Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 8, August 2014, pg.118 – 128

RESEARCH ARTICLE

A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding

V. Dharma Rayudu[1], M.Vazralu[2], M. Sandeep[3]

[1]M. Tech IV semester, Dept. of CSE, Malla Reddy College of Engineering and Technology, Hyderabad
dharmarayudu1991@gmail.com
[2]Associate Professor, Dept. of CSE, Malla Reddy College of Engineering and Technology, Hyderabad
Vazram4u@gmail.com
[3]Assistant Professor, Dept. of CSE, Malla Reddy College of Engineering and Technology, Hyderabad
shreesandy@gmail.com

Abstract- Cloud storage is a service model in which data in maintained, managed and backed up remotely and made available to user over a network. Having your data stored offsite in the cloud makes it accessible from anywhere without the hassle of maintaining your own local storage and file-serving systems. It makes all the difference in a disaster, too. This cloud storage system, having collection of storage servers these are providing long term storage service over the internet. Storing the data into third party's cloud system causes concern over data confidentiality. In this cloud some general encryption schemes protect data confidentiality, but also limit the functionality of the storage system because a few operations are supported over encrypted data. The main objective of this project is constructing a secure storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority. In this project we propose a threshold proxy re-encryption scheme and integrate it which 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. Erasure encoding supports the forwarding scheme and applicable in decentralized distributed system. A decentralized erasure code is used to ensure the data robustness in the distributed cloud storage system. In erasure codes, the copy of the message is stored in the each storage servers. If one of the storage servers is failed, the message can be retrieved by one of the surviving server.

Index Terms— Decentralized erasure code, proxy re-encryption, threshold cryptography, secure storage system

1. INTRODUCTION

Cloud storage is an industry term for managed data storage through hosted network (typically Internet-based) service. Several types of cloud storage systems have been developed supporting both personal and business uses. The most basic form of cloud storage allows users to upload individual files or folders from their personal computers to a central Internet server. This allows users to make backup copies of files in case their originals are lost. Users can also download their files from the cloud to other devices, and sometimes also enable remote access to the files for other people to share. Businesses can utilize cloud storage systems as a commercially-supported remote backup solution. Either continuously or at regular intervals, software agents running inside the company network can securely transfer copies of files and database data to third-party cloud servers. Unlike personal data that is generally stored forever, enterprise data tends to quickly grow obsolete and backup systems include retention policies that purse useless data after time limits have passed. Larger companies can also use these systems to replicate large amounts of data between branch offices. Employees working at one site may create new files and have them automatically shared with colleagues in other sites (either locally or in other countries). Enterprise cloud storage systems typically include configurable policies for "pushing" or caching data efficiently across sites. Cloud networks that serve many customers tend to be expensive to build due to the scalability requirements for reliably handling large amounts of data.

Now a Days the high-speed networks and ubiquitous Internet access become available in recent years, many services are provided on the Internet such that users can use them from anywhere at any time. For example, the email service is probably the most popular one. Cloud computing is a concept that treats the resources on the Internet as a unified entity, a cloud. Users just use services without being concerned about how computation is done and storage is managed. In this paper, we focus on designing a cloud storage system for robustness, confidentiality, and functionality. A cloud storage system is considered as a large scale distributed storage system that consists of many independent storage servers.

Data robustness is a major requirement for storage systems. There have been many proposals of storing data over storage servers.

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.

Hear some of 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. To well fit the distributed structure of systems, we require that servers independently perform all operations. With this consideration, 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. The tight integration of encoding, encryption, and forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding. Accomplishing the integration with consideration of a distributed structure is challenging. Our system meets the requirements that storage servers independently perform

encoding and re-encryption and key servers independently perform partial decryption. Moreover, we consider the system in a more general setting than previous works. This setting allows more flexible adjustment between the number of storage servers and robustness.

SCOPE OF THE PROJECT:

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.

2. Related Work

We concisely review distributed storage systems, proxy re- encryption schemes, and integrity checking mechanisms.

