International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-3, Issue-1, March-2013

A Cloud Storage System with Data Confidentiality and Data Forwarding

N. Jenefa, J. Jayalakshmi

Abstract*—Cloud storage is a model of networked online storage where data is stored in virtualized pools of storage which are generally hosted by third parties. Organizations cite data confidentiality as their serious concern for cloud computing, with uncrypted data stored on third party's cloud system, The functionality of the storage system is limited when general encryption schemes are used for data confidentiality. With this consideration, we propose a new threshold proxy re-encryption scheme to form a secure distributed storage system. This distributed storage system also lets a user forward his data in the storage servers to another user without retrieving the data back 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.

Index Terms*—Distributed storage system, encoding, proxy re-encryption, uncrypted data.

I. INTRODUCTION

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. With cloud computing growing in popularity, tools and technologies are emerging to build, access, manage, and maintain the clouds. Cloud computing offers many benefits, but it also is vulnerable to threats. As the uses of cloud computing increase, it is highly likely that more criminals will try to find new ways to exploit vulnerabilities in the system. To help mitigate the threat, cloud computing stakeholders should invest heavily in risk assessment to ensure that the system encrypts to protect data; establishes trusted foundation to secure the platform and infrastructure; and builds higher assurance into auditing to strengthen compliance. In this paper, we focus on designing a cloud storage system for robustness, confidentiality, and functionality.

There are many underlying challenges and risks in cloud computing that increase the threat of data being compromised. Security concerns must be addressed in order to establish trust in cloud computing technology. Distributed networked storage systems provide the storage service on the Internet. We address the privacy issue of the distributed networked storage system.

It is desired that data stored in the system remain private even if all storage servers in the system are compromised. The major challenge of designing these distributed networked storage systems is to provide a better privacy guarantee while maintaining the distributed structure. To achieve this goal, we introduce secure decentralized erasure code, which combines a threshold public key encryption scheme and a variant of the decentralized erasure code. Our secure distributed networked storage system constructed by the secure decentralized erasure code is decentralized and robust [4].

Cloud computing is surrounded by many security issues like securing data, and examining the utilization of cloud by the cloud computing vendors. Initial registration with a cloud computing service is a pretty simple process. With shared infrastructure resources, organizations should be concerned about the service provider's authentication systems that grant access to data. After the registration process each user will be given a secret key which is generated by him. The user can store, forward and retrieve data in the cloud only after the secret key generation. In the existing system, if the user loses his key, he will be immediately blocked from the system. This may result in blocking many users. So we introduce a system where the user will be given two chances. Therefore only when the user loses his secret key for two times, he will be blocked from the system.

Because of the huge amount of data stored by a cloud, efficient processing and analysis of data has become a challenging one. One way to provide data robustness is to replicate a message such that each storage server stores a copy of the message. Encoding a message of k symbols into a code word of n symbols by erasure coding is another way to provide data robustness. Each code word symbols is stored in a different storage server to store a message. A decentralized erasure code is an erasure code that independently computes each code word symbol for a message. Thus, the encoding process for a message can be split into n parallel tasks of generating code word symbols [1]. Decentralized storage systems aggregate the available disk space of participating computers to provide a large storage facility [2].These systems rely on data redundancy to ensure durable storage despite of node failures. After the message symbols are sent to storage servers, each storage server independently computes a code word symbol for the received message symbols and stores it. This finishes the encoding and storing process. The recovery process is the same.

Data confidentiality is affected when data is stored in third party's cloud. A user can encrypt messages by a cryptographic method before applying an erasure code method to encode and store messages in order to provide strong confidentiality for messages in storage servers. Erasure coding and reduces the storage cost [3]. He needs to retrieve the code word symbols from storage servers, decode

Manuscript received, on March, 2013.

N.Jenefa, Department of ECE, Saveetha engineering college, Chennai, Tamilnadu, India.

J.Jayalakshmi, Department of ECE, Saveetha engineering college, Chennai, Tamilnadu,India.

them, and then decrypt them by using cryptographic keys when he wants to use a message.

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.

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 [1].

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.

II. SYSTEM ARCHITECTURE

When a user wants to share his messages, he sends a re-encryption key to the storage server. The storage server re-encrypts the messages for the authorized user supportting the data forwarding function. Our work further integrates re-encryption, and encoding such that storage robustness is strengthened.

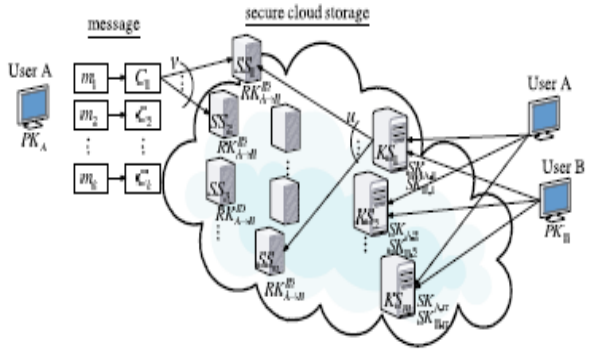

Fig. 1. Model of the system

A. System Implementation

Once the system has been designed, the next step is to convert the designed one in to actual code, so as to satisfy the user requirements as excepted. If the system is approved to be error free it can be implemented. When the initial design was done for the system, the department was consulted for acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system, a demonstration was given to them about working of the system.

