Ijesm
Consulting, help, relaxation

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES
&
MANAGEMENT

A Secure Erasure CBCS System with Data Forwarding Technique

K.Srinivasa Murthy[1], B.Ramesh Babu[2]

[1]PG Scholar, PBR Visvodaya Institute of Technology and Science,sreenu_n_you@yahoo.com

[2]Asst.Professor, PBR Visvodaya Institute of Technology and Science, brameshbabub07@gmail.com

Abstract—A cloud storage system, provides long-term storage services over the Internet. 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 over data confidentiality. General schemes protect data confidentiality, but limits the functionality of the storage system because a few operations are supported over the encrypted data. Constructing a secure storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority. We propose a proxy re-encryption scheme and integrate it with 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 to the sender's location . The basic 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, thus increasing the efficiency in storing the data with confidentiality and forwarding the data to another user by providing security.

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. 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 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

Since storing cryptographic keys in a single device is risky, a user distributes his cryptographic key to key servers. These key servers are highly protected by security mechanisms. 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.

Our contributions. Assume that there are n distributed storage servers and m key servers in the cloud storage system. A message is divided into k blocks and represented as a vector of k symbols. Our contributions are as follows:

1. 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. Our storage system allows the number of storage servers be much greater than the number of blocks of a message. In practical systems, the number of storage servers is much more than k. Nevertheless, the storage size in each storage server does not increase because each storage server stores an encoded result (a codeword symbol), which is a combination of encrypted message symbols.

2 Related Works

We briefly 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 (NAS) [7] and the Network File System (NFS) [8] 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 [1], [2], [9].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, 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 [10], [11], [12], [13], [5]. A message is encoded as a codeword, which is a vector of symbols, and each storage server stores a codeword symbol. To store a message of k blocks, each storage server linearly combines the blocks with randomly chosen coefficients and stores the codeword symbol and coefficients. To retrieve the message, a user queries k storage servers for the stored codeword symbols and coefficients and solves the linear system. The larger v is, the communication cost is higher and the successful retrieval probability is higher. The system has a light data confidentiality because an attacker can compromise k storage servers to get the message.

2.2 Proxy Re-Encryption Schemes

Proxy re-encryption schemes are proposed by Mambo and Okamoto [14] and Blaze et al. [15]. 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. Ateniese et al. [16] 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.

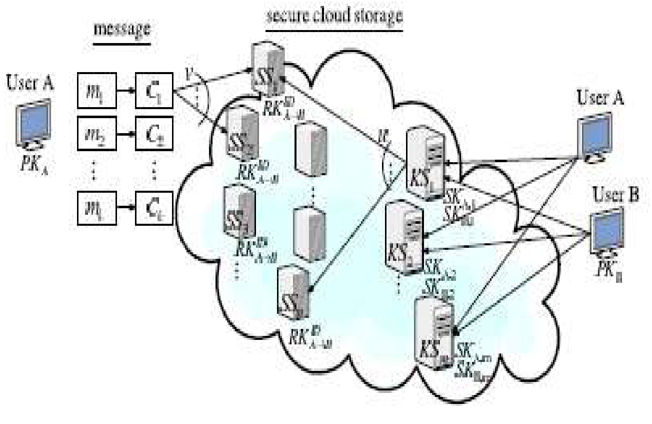


Fig. 1. System Architecture

When a user wants to share his messages, he sends a re-encryption key to the storage server. The storage server re-encrypts the 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.A user can decide which type of messages and with whom he wants to share in this kind of proxy re-encryption schemes.

3. 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 $SK_{A;i}$, $1 < i < m$. 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 $m_1;m_2;\ldots;m_k$ 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 ciphertexts 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.

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 $RK^{ID}{}_{A\text{-->}B}$ and then sends $RK^{ID}{}_{A\text{-->}B}$ 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 ciphertexts under B's public key.

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.

3.1 Problem Solution

A straightforward solution to supporting the data forwarding function in a distributed storage system is as follows: when the owner A wants to forward a message to user B, he downloads the encrypted message and decrypts it by using his secret key. He then encrypts the message by using B's public key and uploads the new ciphertext. When B wants to retrieve the forwarded message from A, he downloads the ciphertext and decrypts it by his secret key. The whole data forwarding process needs three communication rounds for A's downloading and uploading and B's downloading. The communication cost is linear in the length of the forwarded message. The computation cost is the decryption and encryption for the owner A, and the decryption for user B.

