International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)

International Conference on Information Systems and Computing (ICISC-2013), INDIA.

PROVIDING A SECURE DATA FORWARDING IN CLOUD STORAGE SYSTEM USING THRESHOLD PROXY RE-ENCRYPTION SCHEME

S.Poonkodi[1], V.Kavitha[2], K.Suresh[3]

[1,2]*Assistant Professor, Information Technology, Karpaga Vinayaga College of Engineering & Technology, Kanchipuram Dt, Tamil Nadu, India*

[3]*Assistant Professor, Computer Science & Engineering, KCG College of Technology, Chennai, Tamil Nadu, India*

Email Address: *rkpoonkodi@gmail.com, kavija2000@yahoo.com , sureshvk.2008@gmail.com*

Abstract

Cloud computing treats the resources on the Internet as a unified entity, cloud. A cloud storage system is considered as a large scale distributed storage system that consists of many independent storage servers. 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. General encryption schemes protect data confidentiality but also limit the functionality of the storage system because a few operations are supported over encrypted data. In this paper, we address the problem of forwarding data to another user by storage servers directly under the command of data owner. To provide a secure data forwarding, we propose a threshold proxy re-encryption scheme and integrate it with a secure decentralized code to form secure distributed storage system. The tight integration of encoding, encryption, and data forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding.

Keywords **— Cloud storage system; Decentralized code; threshold proxy re-encryption; 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 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]. To provide a data robustness 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 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 code- word symbol for the received message symbols and stores it. This finishes the encoding and storing process. The recovery process is the same.

II. EXISTING SYSTEM

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.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

2.1 Limitations of Existing System

There are three problems in the above straightforward integration of encryption and encoding.

1. The user has to do most computation and the communication traffic between the user and storage servers is high.
2. 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.
3. Finally, besides data storing and retrieving, it is hard for storage servers to directly support other functions

III. Proposed System

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

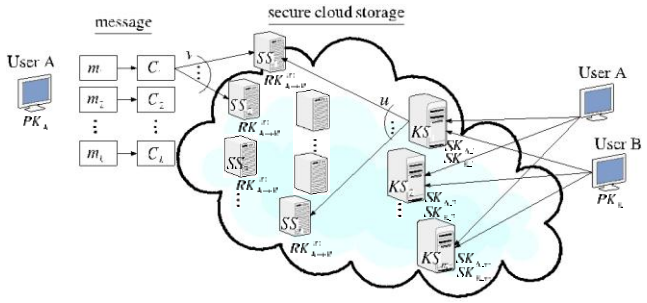

Fig.1. A general system model of our work.

IV. Scenario

We present the scenario of the storage system, the threat model that we consider for the confidentiality issue, and a discussion for a straightforward solution.

4.1 System Model

As shown in Fig. 1, our system model consists of users, n storage servers $SS_1, SS_2, \ldots, SS_n$, and m key servers $KS_1, KS_2, \ldots, KS_m$. 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.

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 $SK_{A,i}$, $1 \le i \le m$. 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 $m_1, m_2, \ldots, m_k$ and has an identifier ID. User A encrypts each block m_i into a ciphertext C_i and sends it to v randomly chosen storage servers. Upon receiving ciphertext 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.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

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 PK_B to compute a re-encryption key $RK_{A->B}^{ID}$ and then sends $RK_{A->B}^{ID}$ 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 ciphertext 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.

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.

V. A Secure Cloud Storage System With Secure Forwarding

As described in Section 4.1, there are four phases of our storage system.

System setup: The algorithm SetUp (1^τ) generates the system parameters μ. A user uses KeyGen(μ) to generate his public and secret key pair and ShareKeyGen(.) to share his secret key to a set of m key servers with a threshold t, where $k \leq t \leq m$. The user locally stores the third component of his secret key.

Data storage: When user A wants to store a message of k blocks m1, m2, . . .,mk with the identifier ID, he computes the identity token $\tau = h^{f(a3,ID)}$ and performs the encryption algorithm Enc(.) on τ and k blocks to get k original ciphertext C1, C2, . . . , Ck. An original ciphertext is indicated by a leading bit b = 0.

User A sends each ciphertext Ci to v randomly chosen storage servers. A storage server receives a set of original ciphertext with the same identity token τ from A. When a ciphertext Ci is not received, the storage server inserts Ci = (0, 1, τ, 1) to the set. The special format of (0, 1, τ, 1) is a mark for the absence of Ci. The storage server performs Encode (.)On the set of k ciphertext and stores the encoded result (codeword symbol).

Data forwarding: User A wants to forward a message to another user B. He needs the first component a1 of his secret key. If A does not possess a1, he queries key servers for key shares. When at least t key servers respond, A recovers the first component a1 of the secret key SKA via the KeyRecover(.) algorithm. Let the identifier of the message be ID. User A computes the re-encryption key $RK_{A->B}^{ID}$ via the ReKeyGen(.)Algorithm and securely sends the re-encryption key to each storage server. By using $RK_{A->B}^{ID}$, a storage server re-encrypts the original codeword symbol C" with the identifier ID into a re-encrypted codeword symbol C" via the ReEnc(.) algorithm such that C" is decrypt by using B's secret key. A re-encrypted codeword symbol is indicated by the leading bit b = 1. Let the public key PKB of user B be (g^{b1}, h^{b2}).