The aim of the system illustration was to identify any malfunctioning of the system. Implementation includes proper training to end-users. The implemented software should be maintained for prolonged running of the software. Initially the system was run parallel with manual system. The system has been tested with data and has proved to be error-free and user-friendly. Training was given to end -user about the software and its features. 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.

The data is been encrypted using cryptographic keys to provide data confidentiality. Authenticating the users entering the network can also be done to secure the data. The cryptographic keys must be kept secret and it must not be lost by the user. They must be allowed to enter the system only after registration process or login process. The following steps are followed in our system:

1. User creates an account.
2. His information will be stored in storage server and a key will be given.
3. The authenticated user can upload files.
4. He can also forward and retrieve files.

The user can also forward data to other user by sharing the id of the data and id of the user to the storage server. It forwards the data to the other user. This reduces the computation done by the user

B. Dataflow diagram

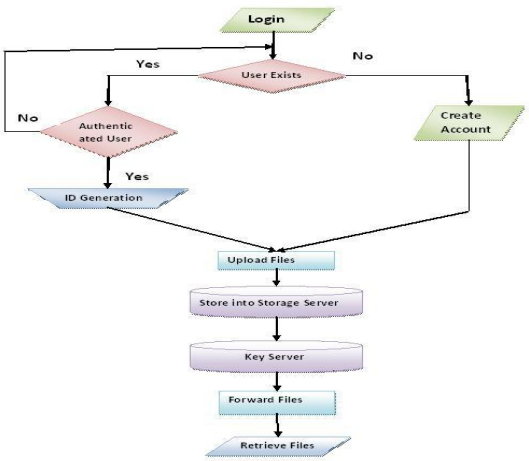

Fig. 2. Process of the system

First the user will be allowed to enter his login id and password to enter into the system. If he is a existing user he will be allowed enter the network otherwise he has to do the registration process. After the authenticaton the user has to generate a id or key for him hrough which he can upload, store and retrive files from the system. This is the login setup.

III. MODULES

A. System Setup

The Set Up process generates the system parameters. A user uses KeyGen to generate his public and secret key pair to share his secret key to a set of m key servers with a threshold t. The user locally stores the third component of his secret key.

B. Data Storage

When user A wants to store a message of k blocks with the identifier ID, he computes the identity token performs the encryption algorithm Encode and k blocks to get k original cipher texts. An original cipher text is indicated by a leading bit b = 0. User A sends each cipher text randomly chosen storage servers. A storage server receives a set of original cipher texts with the same identity token from A. When a cipher text is not received, the storage server inserts to the set. The storage server performs Encoding on the set of k cipher texts and stores the encoded result in the storage server.

C. Data Forwarding

User A wants to forward a message to another user B. He does not take the risk of forwarding the data himself. He just gives the data id to data server and the of the user to whom he wants to send the message to the key server. And the data will be send to the user by the data and key server. This reduces the computation performed by the user.

D. Data Retrieval

There are two cases for the data retrieval phase. The first case is that a user A retrieves his own message. User A informs all key servers with the identity token when he wants to retrieve the message. Original code word symbols are retrieved by the key server and partial decryption is performed on them. The resulting code word is called partially decrypted code word symbol. These symbols and coefficients are sent to user A by the key server. After user A collects replies from at least t key servers and at least k of them, he executes on the t partially decrypted code word symbols to recover the blocks m1; m2; . . .; mk. The second case is that a user B retrieves a message forwarded to him. User B informs all key servers directly. Here the key servers retrieve re-encrypted code word symbols and perform partial decryption.

IV. TABLE DESCRIPTION

A. User Detail

The user detail like username, qualification, department, id, etc. will be stored in the database. When the user enters his user id and password, it will be verified with the database and only if his credentials are correct he will be allowed to enter into the network. This process improves the security of the system.

TABLE I
User details stored in the database

Column Name	Data Type	Description
First Name	Char	First name of the user
Last Name	Char	Last name of the user
Qualification	Char	qualification of the user
Department ID	Numeric	id of the user department
Login ID	Char	id of the user
Password	Char	password of the user

B. Key Server

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.

TABLE 2
Secret keys stored in the storage server

Column Name	Data Type	Description
Login ID	Char	Id of the user
Prod Key	Char	key of the user

V. RESULTS

A. User Login

This is the login page where the user has to enter his user id and password. If he is a new user, he has to undergo registration and his details will be stored in database. When he login the next time, his details will be verified. After that process he will be allowed to enter the system. This results in authenticated user entering the network thus increasing the security of the network.