2.1 Distributed Storage Systems

At the early years, the Network-Attached Storage and the Network File System provide extra storage devices over the network such that a user can access the storage devices via network connection. Afterward, many improvements on scalability, robustness, efficiency, and security were proposed . A decentralized architecture for storage systems offers good scalability, because a storage server can join or leave without control of a central authority. To provide robust- ness against server failures, a simple method is to make replicas of each message and store them in different servers. However, this method is expensive as z replicas result in z times of expansion. One way to reduce the expansion rate is to use erasure codes to encode messages . A message is encoded as a code word, which is a vector of symbols, and each storage server stores a code word symbol. A storage server failure is modeled as an erasure error of the stored code word symbol. Random linear codes support distributed encoding, that is, each code word symbol is independently computed. To store a message of k blocks, each storage server linearly combines the blocks with randomly chosen coefficients and stores the code word symbol and coefficients. To retrieve the message, a user queries k storage servers for the stored code word symbols and coefficients and solves the linear system.

2.2 Proxy Re-Encryption Schemes

Proxy re-encryption schemes are proposed by Mambo and Okamoto and Blaze et al. During a proxy re- encryption theme, a proxy server will transfer a cipher text under a public key PKA to a new one under another public key PKB by victimization the re-encryption key RKA->B. The server doesn't recognize the plaintext throughout transformation. Ateniese et al.planned some proxy re- encryption schemes and applied them to the sharing operate of secure storage systems. In this scheme, messages are 1st encrypted by the owner then keep during a storage server.

Once a user desires to share his messages, he sends a re- encryption key to the storage server. The storage server re- encrypts the encrypted messages for the approved user. Thus, their system has information confidentiality and supports the information forwarding operates. Our work more integrates encryption, re-encryption, and cryptography specified storage robust- terra firma is reinforced.

Type-based proxy re-encryption schemes planned by Tang[10] provide a stronger graininess on the granted right of a re encryption key. A user can decide which type of messages and with whom he desires to share during this

reasonably proxy re-cryptography schemes. Key-private proxy re- encryption schemes square measure planned by Ateniese et al.In a key-private proxy re-encryption theme, given a re- encryption key, a proxy server cannot verify the identity of the recipient. This type of proxy re-encryption schemes provides higher privacy guarantee against proxy servers. Though most proxy re-encryption schemes use pairing operations, there exist proxy re-encryption schemes while not pairing.

2.3 Integrity Checking Functionality

Another vital functionality concerning cloud storage is that the perform of integrity checking. When a user stores information into the storage system, he now not possesses the info at hand. The user might want to see whether or not are properly kept in storage servers. The idea of obvious information possession [13] and therefore the notion of proof of storage [14] are planned. Later, public auditability of keep information is addressed in [15]. However all of them consider the messages within the clear text kind.

3. PROGRAMMERS DESIGN

As shown in Fig. 1, our system model consists of users, n storage servers SS1,SS2,...,SSn, and m key servers KS1, KS2,...,KSm. Storage servers give storage services and key servers give key management services. They work severally. Our distributed storage system consists of four phases: system setup, data storage, data forwarding, and data retrieval. These four phases are represented as follows.

Within the system setup section, the system manager chooses system parameters and publishes them. Every user A is assigned a public-secret key try (PKA,SKA). User A distributes his secret key SKA to key servers specified every key server KSi holds a key share SKAi, 1< i < m. The key is shared with a threshold t.
In the data storage phase, user A encrypts his message M and dispatches it to storage servers. A message M is decomposed into k blocks m1,m2,...,mk and has an symbol ID. User A encrypts every block mi into a cipher text Ci and sends it to v randomly chosen storage servers. Upon receiving cipher texts from a user, every storage server linearly combines them

with randomly chosen coefficients into a code word image and stores it. Note that a storage server could receive but k message blocks and that we assume that each one storage servers know the value k in advance.
In the data forwarding phase, user A forwards his encrypted message with an identifier ID stored in storage servers to user B such that B can decrypt the forwarded message by his secret key. To do so, A uses his secret key SKA and B's public key PKB to reckon a re-encryption key RKID A->B so sends RKID A->B to all or any storage servers. Every storage server uses the re-encryption key to re-encrypt its code word image for later retrieval requests by B. There- encrypted code word symbol is the mixture of cipher texts below B's public key. So as to tell apart re-encrypted code word symbols from intact ones, we tend to decision them original code word symbols and re-encrypted code word symbols, severally.