Data retrieval: There are two cases for the data retrieval phase. The first case is that a user A retrieves his own message. When user A wants to retrieve the message with the identifier ID, he informs all key servers with the identity token τ. A key server first retrieves original codeword symbols from u randomly chosen storage servers and then performs partial decryption ShareDec(.) on every retrieved original codeword symbol C". The result of partial decryption is called a partially decrypted codeword symbol. The key server sends the partially decrypted codeword symbols ζ and the coefficients to user A. After user A collects replies from atleast t key servers and at least k of them are originally from distinct storage servers, he executes Combine (.) on the t partially decrypted codeword 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. The collection and combining parts are the same as the first case except that key servers retrieve re-encrypted codeword symbols and perform partial decryption ShareDec(.) on re-encrypted codeword symbols.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

5.1 Analysis

We analyze storage and computation complexities, correctness, and security of our cloud storage system in this section. Let the bit-length of an element in the group G1 be l1 and G2 be l2. Let coefficients gi,j be randomly chosen from $\{0, 1\}^{l3}$.

Storage cost: To store a message of k blocks, a storage server SSj stores a codeword symbol (b, α_j, τ, γ_j) and the coefficient vector ($g_{1,j}$, $g_{2,j}$, . . . , $gk_{,j}$). They are total of ($1+2l_1 +l_2+kl_3$) bits, where α_j, τ ε G1 and γ_j ε G2. The average cost for a message bit stored in a storage server is ($1+2l_1 +l_2+kl_3$) / kl_2 bits, which is dominated by l_3/l_2 for a sufficiently large k. In practice, small coefficients, i.e., $l_3<<l_2$, reduce the storage cost in each storage server.

Computation cost: We measure the computation cost by the number of pairing operations, modular exponentiations in G1 and G2, modular multiplications in G1 and G2, and arithmetic operations over GF(p). These operations are denoted as Pairing, Exp1, Exp2, Mult1, Mult2, and Fp, respectively. Computing an Fp takes much less time than computing a Mult1 or a Mult2. The time of computing an Exp1 is 1.5[log p] times as much as the time of computing a Mult1, on average, (by using the square-and-multiply algorithm). Similarly, the time of computing a Exp2 is 1.5[log p] times as much as the time of computing a Mult2, on average.

Correctness: There are two cases for correctness. The owner A correctly retrieves his message and user B correctly retrieves a message forwarded to him. The correctness of encryption and decryption for A can be seen in (1). The correctness of re-encryption and decryption for B can be seen in (2). As long as at least k storage servers are available, a user can retrieve data with an overwhelming probability. Thus, our storage system tolerates n - k server failures.

The probability of a successful retrieval: A successful retrieval is an event that a user successfully retrieves all k blocks of a message no matter whether the message is owned by him or forwarded to him. The randomness comes from the random selection of storage servers in the data storage phase, the random coefficients chosen by storage servers, and the random selection of key servers in the data retrieval phase. The probability of a successful retrieval depends on (n, k, u, v) and all randomness.

Security: 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.

VI. Discussions And 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. 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 [7], [8], [9] are highly compatible. Our storage servers act as storage nodes in a content addressable storage system for storing content addressable blocks. Our key servers act as access nodes for providing a front-end layer such as a traditional file system interface. Further study on detailed cooperation is required.

REFERENCES

[1] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An Architecture for Global-Scale Persistent Storage," Proc. Ninth Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 190- 201, 2000.

[2] P. Druschel and A. Rowstron, "PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility," Proc. Eighth Workshop Hot Topics in Operating System (HotOS VIII), pp. 75-80, 2001.

[3] A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, and R. Wattenhofer, "Farsite: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment," Proc. Fifth Symp. Operating System Design and Implementation (OSDI), pp. 1-14, 2002.

[4] A. Haeberlen, A. Mislove, and P. Druschel, "Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures," Proc. Second Symp. Networked Systems Design and Implementation (NSDI), pp. 143-158, 2005.

[5] Z. Wilcox-O'Hearn and B. Warner, "Tahoe: The Least-Authority Filesystem," Proc. Fourth ACM Int'l Workshop Storage Security and Survivability (StorageSS), pp. 21-26, 2008.

[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] A. Shamir, "How to Share a Secret," ACM Comm., vol. 22, pp. 612- 613, 1979.

[8] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, and M. Welnicki, "Hydrastor: A Scalable Secondary Storage," Proc. Seventh Conf. File and Storage Technologies (FAST), pp. 197-210, 2009.

[9] C. Ungureanu, B. Atkin, A. Aranya, S. Gokhale, S. Rago, G. Calkowski, C. Dubnicki, and A. Bohra, "Hydrafs: A High-Throughput File System for the Hydrastor Content-Addressable Storage System," Proc. Eighth USENIX Conf. File and Storage Technologies (FAST), p. 17, 2010.

[10] W. Dong, F. Douglis, K. Li, H. Patterson, S. Reddy, and P. Shilane, "Tradeoffs in Scalable Data Routing for Deduplication Clusters," Proc. Ninth USENIX Conf. File and Storage Technologies (FAST), p. 2, 2011.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.