International Journal of Smart Sensor and Adhoc Network

Volume 3 | Issue 1 Article 3

October 2012

A SECURE DATA FORWARDING SCHEMA FOR CLOUD STORAGE SYSTEMS

G.CHINNA PULLAIAH
CSE Department, Affiliated to JNTUA University., SJCET- Kurnool., pullaiahgcp@gmail.com

DILIP VENKATA KUMAR VENGALA
CSE Department, Affiliated to JNTUA University., SJCET- Kurnool., vdilipvenkatakumar@gmail.com

A SECURE DATA FORWARDING SCHEMA FOR CLOUD STORAGE SYSTEMS

G.CHINNA PULLAIAH [1], DILIP VENKATA KUMAR VENGALA[2]

[1]M.Tech(CSE) Student ,[2]Associate Professor,CSE Department, Affiliated to JNTUA University., SJCET- Kurnool.
E-mail: [1]pullaiahgcp@gmail.com, [2]vdilipvenkatakumar@gmail.com

Abstract- Cloud Computing has been envisioned as the next-generation architecture of IT Enterprise. It moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. This unique paradigm brings about many new security challenges, which have not been well understood. This work studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, we consider the task of allowing a threshold proxy re-encryption, on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of the client through the auditing of whether his data stored in the cloud are indeed intact, which can be important in achieving economies of scale for Cloud Computing. 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, since services in Cloud Computing are not limited to archive or backup data only. While prior works on ensuring remote data integrity often lacks the support of either public Audit ability or dynamic data operations, this paper achieves both. We first identify the difficulties and potential security problems of direct extensions with fully dynamic data updates from prior works and then show how to construct an elegant verification scheme for the seamless integration of these two salient features in our protocol design A decentralized erasure code is an erasure code that independently computes each codeword symbol for a message, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis show that the proposed schemes are highly efficient and provably secure.

Keywords- *Decentralized erasure code, proxy re-encryption, threshold cryptography, secure storage system.*

I. 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. 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 [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 this paper, we are address the problem of directly data forwarding to another user by storage servers under the control 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 precarious, 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 difficult. 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. EXISTING SYSTEM

In Existing System we use a straightforward integration method. In straightforward integration method 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 General encryption schemes protect data confidentiality, but also limit the functionality of the storage system because a few operations are supported over encrypted data. A decentralized architecture for storage systems offers good scalability, because a storage server can join or leave without control of a central authority.

III. DRAWBACKS OF EXISTING SYSTEM

1) The user can perform more computation and communication traffic between the user and storage servers is high.
2) The user has to manage his cryptographic keys otherwise the security has to be broken.
3) The data storing and retrieving, it is hard for storage servers to directly support other functions.
4) The user is unable to share the data confidentiality to the destination.

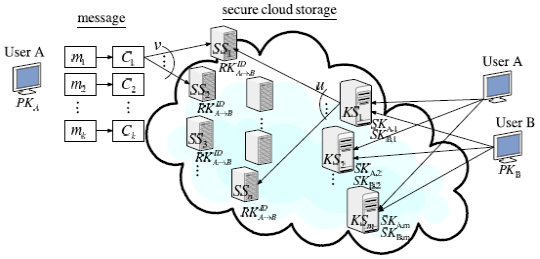

Fig. 1. A general system model of our work.

IV. PROPOSED SYSTEM

In the proxy Re-encryption key the 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. An encryption scheme is multiplicative homomorphic if it supports a group operation on encrypted plaintexts without decryption. The 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. A secret key is shared to key servers with a threshold value t. To decrypt for a set of k message symbols, each key server independently queries 2 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 cipher texts. 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.

V. PROXY RE-ENCRYPTION SCHEMES

In a proxy re-encryption scheme, a proxy server can transfer a ciphertext under a public key PKA to a new one under another public key PKB by using the re-

encryption key $RKA->B$. The server does not know the plaintext during transformation. Ateniese et al.[6] proposed some proxy re-encryption schemes and applied them to the sharing function of secure storage systems. 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. Our work further integrates encryption, re-encryption, and encoding such that storage robustness is strengthened.

Type-based proxy re-encryption schemes proposed by Tang [7] provide a better granularity on the granted right of a re-encryption key. A user can decide which type of messages and with whom he wants to share in this kind of proxy reencryption schemes. Key-private proxy re-encryption schemes are proposed by Ateniese et al. [8]. In a key-private proxy re-encryption scheme, given a re-encryption key, a proxy server cannot determine the identity of the recipient. This kind of proxy re-encryption schemes provides higher privacy guarantee against proxy servers. Although most proxy re-encryption schemes use pairing operations, there exist proxy re-encryption schemes without pairing [9].

VI. SYSTEM MODEL

As shown in Fig. 1, our system model consists of users, n storage servers SS_1; SS_2; . . . ; SSn, and m key servers KS_1;KS_2; . . . ; KSm. Storage servers provide storage services and key servers provide key management services. They work independently. Our distributed storage system consists of four phases: system setup, data storage, data forwarding, and data retrieval. These four phases are described as follows.

