ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 3, March 2013

AUTOMATED AUDITING AND LOGGING MECHANISM FOR SECURE DATA STORAGE IN CLOUD USING PROXY RE-ENCRYPTION

J.Shyamala, D.Femila, B.Vinisha Cathrine Antonus

Abstract: High-speed networks and ubiquitous Internet access become available to users for access anywhere at anytime. Cloud computing is a concept that treats the resources on the Internet as a unified entity, a cloud. 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. 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. We propose an object-centered approach that enables enclosing our logging mechanism together with users' data and policies. To strengthen user's control, we also provide distributed auditing mechanisms. Users will send their data along with any policies such as access control policies and logging policies that they want to enforce, enclosed in JAR files, to cloud service providers. Any access to the data will trigger an automated and authenticated logging mechanism local to the JARs. Our system is highly distributed where storage servers independently encode and forward messages and key servers independently perform partial decryption.We analyze suitable parameter for more flexible adjustment between the number of storage servers and robustness. We provide extensive experimental studies that demonstrate the efficiency and effectiveness of the proposed approaches.

J.Shyamala,*Department of computer science and engineering, Anna University, Tuticorin, India. 91-9791657924*

D.Femila,*Department of computer science and engineering, Anna University, Tuticorin, India. 91-7708464047*

B.Vinisha Cathrine Antonus,*Department of computer science and engineering, Anna University, Tuticorin, India.91-9597237148.*

Index terms: Decentralized Erasure Code ,Proxy Re-encryption,Accountability and Data Sharing.

1 INTRODUCTION

As 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. In this, focus on designing a cloud storage system for confidentiality. Data robustness is a major requirement for storage systems. One way to provide data robustness is to replicate a message [3]. Another way is to encode a message of k symbols into a codeword of n symbols by erasure coding A decentralized erasure code is an erasure code that independently computes each codeword symbol for a message[4]. 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.

Data should be stored in the third party cloud system causes data confidentiality[1]. 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. Straightforward integration method causes some problems of encryption and encoding. computation and the communication traffic between the user and storage servers is high. Second, the user has to manage their cryptographic keys carefully. Finally, besides data storing and retrieving, it is hard for storage servers to directly support other functions 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.with this consideration, 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. Our system meets the requirements that storage servers independently perform encoding and re-encryption and key servers independently perform partial decryption.In order to,data handling can be outsourced by the direct cloud service provider (CSP) to other entities in the cloud and these entities can also delegate the tasks to others, and so on.Second, entities are allowed to join and leave the cloud in a flexible manner. As a result, data handling in the cloud goes through a complex and dynamic hierarchical service chain which does not exist in conventional environments. To overcome the above problems, we propose a novel approach, namely Cloud Information Accountability (CIA) framework, based on the notion of information accountability.

2 RELATED WORKS

We briefly review distributed storage systems, proxy re-encryption schemes.

2.1 Distributed Storage Systems

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 robustness against server failures, to make replicas of each message and store them in different servers[3]. 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 message . A message is encoded as a codeword, which is a vector of symbols, and each storage server stores a codeword symbol. A storage server failure is modeled as an erasure error of the stored codeword symbol. Random linear codes support distributed encoding, that is, each codeword symbol is independently computed. Lin and Tzeng [3] addressed robustness and confidentiality issues by presenting a secure decentralized erasure code for the networked storage system. In addition to storage servers, their system consists of key servers, which hold cryptographic key shares and work in a distributed way.

