International Journal of Computers & Technology
Volume 4 No. 1, Jan-Feb, 2013, ISSN 2277-3061

Authentication Based Cloud Storage and Secure Data Forwarding

Rajasekaran.S[1], Kalifulla.Y[2], Murugesan.S[3], Ezhilvendan.M[4], Gunasekaran.J[5]
M.E-Network Engineering1, 3, 4, 5, Assistant Professor2
rajasekaran009@gmail.com(1), kalifulla@gmail.com(2), s.murugesan19@gmail.com(3)
vendannetwork@gmail.com(4), guna.vnb@gmail.com(5)
Veltech Multitech Dr.Rangarajan Dr.Sakunthala Engineering College, Avadi, Chennai-600 062

***Abstract*—A cloud storage system, consisting of a collection of storage servers, provides long-term storage services over the Internet. Storing data in a third party's cloud system causes serious concern over data confidentiality. General encryption schemes protect data confidentiality, but also limit the functionality of the storage system because a few operations are supported over 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 threshold 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. 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. 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.**

***Index Terms*—Decentralized erasure code, proxy re-encryption scheme, secure storage system, threshold cryptography.**

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. User just uses a

service 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]. 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 it's a codeword symbol 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 send 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. 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, 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 server 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 that consists of distributed storage servers and key servers. Since storing cryptographic keys in a single device is risky, a user distributes his

International Journal of Computers & Technology
Volume 4 No. 1, Jan-Feb, 2013, ISSN 2277-3061

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. This setting allows more flexible adjustment between the number of storage servers and robustness.

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. We present a general setting for the parameters of our secure cloud storage system. Our parameter setting of n=akc supersedes the previous one of n=ak√k1 where c≥1.5 and a>√2 [2]. Our result n=akc allows the number of storage servers be much greater than the number of blocks of a message.

II. RELATED WORKS

We briefly review distributed storage system, proxy re-encryption scheme, and integrity checking mechanisms.

A. Distributed Storage Systems

At the early years, the Network-Attached Storage (NAS) [2] and the Network File System (NFS) 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].

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 [3]. 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. 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.

B. Proxy Re-Encryption Schemes

Proxy re-encryption Scheme, a proxy server can transfer a cipher text under a public Key PKA to a new one under another public key PKB by using The re-encryption key RK A→B. The server does not know the Plaintext during transformation. In their work, message are first Encrypted by the owner and then stored in a storage server. When a user wants to share his messages, he 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 [4] provide a better granularity on the granted right of a re-encryption key.

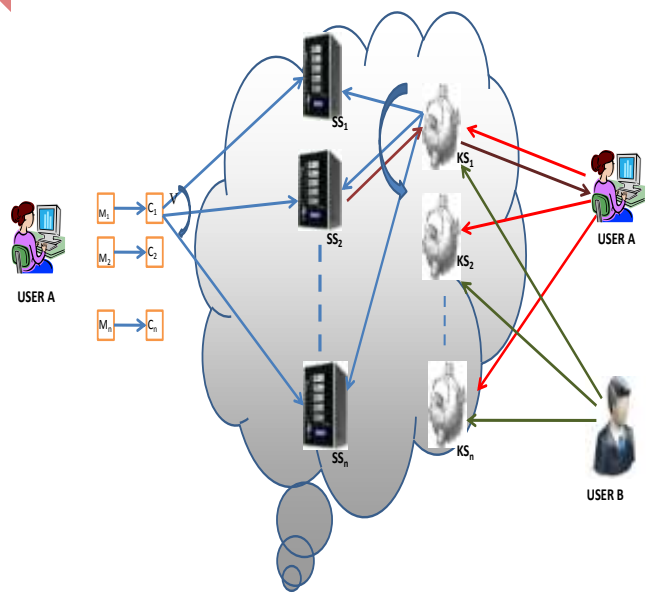


Fig 1: A general system model of our work

A user can decide which type of message and with whom he wants to share in this kind of proxy re-encryption schemes. Key

International Journal of Computers & Technology
Volume 4 No. 1, Jan-Feb, 2013, ISSN 2277-3061

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 server. Although most proxy re-encryption schemes use pairing operations, there exit proxy re-encryption schemes without pairing.

C. Integrity Checking Functionality

Another important functionality about cloud storage is the function of integrity checking. After a user stores data into the storage system, he no longer possesses the data at hand. The user may want to check whether the data are properly stored in storage servers.