1) In the system setup phase, the system manager chooses system parameters and publishes them. Each user A is assigned a public-secret key pair $(PK_A; SK_A)$ User A distributes his secret key SK_A to key servers such that each key server KS_i holds a key share $SK_{A,i}$; $1 \leq i \leq m$. The key is shared with a threshold t.

2) In the data storage phase, user A encrypts his message M and dispatches it to and dispatches it to storage servers. A message M is decomposed into k blocks $m_1 m_2 \ldots\ldots m_k$ and has an identifier ID. User A encrypts each block m_i into a cipher text C_i and sends it to v randomly chosen storage servers. Upon receiving cipher texts from a user, each storage server linearly combines them with randomly chosen coefficients into a codeword symbol and stores it. Note that a storage server may receive less than k message blocks and we assume that all storage servers know the value k in advance.

3) 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 SK_A and B's public key is PK_B to compute a re-encryption key $RK_{A\to B}^{ID}$ and then sends $RK_{A\to B}^{ID}$ to all storage servers. Each storage server uses the re-encryption key to re-encrypt its codeword symbol for later retrieval requests by B. The re-encrypted codeword symbol is the combination of cipher texts under B's public key. In order to distinguish re-encrypted codeword symbols from intact ones, we call them original codeword symbols and re-encrypted codeword symbols, respectively.

4) In the data retrieval phase, user A requests to retrieve a message from storage servers. The message is either stored by him or forwarded to him. User A sends a retrieval request to key servers. Upon receiving the retrieval request and executing a proper authentication process with user A, each key server KS_i requests u randomly chosen storage servers to get codeword symbols and does partial decryption on the received codeword symbols by using the key share$SK_{A,i}$. Finally, user A combines the partially decrypted codeword symbols to obtain the original message M.

System recovering: When a storage server fails, a new one is added. The new storage server queries k available storage servers, linearly combines the received codeword symbols as a new one and stores it. The system is then recovered.

VII. 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, nontarget users, and up to $(t-1)$ key servers.

The attacker analyzes stored messages in storage servers, the secret keys of nontarget 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 attack[1] of the proxy re-encryption scheme in a threshold version, as shown in Fig. 2.

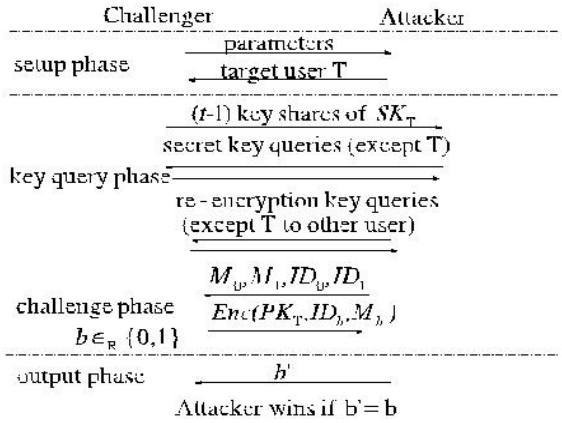

Fig. 2. The security game for the chosen plaintext attack.

The challenger C provides the system parameters. After the attacker A chooses a target user , the challenger gives him $(t-1)$ key shares of the secret key SK_T of the target user T to model $(t-1)$ compromised key servers. Then, the attacker can query secret keys of other users and all re-encryption keys except those from T to other users. This model compromised nontarget users and storage servers. In the challenge phase, the attacker chooses two messages M_0 and M_1 with the identifiers ID_0 and ID_1, respectively. The challenger throws a random coin b and encrypts the message M_b with T 's public key PK_T . After getting the ciphertext from the challenger, the attacker outputs a bit b' for guessing b. In this game, the attacker wins if and only if $b' = b$. The advantage of the attacker is defined as $[1/2 - \Pr[b' = b]]$.

A cloud storage system modeled in the above is secure if no probabilistic polynomial time attacker wins the game with a nonnegligible advantage. A secure cloud storage system implies that an unauthorized user or server cannot get the content of stored messages, and a storage server cannot generate re-encryption keys by himself. If a storage server can generate a re-encryption key from the target user to another user B, the attacker can win the security game by re-encrypting the ciphertext to B and decrypting the re-encrypted
ciphertext using the secret key SK_B. Therefore, this model addresses the security of data storage and data forwarding.

VIII. ADVANTAGES OF PROPOSED SYSTEM

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

IX. SECURITY ANALYSIS

The data confidentiality of our cloud storage system is guaranteed even if all storage servers, nontarget users, and up to $(t-1)$ key servers are compromised by the attacker. Recall the security game illustrated in Fig. 2.

X. CONCLUSION

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 performs partial decryption.

Our storage system and some newly proposed content addressable file systems and storage system [27], [28], [29] 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 Persis-tent 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 Fail-ures," Proc. Second Symp. Networked Systems Design and Implemen-tation (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] 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.

[7] 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.

[8] G. Ateniese, K. Benson, and S. Hohenberger, "Key-Private Proxy Re-Encryption," Proc. Topics in Cryptology (CT-RSA), pp. 279-294, 2009.

[9] J. Shao and Z. Cao, "CCA-Secure Proxy Re-Encryption without Pairings," Proc. 12th Int'l Conf. Practice and Theory in Public Key Cryptography (PKC), pp. 357-376, 2009.