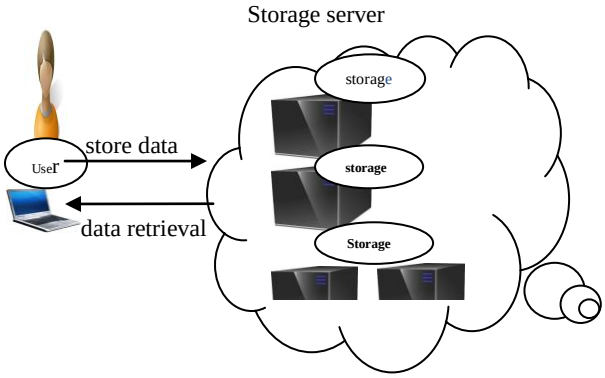

Fig.1. Distributed Storage Server

2.2 Proxy Re-Encryption Schemes

The Proxy Re-encryption schemes are cryptosystems which allow third-parties (proxies) to alter a ciphertext which has been encrypted for one party, so that it may be decrypted by another. Proxy re-encryption schemes are similar to traditional symmetric or asymmetric encryption schemes, Delegation - allows a message recipient (keyholder) to generate a re-encryption key based on his secret key and the key of the delegated user. This re-encryption key is used by the proxy as input to the re-encryption function, which is executed by the proxy to translate ciphertexts to the delegated user's key. Asymmetric proxy re-encryption schemes come in bi-directional and uni-directional varieties[9]. In a proxy re-encryption scheme, a proxy server can transfer a ciphertext under a public key PK_A to a new one under another public key PK_B by using the re-encryption key $RK_{A->B}$. The server does not know the plaintext during transformation.. In their work, messages are first encrypted by the owner and then stored in a storage server. When a user wants to share his messages, he sends a re-encryption key to the storage server. The storage server re-encrypts the encrypted messages for the authorized user. Thus, their system has data confidentiality and supports the data forwarding function.

2.3 Accountability Framework

A novel highly decentralized information accountability framework to keep track of the actual usage of the users' data in the cloud. In particular, an object centered approach is proposed that enables enclosing our logging mechanism together with users' data and policies. The JAR programmable capabilities is used for creating a dynamic and traveling object, and to ensure that any access to the users' data will trigger authentication and automated logging local to the JARs.

3 SCENARIO

We present the scenario of the storage system, that we consider for the confidentiality issue.

3.1 System Model

As shown in Fig. 2, our system model consists of users, n storage servers pk_1; pk_2; . . . ; pk_n, and m key servers KS_1; KS_2; . . . ; KS_m. Storage servers provide storage services and key servers provide key management services.

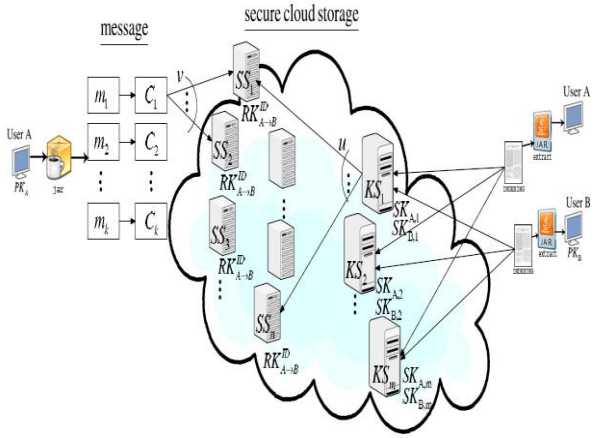

Fig.2. System Overview

In this storage system consists of four phases: Authentication Module,Jar Generation Module,CRL Verification Module,Jar Modifying Module.

Authentication Module: In login module the user has to register their details in cloud system. Before entering this module the user has to check the web services in the cloud system. The registration process contains the following fields. There are name, Email, password, date of birth and sex. To fill these details and click the enter option. Then suddenly the secret key will generate and send to the user's mail-id. After that the user can go and check his/her mail to copy the received secret key or master key. Now, the user has to login by giving his/her name and password and enter into the cloud system. The cloud system contains jar generation process, CRL verification process and the data owner process or jar modifying process.

Data storage: In data storage the new folder can be create for storing the files. 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.

JAR Generation Module: In jar generation module first the user can browse the directory from our computer. After selecting the directory the jar name should be entered. The master key or secret key can entered from our email. Then we have to select anyone (Read or Write) operation to perform. After selecting the operation click the enter option. Now, the user has to view the jar file. Then the jar file should create the log file for that process. By creating the log file the size of the file can be reduced. The master key should be varying for every person. In this module the image files also can selected in read or write operation. The size of the file can reduced accordingly.