Proxy re-encryption schemes can significantly decrease communication and computation cost of the owner. In a proxy re-encryption scheme, the owner sends a re-encryption key to storage servers such that storage servers perform the re-encryption operation for him. Thus, the communication cost of the owner is independent of the length of forwarded message and the computation cost of re-encryption is taken care of by storage servers. Proxy re-encryption schemes significantly reduce the overhead of the data forwarding function in a secure storage system.

4 CONSTRUCTION OF SECURE CLOUD STORAGE SYSTEMS

Our approach. We use a threshold proxy re-encryption scheme with multiplicative homomorphic property. An encryption scheme is multiplicative homomorphic if it supports a group operation $\odot$ on encrypted plaintexts without decryption

$$D(SK, E(PK, m_1) \odot E(PK, m_2)) = m_1 \cdot m_2,$$

where E is the encryption function, D is the decryption function, and P_K-S_K is a pair of public key and secret key. Given two coefficients g_1 and g_2, two message symbols m_1 and m_2 can be encoded to a codeword symbol $m_1^{g_1} m_2^{g_2}$ in the encrypted form

$$C = E(PK, m_1)^{g_1} \odot E(PK, m_2)^{g_2} = E(PK, m_1^{g_1}\, m_2^{g_2}).$$

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. A secret key is shared to key servers with a threshold value t via the Shamir secret sharing scheme [26], where t k.

In our system, 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 ciphertexts.

4.1 A Secure Cloud Storage System with Secure Forwarding

As described in Section 3, there are four phases of our storage system.

System setup. The algorithm SetUp(1^τ) generates the system parametersμ. A user uses KeyGen(μ) to generate his public and secret key pair and ShareKeyGen$(\cdot)$ 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.

- SetUp(1^λ) .,
- KeyGen(μ) .,
- ShareKeyGen$(\mathrm{SK_A}, t, m)$.

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 $\tau = h^{f(a_3,\mathrm{ID})}$ and performs the encryption algorithm $\mathrm{Enc}(\cdot)$ on τ and k blocks to get k original ciphertexts C1; C2; . . . ; Ck.

- Enc$(\mathrm{PK_A}, \tau, m_1, m_2, \ldots, m_k)$.
- Encode$(C_1, C_2, \ldots, C_k)$.

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$(\cdot)$
- ReKeyGen$(\cdot)$
- ReEnc$(\cdot)$

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 tokenτ . A key server first retrieves original codeword symbols from u randomly chosen storage servers and then performs partial decryption using

- ShareDec$(\cdot)$
- Combine(.)

4.2 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 G_1be l_1 and G_2be l_2.

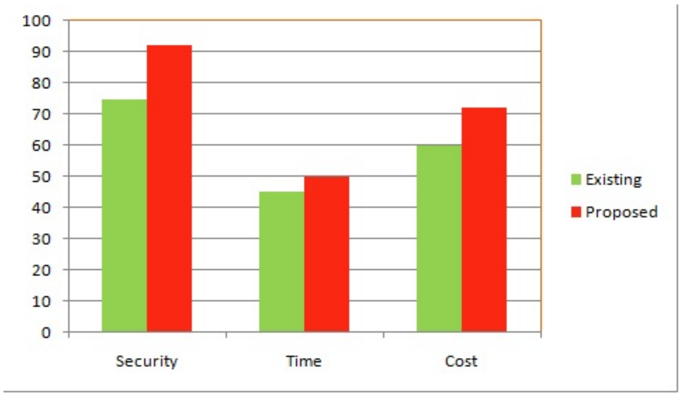
Storage cost. To store a message of k blocks, a storage server $\mathrm{SS_j}$ stores a codeword symbol and the coefficient vector, They are total of $(1 + 2l_1 + l_2 + kl_3)$ bits.

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., $l_3 \ll l_2$, reduce the storage cost in each storage server.

Computation cost. We measure the computation cost by the number of pairing operations, modular exponentiations in G_1 and G_2 modular multiplications in $G_1$and G_2, and arithmetic operations over $GF(p)$. These operations are denoted as Pairing, Exp_1, Exp_2, Mult_1, Mult_2, and F_p, respectively.

4.3 Experimental Results

The analysis of experimental results shows that our approach is practical in data storing and forwarding in distributed environment cloud storage. The empirical results gives a quick view w.r.t Security ,Cost and Time in existing and proposed systems.



5 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 reencryption scheme supports encoding, forwarding and partial decryptions 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.

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.

[16] 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.

[17] 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.

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

[19] 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.

[20] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable Data Possession at Untrusted Stores,” *Proc. 14th ACM Conf. Computer and Comm. Security (CCS),* pp. 598-609, 2007.