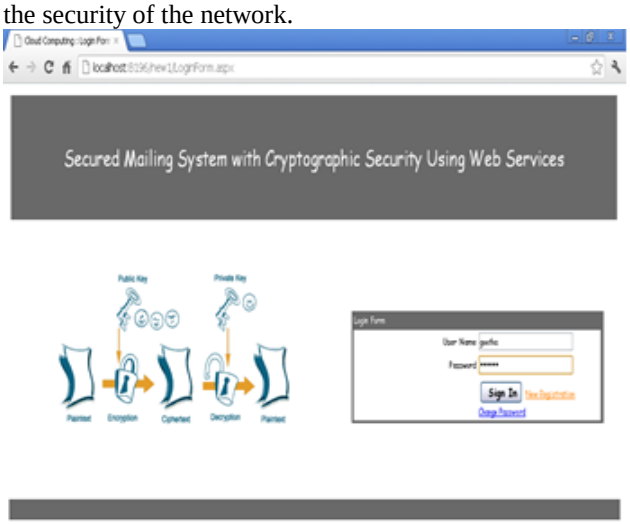

Fig. 3. Screenshot of user login

B. Key generation

Each authenticated user can generate a secret key of his own and only with that secret key, he can store, forward and

retrieve files. Secret key is different to each users. It must be known only to the user. The key can be generated only by the user and after the key generation he can store the message. This secret key must not be shared with other users.

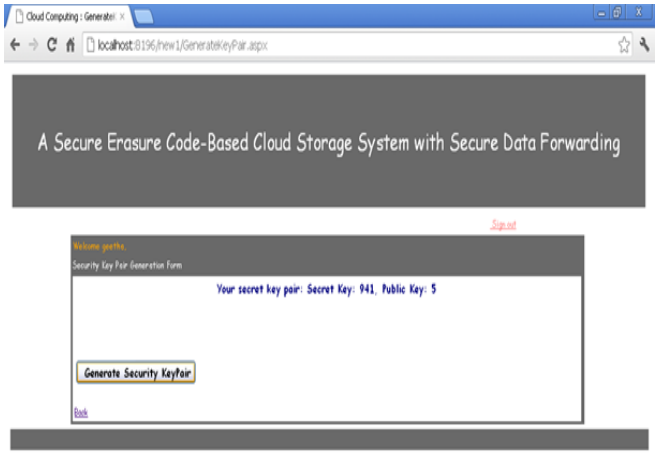

Fig. 4. Screenshot of Key generation

C. *Data Storage*

After secret key generation process, the user can store his message in the storage server. He can also forward and retrieve his message. The messages of the user will be stored in the storage server. The storage server consists of the data of the user and secret keys. The messages of the user can be encrypted using the secret key.

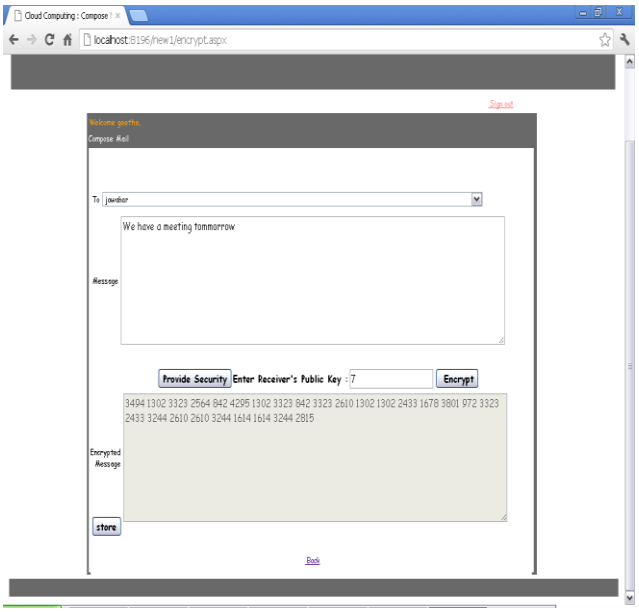

Fig. 4. Screenshot of data storage

VI. CONCLUSION AND FUTURE WORK

Thus by using a secure cloud system was setup and the users were able to store their data. Future work is to newly propose a more secured system in which only authorized users can access all the data in the cloud. If the users access data without permission they must be blocked from the network. A threshold proxy re-encryption scheme can also be proposed that supports encoding, forwarding, etc. Data forwarding method can be more secured using cryptographic keys.

REFERENCES

[1] Hsiao-Ying Lin, Member, IEEE, and Wen-Guey Tzeng, Member ”A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding”IEEE Transactions On Parallel And Distributed Systems, VOL. 23, NO. X, XXX 2012.

[2] 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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.