IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 9, Issue 2 (Jan. - Feb. 2013), PP 22-27
www.iosrjournals.org

A Secure Code Based Cloud Storage System Using Proxy Re-Encryption Scheme in Cloud Computing

Priyadharshini. B.[1], Mrs. Carmel Mary Belinda[2], M. Ramesh Kumar[3]
[1](M. E. Student VelTech MultiTech Dr. Rangarajan Dr. Sakunthala Engineering College)
[2, 3] (Assistant professor VelTech MultiTech Dr. Rangarajan Dr. Sakunthala Engineering College)

Abstract: *Cloud computing is a model for enabling convenient, on demand network access to a shared pool of computing resources. The cloud storage system consists of a collection of storage servers and key servers. Storing data in a third party cloud system causes serious concern on data confidentiality, so a user divides the data into blocks, encrypts and stores them in various storage servers. The storage server encodes the data using erasure codeword symbol. When the sender wants to share his messages, he sends a re-encryption key to the storage server. The storage server re-encrypts the original codeword symbol into a re-encrypted codeword symbol. The re-encryption scheme is integrated with a secure decentralized erasure code so that a secure distributed system is designed. The proxy re-encryption scheme supports the encoding operations over encrypted messages as well as forwarding operations. The key server retrieves re-encrypted codeword symbols and performs partial decryption so that the receiver combines the blocks to retrieve the data.*

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

I. INTRODUCTION

Cloud computing is the use of computing resources that are delivered as a service over a network. 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. This paper focuses 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. 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. A decentralized erasure code is an erasure code that independently computes each codeword symbol for a message. 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.

Storing data in a third party's cloud system causes serious concern on data confidentiality. To provide strong confidentiality for messages in storage servers, a user encrypts 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. This addresses 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 function on behalf of the user. These key servers are highly protected by security mechanisms. We use 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. Our system meets the requirements that storage servers independently perform encoding and re-encryption.

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.
The contributions are:

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. 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.

II. RELATED WORK

We briefly review distributed storage systems, proxy re-encryption schemes and decentralized erasure code.

2.1 Distributed Storage Systems

The Network-Attached Storage (NAS) and the Network File System (NFS) provide storage devices over the network such that a user can access the storage devices via network connection. 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. One way to reduce the expansion rate is to use erasure codes to encode messages. 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. 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 system has light data confidentiality because an attacker can compromise k storage servers to get the message.

2.2 Proxy Re-Encryption Scheme

In a proxy re-encryption scheme, a proxy server can transfer a cipher text under a private key A to a new one under another public key B. The server does not know the plaintext during transformation. The data is first encrypted with a symmetric data encryption key and then stored in the cloud storage server. The cloud storage server uses a re-encryption algorithm to transfer the encrypted DEK into the format that can be decrypted by the recipient's private key. The recipient then can download the encrypted data from the cloud and use the DEK for decryption. A re-encryption key is generated from the data owner's private key and a recipient's public key.

A data owner may share different files with different recipient groups. Therefore, a recipient cannot read data for a group it does not belong to. The cloud, on the other hand, acts as an intermediate proxy. It cannot read the data as it cannot get DEKs. Thus the system has data confidentiality and supports the data forwarding function.

2.3 Decentralized Erasure Code

A decentralized erasure code is a random linear code with a sparse generator matrix. The generator matrix G constructed by an encoder is as follows: First, for each row, the encoder randomly marks an entry as 1 and repeats this process for an ln k/k times with replacement. Second, the encoder randomly sets a value from IF for each marked entry. This finishes the encoding process. A decoding is successful if and only if $k \times k$ submatrix formed by the k-chosen columns is invertible. Thus, the probability of a success decoding is the probability of the chosen sub matrix being invertible.

The owner randomly selects v servers with replacement and sends a copy of M_i to each of them. Each server randomly selects a coefficient for each received cipher text and performs a linear combination of all received cipher texts. Those coefficients chosen by a server form a column of the matrix and the result of the linear combination is a codeword element. Each server can perform the computation independently. This makes the code decentralized.

III. 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 attack1 of the proxy re-encryption scheme in a threshold version.

A cloud storage system modeled in the above is secure if no probabilistic polynomial time attacker wins the game with a non negligible 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 cipher text to B and decrypting the re-encrypted cipher text using the secret key SK_B. Therefore, this model addresses the security of data storage and data

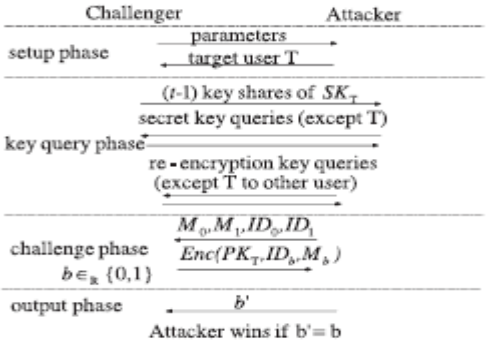

Fig.1.The Security game for the chosen plaintext attack

3.1 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 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 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 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

IV. SYSTEM MODEL

The system model consists of users, n storage servers SS_1; SS_2; . . . ; SS_n, and m key servers KS_1; KS_2; . . . ; 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 phase, data storage, data forwarding, and data retrieval. These four phases are described as follows.

