International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

A Secure Decentralized Erasure for Code-Based Cloud Storage System

Shaik Fathima Zahera

Nawab Shah College of Engineering & Technology (Affliated to JNTUH), Hyderabad, Telangana 500024, India

Abstract: ***A cloud storage system, consisting of a collection of storage servers, provides long-term storage services over the Internet. Storing data in a third party's cloud system causes serious concern over data confidentiality. General encryption schemes protect data confidentiality, but also limit the functionality of the storage system because a few operations are supported over encrypted data. Constructing a secure storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority. We propose a threshold proxy re-encryption scheme and integrate it with a decentralized erasure code such that a secure distributed storage system is formulated. The distributed storage system not only supports secure and robust data storage and retrieval, but also lets a user forward his data in the storage servers to another user without retrieving the data back. The main technical contribution is that the proxy re-encryption scheme supports encoding operations over encrypted messages as well as forwarding operations over encoded and encrypted messages. Our method fully integrates encrypting, encoding, and forwarding. We analyze and suggest suitable parameters for the number of copies of a message dispatched to storage servers and the number of storage servers queried by a key server. These parameters allow more flexible adjustment between the number of storage servers and robustness.***

Keywords: Encryption, Threshold Proxy Re-encryption, Severs, Cloud storage, Storage sever, Decentralized

1. Introduction

This paper describes Farsite, a serverless distributed file system that logically functions as a centralized file server but whose physical realization is dispersed among a network of untrusted desktop workstations. Farsite is intended to provide both the benefits of a central file s access, and reliable data storage) and the benefits of local desktop file systems (low cost, privacy from nosy sysadmins, and resistance to geographically localized faults). Farsite replaces the physical security of a server in a locked room with the virtual security of cryptography, randomized replication, and Byzantine fault-tolerance Farsite is designed to support typical desktop file-I/O workloads in academic and corporate environment. Data robustness is a major requirement for storage systems. There have been many proposals of storing data over storage servers [1], [2], [3], [4], [5]. One way to provide data robustness is to replicate a message such that each storage server stores a copy of the message. It is very robust because the message can be retrieved as long as one storage server survives. Another way is to encode a message of k symbols into a codeword of n symbols by erasure coding. To store a message, each of its codeword symbols is stored in a different storage server. A storage server failure corresponds to an erasure error of the codeword symbol. As long as the number of failure servers is under the tolerance threshold of the erasure code, the message can be recovered from the codeword symbols stored in the available storage servers by the decoding process. This provides a tradeoff between the storage size and the tolerance threshold of failure servers. A decentralized erasure code is an erasure code that independently computes each codeword symbol for a message. Thus, the encoding process for a message can be split into n parallel tasks of generating codeword symbols. A decentralized erasure code is suitable for use in a distributed storage system. After the message symbols are sent to storage servers, each storage server independently computes a codeword symbol for the received message symbols and stores it.

This finishes the encoding and storing process. The recovery process is the same. Storing data in a third party's cloud system causes serious concern on data confidentiality. In order to provide strong confidentiality for messages in storage servers, a user can encrypt messages by a cryptographic method before applying an erasure code method to encode and store messages. When he wants to use a message, he needs to retrieve the codeword symbols from storage servers, decode them, and then decrypt them by using cryptographic keys. There are three problems in the above straightforward integration of encryption and encoding.

First, the user has to do most computation and the communication traffic between the user and storage servers is high. Second, the user has to manage his cryptographic keys. If the user's device of storing the keys is lost or compromised, the security is broken. Finally, besides data storing and retrieving, it is hard for storage servers to directly support other functions. For example, storage servers cannot directly forward a user's messages to another one. The owner of messages has to retrieve, decode, decrypt and then forward them to another user.