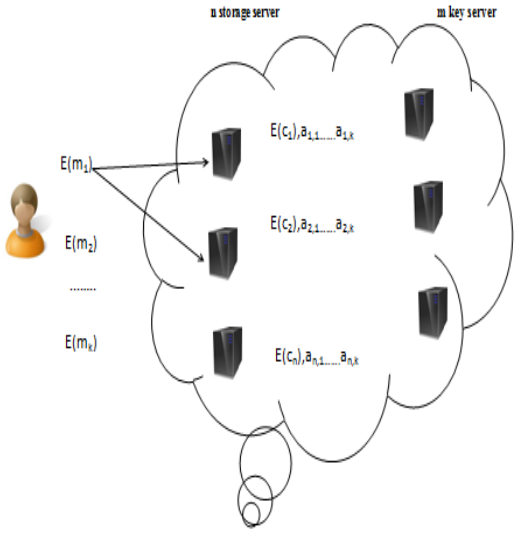

Fig.3,Storing process

CRL Verification Module: In CRL verification module the user will get key for further processing. First, the user can enter the secret key and mail-id. Then select the jar file list. Then click the enter option the jar file key (IBE key) will be send to user's mail. After that the user can go and verify the mail. Then the user to enter the secret key, mail-id, jar file list and also the signature of the jar file. To enter the IBE key from his/her mail it will be checked by the certificate authority. The signature is verified the user has to enter otherwise the user will not enter. The secret key is not matched the cloud system will send to the notification message (signature not matched) to user's mail.

JAR Modifying Module: In jar modifying module the data owner has to enter the secret key for the jar file. If the IBE key value is matched the certificate authority allows permission to process. If the IBE key value is not matched the certificate authority will not allows

permission to process. The IBE key has to match then the user can go and see the message jar file and corresponding log file. The IBE key is not matched the cloud system will send to the notification message (signature not matched) to data owner's mail. Here, the data owner has to forget the secret key his using the forget key option. In this option contains secret key, mail-id, jar file list and enter. Now the newly modified secret key is generated and it will send to the mail.

4 SECURE CLOUD STORAGE SYSTEMS

Before presenting our storage system, we briefly introduce the algebraic setting, cyclic multiplicative group for erasure code, and multiplicative homomorphic approach.

Bilinear map.Let G_1 and G_2 be cyclic multiplicative groups with a prime order p and g € G_1 be a generator. A map $\tilde{e} : G_1 \times G_1 \rightarrow G_2$ is a bilinear map if it is efficiently computable and has the properties of bilinearity and nondegeneracy where $\tilde{e}(g^a,h^b)=\tilde{e}(g,h)^{ab}$[1].

cyclic multiplicative group for Erasure codes: We consider that the message domain is the cyclic multiplicative group G2 described. An encoder generates a generator matrix G =$|g_{i,j}|$ for $1 \le i \le k$; $1 \le j \le n$ as follows: for each row, the encoder randomly selects an entry and randomly sets a value from Z^*_p to the entry. The encoder repeats this step n times with replacement for each row. An entry of a row can be selected multiple times but only set to one value. The values of the rest entries are set to 0. Let the message be $(m_1;m_2; \ldots;m_k)$ € G^k_2.The encoding process is to generate $(w_1; w_2; \ldots; w_n)$ € G^n_2.The first step of the decoding process is to compute the inverse of a k x k submatrix K of G. Let K be $|g_{i,j}|$ for $1 \le i \le k$; $1 \le j \le n$. The final step of the decoding process is to compute the original message.

Multiplicative homomorphic:We use a threshold proxy re-encryption scheme with multiplicative homomorphic property. An encryption scheme is multiplicative homomorphic if it supports a group operation ⊙ on encrypted plaintexts without decryption[3]

$$E(m_1)^a \odot E(m_2)^b = E(m^a_1 \odot m b_2)$$

where E is the encryption function.

Key gen:pk=($\tilde{e}$,g,g^x,p); sk=(x)

Where pk is the storage server and sk is the key server.

Encrypt:h is the message ID

E(m)=(α,β,γ)=(g^r,h,mẽ(g^x.h^r))

Decrypt:

$$D(\alpha,\beta,\gamma)=\frac{\gamma}{\tilde{e}(\alpha,\beta)^{\hat{x}}}= m\tilde{e}(g^x,h^r)/\ \tilde{e}(g^r,h^x)=m$$

Thus, a multiplicative homomorphic encryption scheme supports the encoding operation over encrypted messages. We then convert a proxy re-encryption scheme with multiplicative homomorphic property into a threshold version In our system, to decrypt for a set of k message symbols, each key server independently queries to storage servers and partially decrypts two encrypted codeword symbols. As long as t key servers are available, k codeword symbols are obtained from the partially decrypted ciphertexts.