III. SCENARIOS

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.

A. System Model

As shown in Fig 1, our system model consists of user, n storage servers SS1, SS2… SSn and m key servers KS1, KS2... KSm. Storage servers provide storage services and key servers provide key management service. 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 (PKA, SKA). User A distributes his secret key SKA to key servers such that each key server KSi holds a key share SKA, i, $1 \leq i \leq m$. 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 m1, m2..., mk and has an identifier ID. User A encrypts each block mi into a cipher text Ci 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.

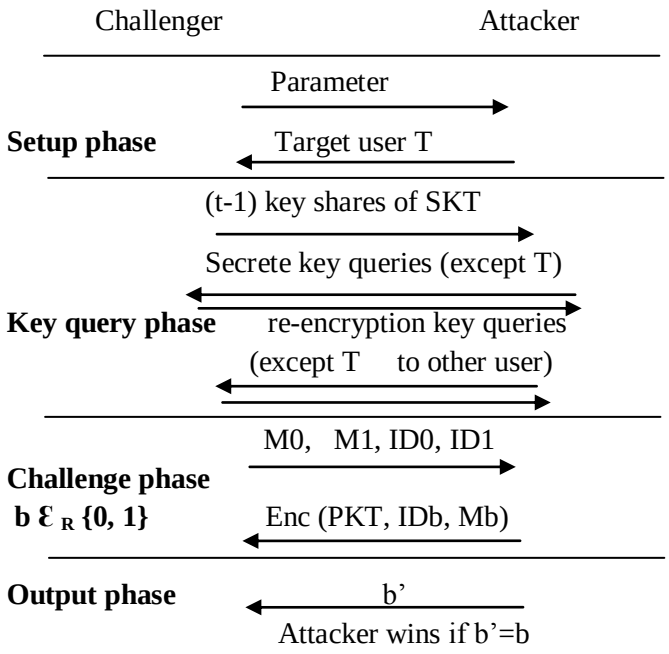
In the data forwarding phase, user A forwards his encrypted message with an identifier ID stored in storage servers to user B such that B can decrypt the forwarded message by his secret key. To do so, A uses his secret key SKA and B's public key PKB to compute a re-encryption key RKIDA→B to all storage 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 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.

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 KSi requests u randomly chosen storage servers to get codeword symbols and does partial decryption on the received codeword symbols by using the key share SKA, 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.

B. 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, non target users, and up to (t-1) key servers. The attacker analyzes stored messages in storage servers, the secret keys of non target 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 of the proxy Re-encryption scheme in a threshold version, as shown in Fig 2. A cloud storage system modeled in the above is secure if no probabilistic polynomial time attacker wins the game with a non negligible advantage.



International Journal of Computers & Technology
Volume 4 No. 1, Jan-Feb, 2013, ISSN 2277-3061

Fig 2: The security game for the chosen plaintext attack

C. A straightforward solution

A straightforward solution to supporting the data forwarding function in a distributed storage system is as follows: When the owner A wants to forwarding 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 cipher text. When B wants to retrieve the forwarded message from A, he downloads the cipher text 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 to storage servers such that storage server 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.

IV. Constructions of Secure Cloud Storage Systems

Before presenting our storage system, we briefly introduce the algebraic setting, the hardness assumption, and an erasure code over exponents, and our approach.

Bilinear map: Let G1 and G2 be cyclic multiplicative groups with a prime order p and g € G1 be a generator. A map ẽ: G1 × G1→G2 is a bilinear map if it is efficiently computable and has the properties of bilinearity and non degeneracy: for any x, y € Zp*, ẽ (gx, gy) = ẽ (g, g) xy and ẽ (g, g) is not the identity element in G2. Let Gen (1λ) be an algorithm generating (g, ẽ, G1, G2, p), where λ is the length of p. Let x €R X denote that x is randomly chosen from the set X.