In designing Farsite, our goal has been to harness the collective resources of loosely coupled, insecure, and unreliable machines to provide logically centralized, secure, and reliable file-storage service. Our system protects and preserves file data and directory metadata primarily through the techniques of cryptography and replication. Since file data is large and opaque to the system, the techniques of encryption, one-way hashing, and raw replication provide means to ensure its privacy, integrity, and durability, respectively. By contrast, directory metadata is relatively small, but it must be comprehensible and revisable directly by the system; therefore, it is maintained by Byzantine-replicated state-machines [8, 36] and specialized cryptographic techniques that permit metadata syntax enforcement without compromising privacy [15]. One of Farsite's key design objectives is to provide the benefits of

Byzantine fault-tolerance while avoiding the cost of full Byzantine agreement in the common case, by using signed and dated certificates to cache the authorization granted through Byzantine operations. Both Farsite's intended workload and its expected machine characteristics are those typically observed on desktop machines in academic and corporate settings. These workloads exhibit high access locality, a low persistent update rate, and a pattern of read/write sharing that is usually sequential and rarely concurrent [22, 48]. The expected machine characteristics include a high fail-stop rate (often just a user turning a machine off for a while) [6] and a low but significant rate [41] of malicious or opportunistic subversion. In our design, analysis, evaluation, and discussion, we focus on this environment, but we note that corporate administrators might choose to supplement Farsite's reliability and security by adding userless machines to the system or even running entirely on machines in locked rooms. Farsite requires no central administration beyond that needed to initially configure a minimal system and to authenticate new users and machines as they join the system. Administration is mainly an issue of signing certificates: Machine certificates bind machines to their public keys; user certificates bind users to their public keys; and namespace certificates bind namespace roots to their managing machines. Beyond initially signing the namespace certificate and subsequently signing certificates for new machines and users, no effort is required from a central administrator.

2. Literature Survey

Designing a cloud storage system for robustness, confidentiality and functionality. The proxy re-encryption scheme supports encoding operations over encrypted messages as well as forwarding operations over encoded and encrypted messages. To provide data robustness is to replicate a message such that each Storage server stores a copy of the message. It is very robust because the message can be retrieved as long as one storage server survives.

The number of failure servers is under the tolerance threshold of the erasure code, the message can be recovered from the codeword symbols stored in the available storage servers by the decoding process. This provides a tradeoff between the storage size and the tolerance threshold of failure servers.

A decentralized erasure code is an erasure code that independently computes each codeword symbol for a message. A decentralized erasure code is suitable for use in a distributed storage system. A storage server failure is modeled as an erasure error of the stored codeword symbol. We construct a secure cloud storage system that supports the function of secure data forwarding by using a threshold proxy re-encryption scheme. The encryption scheme supports decentralized erasure codes over encrypted messages and forwarding operations over encrypted and encoded messages. Our system is highly distributed where storage servers independently encode and forward messages and key servers independently perform partial decryption.

3. Problem Definition

In this paper, we address the problem of forwarding data to another user by storage servers directly under the command of the data owner. We consider the system model that consists of distributed storage servers and key servers. Since storing cryptographic keys in a single device is risky, a user distributes his cryptographic key to key servers that shall perform cryptographic functions on behalf of the user. These key servers are highly protected by security mechanisms.

The distributed systems require independent servers to perform all operations. We propose a new threshold proxy re-encryption scheme and integrate it with a secure decentralized code to form a secure distributed storage system. The encryption scheme supports encoding operations over encrypted messages and forwarding operations over encrypted and encoded messages.

Advantages of Proposed System

- Tight integration of encoding, encryption, and forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding.
- The storage servers independently perform encoding and re-encryption process and the key servers independently perform partial decryption process.
- More flexible adjustment between the number of storage servers and robustness.

4. System Architecture

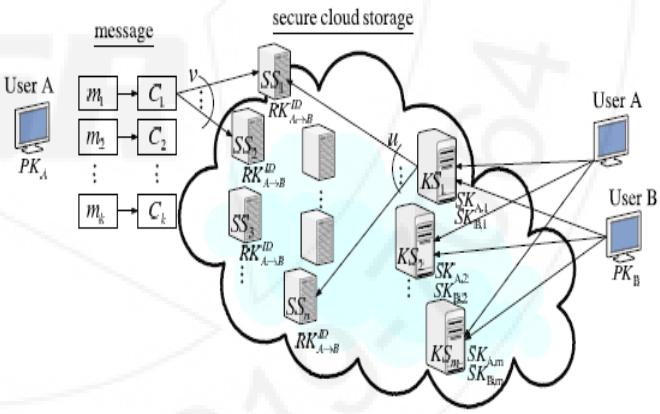

Figure 1

5. Methodologies

5.1 Construction of Cloud Data Storage Module

In Admin Module the admin can login to give his username and password. Then the server setup method can be opened. In server setup process the admin first set the remote servers Ip-address for send that Ip-address to the receiver. Then the server can skip the process to activate or Dis-activate the process. For activating the process the storage server can display the Ip-address. For Dis-activating the process the storage server cannot display the Ip-address. These details can be viewed by clicking the key server. The activated Ip-addresses are stored in available storage server. By clicking

the available storage server button we can view the currently available Ip-addresses.

