the user specification logging.

The secret key is the confidential generated for each candidate during registration

File Storage

The File Storage module the file stored for the further usage of the consumer and the file is provided the option to view and Download based on the time period keys.

Generate time period key;

The time period key is generated such to use the file or to perform operation on it based on time

Indexing of the files

The indexing of the files is specified such that to view the download or to generate key or to download or perform the operation on the file.

View and Download files.

The files can be viewed or download based on the time period key authentication of the user.

Auditor Public Key.

The auditor public key is generated to perform all the operation with a single key on all the modules

SYSTEM CONFIGURATION

SOFTWARE REQUIREMENTS:

- Operating System : Windows 7
- Technology : Java7 and J2EE
- Web Technologies : Html, JavaScript, CSS
- IDE : Eclipse Juno
- Web Server : Tomcat

- Database : My SQL
- Java Version : J2SDK1.5

HARDWARE REQUIREMENTS:

- Hardware : Pentium Dual Core
- Speed : 2.80 GHz
- RAM : 1GB
- Hard Disk : 20 GB
- Floppy Drive : 1.44 MB
- Key Board : Standard Windows Keyboard
- Mouse : Two or Three Button Mouse
- Monitor : SVGA