Within the information retrieval section, user A requests to retrieve a message from storage servers. The message is either keeps by him or forwarded to him. User A sends a retrieval request to key servers. Upon receiving the retrieval request and executing a correct authentication method with user A, every key server KSi requests u randomly chosen storage servers to induce code word symbols and will partial decryption on the received code word symbols by using the key share SKAi. Finally, user A combines the part decrypted code word symbols to get the initial message M.*System recovering**. **When a storage server fails, a new one is added. The new storage** server queries k on the market storage servers linearly combines the received code word symbols as a replacement one and stores it. The system is then recovered.*

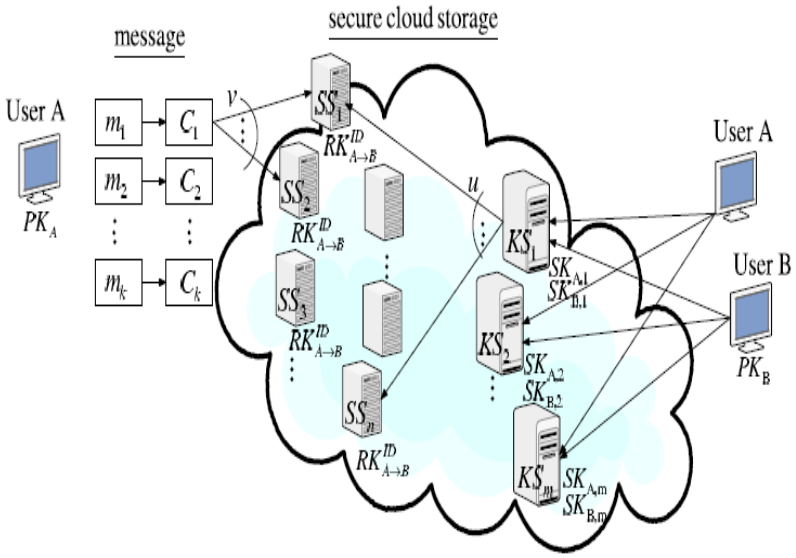

Fig. 1. A general system model of our work.

Threat Model

We consider data confidentiality for both data storage and data forwarding. In this threat model, an attacker wants to break data confidentiality of a target user. To do so, the attacker colludes with all storage servers, non target users, and up to (t _ 1) key servers. The attacker analyzes stored messages in storage servers, the secret keys of non target users, and the shared keys stored in key servers. Note that the storage servers store all re-encryption keys provided by users. The attacker may try to generate a new re-encryption key from stored re-encryption keys. We formally model this attack by the standard chosen plaintext attack1 of the proxy re-encryption scheme in a threshold version, as shown in Fig. 2.