5.2 Data Encryption Module

In cloud login module the user can login his own details. If the user cannot have the account for that cloud system first the user can register his details for using and entering into the cloud system. The Registration process details are Username, E-mail, password, confirm password, date of birth, gender and also the location. After entering the registration process the details can be stored in database of the cloud system. Then the user has to login to give his corrected username and password the code has to be send his/her E-mail. Then the user will go to open his account and view the code that can be generated from the cloud system.

In Upload Module the new folder can be create for storing the files. In folder creation process the cloud system may ask one question for that user. The user should answer the question and must remember that answer for further usage. Then enter the folder name for create the folder for that user. In file upload process the user has to choose one file from browsing the system and enter the upload option. Now, the server from the cloud can give the encrypted form of the uploading file.

5.3 Data Forwarding Module

In forward module first we can see the storage details for the uploaded files. When click the storage details option we can see the file name, question, answer, folder name, forward value (true or false), forward E-mail. If the forward column display the forwarded value is true the user cannot forward to another person. If the forward column display the forwarded value is false the user can forward the file into another person. In file forward processes contains the selected file name, E-mail address of the forwarder and enter the code to the forwarder. Now, another user can check his account properly and view the code forwarded from the previous user. Then the current user has login to the cloud system and to check the receive details. In receive details the forwarded file is present then the user will go to the download process.

5.4 Data Retrieval Module

In Download module contains the following details. There are username and file name. First, the server process can be run which means the server can be connected with its particular client. Now, the client has to download the file to download the file key. In file key downloading process the fields are username, filename, question, answer and the code. Now clicking the download option the client can view the encrypted key. Then using that key the client can view the file and use that file appropriately.

6. Conclusion and Future Work

In this paper, we consider a cloud storage system consists of storage servers and key servers. We integrate a newly proposed threshold proxy re-encryption scheme and erasure codes over exponents. The threshold proxy reencryption scheme supports encoding, forwarding, and partial decryption operations in a distributed way. To decrypt a message of k blocks that are encrypted and encoded to n codeword symbols, each key server only has to partially decrypt two codeword symbols in our system. By using the threshold proxy re-encryption scheme, we present a secure cloud storage system that provides secure data storage and secure data forwarding functionality in a decentralized structure. Moreover, each storage server independently performs encoding and re-encryption and each key server independently performs partial decryption. Our storage system and some newly proposed content addressable file systems and storage system are highly compatible. Our storage servers act as storage nodes in a content addressable storage system for storing content addressable blocks. Our key servers act as access nodes for providing a front-end layer such as a traditional file system interface. Further study on detailed cooperation is required.

References

[1] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, “Oceanstore: An Architecture for Global-Scale Persistent Storage,” Proc. Ninth Int’l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 190- 201, 2000.
[2] P. Druschel and A. Rowstron, “PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility,” Proc. Eighth Workshop Hot Topics in Operating System (HotOS VIII), pp. 75-80, 2001.
[3] A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, and R. Wattenhofer, “Farsite: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment,” Proc. Fifth Symp. Operating System Design and Implementation (OSDI), pp. 1-14, 2002.
[4] A. Haeberlen, A. Mislove, and P. Druschel, “Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures,” Proc. Second Symp. Networked Systems Design and Implementation (NSDI), pp. 143-158, 2005.
[5] Z. Wilcox-O’Hearn and B. Warner, “Tahoe: The Least-Authority Filesystem,” Proc. Fourth ACM Int’l Workshop Storage Security and Survivability (StorageSS), pp. 21-26, 2008.

Author Profile

Shaik Fathima Zehra is pursuing M.Tech (C.S), working in Nawab Shah College of Engineering & Technology, Hyderabad, Telangana 500024, India and as an Assistant Professor in CSIT department