Decisional bilinear Diffie-Hellman assumption: This assumption is that it is computationally infeasible to distinguish the distribution (g, gx, gy, gz, ẽ (g, g) xyz) and (g, gx, gy, gz, ẽ (g, g) r) where x, y, z, r €R Zp*. Formally, for any probabilistic polynomial time algorithm Ȧ, the following is negligible (in λ):

$$| \Pr [Ȧ (g, gx, gy, gz, Qb) = b: x, y, z, r €R Zp^*,$$

$$Q0 = ẽ (g, g) xyz; Q1 = ẽ (g, g) r; b €R \{0, 1\}]-1/2|.$$

Erasure coding over exponents: We consider that the message domain is the cyclic multiplicative group G2 described above. An encoder generates a generator matrix G = [gi, j] for 1≤ i ≤ k, 1≤ j ≤ n as follows: for each row, the encoder randomly selects an entry and randomly sets a value form Zp* to the entry. The encoder repeats this step v times with replacement for each row. An entry of a row can be selected multiple times but only set to one value. The value of the rest entries are set to 0. Let the message be (m1, m2….mk) € G2k. The encoding process is to generate (w1, w2…..wn) € G2k, where wj = m1gi, 1, m2gi, 2….mkgk, j for 1≤ j ≤ n. The first step of the decoding process is to compute the inverse of a k × k sub matrix K of G.

Our approach: We use a threshold proxy re-encryption scheme with multiplication homomorphism property. An encryption scheme is multiplicative homomorphism if it supports a group operation Θ on encrypted plaintexts without decryption

$$D (SK, E (PK, m1) Θ E (PK, m2)) = m1•m2,$$

Where E is the encryption function, D is the decryption function, and (PK, SK) is a pair of public key and secret key. Given two coefficients g1 and g2, two message symbols m1 and m2 can be encoded to a codeword symbol m1g1, m2g2 in the encrypted form

$$C=E (PK, m1) g1 Θ E (PK, m2) g2 = E (PK, m1g1 •m2g2).$$

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. 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 symbol.

A. 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 coefficient 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, αj, r, γj) and the coefficient vector (g1, j, g2, j,….,gk, j). There are total of (1+2l1 +l2+kl3) bits. The average cost for a message bit stored in a storage server is (1+2l1 +l2+kl3)/kl2 bits.

Computation Cost: We measure the computation cost by the number of pairing operation, modular exponentiations in G1 and G2, modular multiplications in G1 and G2, and arithmetic operation over GF (p). These operations are denoted as Pairing, Exp1, Exp2, Mult1, Mult2, and Fp, respectively. The cost is summarized in Table 1. Computing an Fp take much less time than computing a Mult1 or a Multi2. The time of computing an Exp1 is 1.5[log p] times as much as the time of computing a Multi1, on average, (by using the square and multiply algorithm). Similarly,

International Journal of Computers & Technology
Volume 4 No. 1, Jan-Feb, 2013, ISSN 2277-3061

the time of computing an Exp2 is 1.5[log p] times as much as the time of computing a Multi2, on average.

Table 1

The Computation Cost of Each Algorithm in Our Secure Cloud Storage System

Operation	Computation Cost
Enc	k Pairing + k Exp1 + k Mult2
Encode(for each storage server)	k Exp1 + Exp2 + (k-1) Mult1 + (k-1) Mult2
KeyRecover	O(t2) Fp
ReKeyGen	1 Exp1
ReEnc(for each storage server)	1 Pairing + 1 Mult2
ShareDec(for t key servers)	T Exp1
Combine	K Pairing + t Mult1 + (t-1) Exp1 + O(t2 + k3) Fp + k2 Exp2 + (k+1)k Mult2

- Pairing: a pairing computation of ĕ.
- Exp1 and Exp2: modular exponentiation computation in G1 and G2, respectively.
- Mult1 and Mult2: a modular multiplication computation in G1 and G2, respectively.
- Fp: an arithmetic operation in GF (p).

V. CONCLUSIONS

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 code 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 perform partial decryption.

Our storage system and some newly proposed content addressable file systems and storage system [5] 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] P.Druschel and A.Rowstron, "PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility," Proc. Eighth workshop Hot Topic in Operating System (Hot OS VIII), pp. 75-80, 2001.

[2] 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

[3] A.G. Dimakis, V. Prabhakaran,"Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks Through Decentralized Erasure Code," Proc. Fourth Int'l Symp. Information Processing in Sensor Network (IPSN), pp. 111-117, 2005.

[4] Q. Tang, "Type-Based Proxy Re-Encryption and Its Construction," Proc. Ninth Int'l Conf. Cryptography in India: Progress in Cryptography (INDOCRYPT), pp. 130-144, 2008.

[5] C. Dubinsky, L. Gryz, L. Heldt, W. Kilian,"Hydrastor: A Scalable Secondary Storage," Proc.seventh conf. File and Storage Technologies (FAST), pp.197-210, 2009.