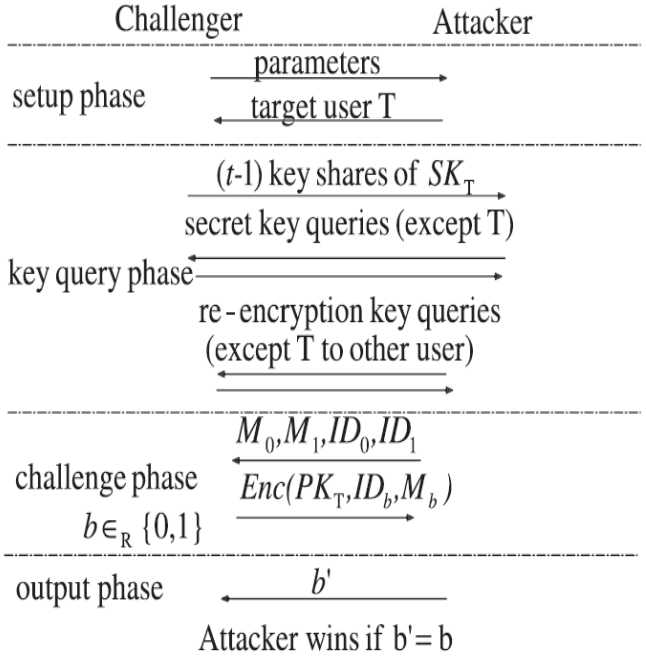

Fig. 2. The security game for the chosen plaintext attack.

Existing System

General encryption schemes protect data confidentiality, but also limit the functionality of the storage system because a few operations are supported over encrypted data. Storing data in a third party's cloud system causes serious concern on data confidentiality. In the existing system, constructing a secure storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority.

DISADVANTAGES OF EXISTING SYSTEM:

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.

4. Proposed System

In the proposed system, 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.

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

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 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. The tight integration of encoding, encryption, and forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding. Accomplishing the integration with consideration of a distributed structure is challenging. Our system meets the requirements that storage servers independently perform encoding and re-encryption and key servers independently perform partial decryption. Moreover, we consider the system in a more general setting than previous works. This setting allows more flexible adjustment between the number of storage servers and robustness.

Advantages:

The threshold proxy re encryption 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 perform 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.

Main Modules:-

MODULES DESCRIPTION:

Construction of Cloud Data Storage Module

This Module deals with the loading of application data into the server and also deals with the registration of new users to the application. It also provides the login feature to the application where the registered users will be able to login into the application and make use of the cloud storage server application.

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.

Data Encryption Module

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. File which was being uploaded into the storage server is firstly encrypted and stores the encrypted file along with the key file in the cloud location. It also stores all the details which was being entered while uploading the file.

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.

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.

Now enter the code which was sent by application admin while registration. This code is used to download the encrypted key file. Decrypt the encrypted file by giving encrypted key file as input and download the original file.

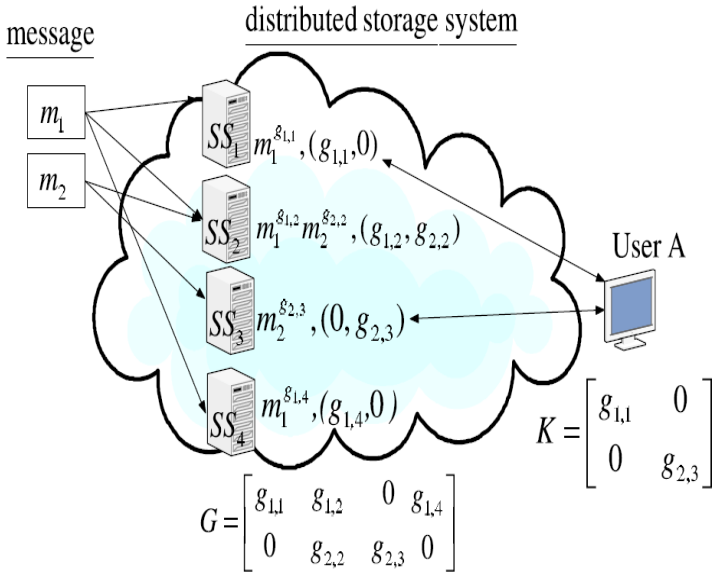

Fig. 3. A storage system with random linear coding over exponents.

5. Software Requirements Specification

A software requirements specification is developed as a consequence of analysis. Review is essential to ensure that the developer and customers have the same perception.

Software requirements specification (SRS) is the starting point of the software development activity. The Software Requirements Specification is produced at the culmination of the analysis task. The introduction of the software requirements specification states the goals and objectives of the software, describing it in the context of the computer-based system. The software requirements specification includes an information description, functional description, behavioral description, validation criteria.