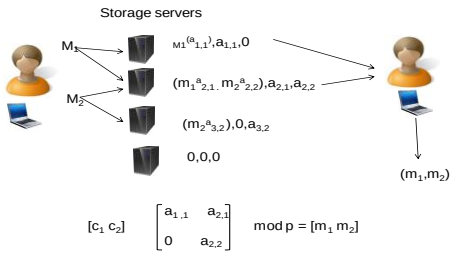

Fig.4. Decentralized Erasure Code for Storage

4.1 Performance analysis

Log creation time: In the experiments, the times taken to create a log file for original file and JAR files.The size of the files are denoted in KB.The time is denoted by milli sec(ms).From that graph half of the file size should be reduced by using compressed JAR.

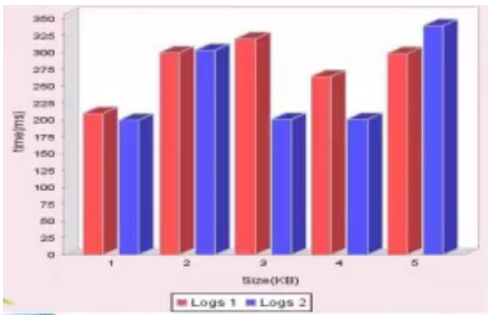

Fig.5.Log creation time

5. DISCUSSION AND CONCLUSION

In this paper, we consider a cloud storage system consists of storage servers and key servers. The 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. The logger is strongly coupled with user's data

(either single or multiple data items). Its main tasks include automatically logging access to data items that it contains, encrypting the log record using the public key of the content owner. Moreover, one of the main features of this work is that it enables the data owner to audit even those copies of its data that were made without their knowledge.

ACKNOWLEDGMENT

The author are indebted to Prof.R.sivasubra Narayanan and prof.G.Sindhu for providing the source code for algorithm.In addition,the author would like to thank Prof.A.Jeyamurugan for providing the experimental setup for implementing testing and verification of this work.

REFERENCES

[1]Hsiao-Ying Lin and Wen-Guey. Tzeng, "A Secure Erasure Code based Cloud Storage System with Secure Data Forwarding ," IEEE trans. parallel and distributed systems, vol. 23, no. 06, pp. jun 2012.
[2]smitha Sundareswaran, Anna C. Squicciarini And Dan Lin,"Ensuring Distributed Accountability For Data Sharing In The Cloud", IEEE Transactions On Dependable And Secure Computing Vol.9 No.4 Year 2012.
[3] 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.
[4] 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.
[5] S. Pearson and A. Charlesworth, "Accountability as a Way Forward for Privacy Protection in the Cloud," Proc. First Int'l Conf. Cloud Computing, 2009.
[6] A. Pretschner, F. Schuo¨ tz, C. Schaefer, and T. Walter, "Policy Evolution in Distributed Usage Control," Electronic Notes Theoretical Computer Science, vol. 244, pp. 109-123, 2009.
[7] R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely, "Towards a Theory of Accountability and Audit," Proc. 14th European Conf. Research in Computer Security (ESORICS), pp. 152-167, 2009.
[8] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," ACM Trans. Information and System Security, vol. 9, no. 1, pp. 1-30, 2006.
[9] Q. Tang, "Type-Based Proxy Re-Encryption and Its Construction," Proc. Ninth Int'l Conf. Cryptology in India: Progress in Cryptology (INDOCRYPT), pp. 130-144, 2008.
[10] M. Xu, X. Jiang, R. Sandhu, and X. Zhang, "Towards a VMM Based Usage Control Framework for OS Kernel Integrity Protection, SACMAT '07: Proc. 12th ACM Symp. Access Control Models and Technologies, pp. 71-80, 2007

J.Shyamala received the B.E. degree in Computer Science and Engineering from the Anna University, Tamilnadu, India, in 2011, and is currently pursuing the M.E. degree in computer science and Engineering at Anna University, Tamilnadu.

D.Femila received the B.E. degree in Computer Science and Engineering from the Anna University, Tamilnadu, India, in 2011, and is currently pursuing the M.E. degree in computer science and Engineering at Anna University, Tamilnadu.

B.Vinisha Cathrine Antonus received the B.E. degree in Computer Science and Engineering from the Anna University, Tamilnadu, India, in 2008, and is currently pursuing the M.E. degree in computer science and Engineering at Anna University, Tamilnadu.