Function Requirements

A functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to

accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases.

The Functional Requirements which are identified in this project are as follows

1. Data Uploading
2. View Data and Forward Data
3. Receiving Data and Download Data

Non Functional Requirements

Non-Functional requirements describe how the product should be implemented. A Non-Functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-Functional requirements are often called qualities of a system. The major Non-Functional requirements of the system are as follows.

USABILITY

The system is designed with completely automated process hence there is no or less user intervention. The end user can easily navigate the entire system as it is developed in windows application. The application gives the status messages regularly based on the user actions performed. Thus the access to this system is very easy.

RELIABILITY

The system is more reliable because of the qualities that are inherited java language. This application donot depend on the external memory and hence its space complexity is very less.

IMPLEMENTATION

The system is implemented in windows environment. Java language is used as front end. The user interface is designed in such a way that it is very flexible and can be easily accessed by the end users. My Sql is used as back end for storing the data.

HARDWARE REQUIREMENTS:

- Processor - Pentium –III
- Speed - 1.1 Ghz
- RAM - 256 MB(min)
- Hard Disk - 20 GB
- Floppy Drive - 1.44 MB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

SOFTWARE REQUIREMENTS:

- Operating System : Windows95/98/2000/XP
- Application Server : Glassfish Server
- IDE : Net Beans 6.8
- Front End : Java, JSP
- Script : JavaScript
- Mail Server : James Server 2.1.3
- Mail Application : Mozilla Thunderbird
- Database : MYSQL

6. Conclusions

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 re-encryption 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 perform 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.

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.

[6] H.-Y. Lin and W.-G. Tzeng, "A Secure Decentralized Erasure Code for Distributed Network Storage," IEEE Trans. Parallel and Distributed Systems, vol. 21, no. 11, pp. 1586-1594, Nov. 2010.

[7] D.R. Brownbridge, L.F. Marshall, and B. Randell, "The Newcastle Connection or Unixes of the World Unite!," Software Practice and Experience, vol. 12, no. 12, pp. 1147-1162, 1982.

[8] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network Filesystem," Proc. USENIX Assoc. Conf., 1985.

[9] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," Proc. Second USENIX Conf. File and Storage Technologies (FAST), pp. 29- 42, 2003.

[10] S.C. Rhea, P.R. Eaton, D. Geels, H. Weatherspoon, B.Y. Zhao, and J. Kubiatowicz, "Pond: The Oceanstore Prototype," Proc. Second USENIX Conf. File and Storage Technologies (FAST), pp. 1-14, 2003.

[11] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G.M. Voelker, "Total Recall: System Support for Automated Availability Management," Proc. First Symp. Networked Systems Design and Implementation (NSDI), pp. 337-350, 2004.

[12] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks through Decentralized Erasure Codes," Proc. Fourth Int'l Symp. Information Processing in Sensor Networks (IPSN), pp. 111-117, 2005.

[13] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized Erasure Codes for Distributed Networked Storage," IEEE Trans. Information Theory, vol. 52, no. 6 pp. 2809-2816, June 2006.

[14] M. Mambo and E. Okamoto, "Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts," IEICE Trans. Fundamentals of Electronics, Comm. and Computer Sciences, vol. E80-A, no. 1, pp. 54-63, 1997.

[15] M. Blaze, G. Bleumer, and M. Strauss, "Divertible Protocols and Atomic Proxy Cryptography," Proc. Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT), pp. 127-144, 1998.

Authors Biography

First Author : V. Dharma Rayudu

M.Tech in CSE from JNTUH,
Malla Reddy College of Engineering and Technology,
Hyderabad

Second Author M.Vazralu

M.Tech in CSE from JNTUH,
Assoc Prof, Dept of CS&E,
Malla Reddy College of Engineering and Technology,
Hyderabad

Third Author M. Sandeep

M.Tech in CSE from JNTUH,
Asst Prof, Dept of CS&E,
Malla Reddy College of Engineering and Technology,
Hyderabad