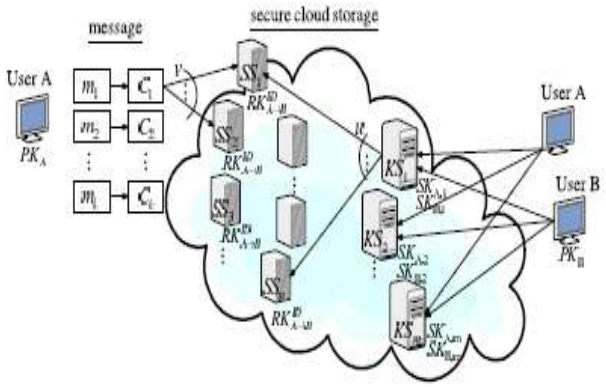

Fig.2. System Architecture

In the **system setup phase**, the system manager chooses system parameters and publishes them. Each user is assigned a public-secret key pair. User A distributes his secret key A to key servers such that each key server KS_i holds a key share. 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 mi 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.

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 and B's public key to compute a re-encryption key and then sends 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 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 recovery:

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. CONSTRUCTION OF SECURE CLOUD STORAGE SYSTEMS

We use a threshold proxy re-encryption scheme with multiplicative homomorphism property. An encryption scheme is multiplicative homomorphic if it supports a group operation where E is the encryption function, D is the decryption function and (PK, SK) is a pair of public key and secret key.

Thus, a multiplicative homomorphic encryption scheme supports the encoding operation over encrypted messages. We 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, codeword symbols are obtained from the partially decrypted cipher texts.

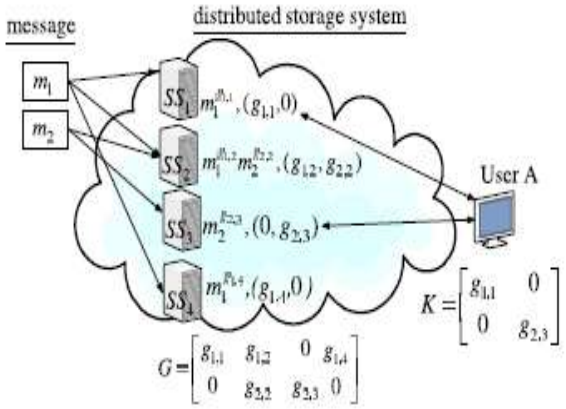

Fig.3. A Storage System with Random Linear Coding over exponents

5.1 A Secure Cloud Storage System with Secure Forwarding

We assume that there are n storage servers which store data and m key servers which own secret key shares and perform partial decryption. The owner shares the secret key x to m key servers with a threshold t. The storage process, re-encryption process and the retrieval process are described.

Storage process:

To store k messages, the storage process is as follows:

- Message encryption. The owner encrypts all k messages via the data encryption key with the identifier ID for the set of messages $M_1, M_2, \ldots, M_k$.

-Cipher text distribution. For each C_i , the owner randomly chooses v storage servers (with replacement) and sends each of them a copy of C_i.

-Decentralized Encoding. For all received cipher texts with the same message identifier h_{ID}, the storage server SS_j groups them as N_j. The storage server SS_j selects a random coefficient and forms a generator matrix of the decentralized erasure code.

Re-Encryption Process:

To forward a message, the re-encryption process is as follows:

-Key Recover. User A queries key servers for key shares. When at least t key servers respond, A recovers the first component a1 of the secret key SK_A via the Key Recover algorithm.

-Key Re-Generation. User A computes the re-encryption key algorithm and securely sends the re-encryption key to each storage server. By using $RK_{ID,}$ a storage server re-encrypts the original codeword symbol

Retrieval process:

To retrieve k messages, the retrieval process is as follows:

- Retrieval command. The owner sends a command to the m key servers with the message identifier h_{ID}.- Partial decryption. Each key server KS_i randomly queries u storage servers with the message identifier h_{ID} and obtains at most u stored data from the storage servers. The key server retrieves the re-encrypted codeword symbols from the storage server and performs partial decryption. Then, the key server KS_i performs ShareDec on each received cipher text by its secret key share to obtain a decryption share of the cipher text.

- Combining and decoding. The owner collects replies from at least 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 such that the original data is recovered.

5.2 Analysis

We analyze storage and computation complexities, correctness, and security of our cloud storage system in this section. The bit-length of an element in the group be G1 and G2.

Storage cost:

To store a message of k blocks, a storage server SS_j stores a codeword symbol and the coefficient vector .The average cost for a message stored in a storage server is bits, which is dominated for a sufficiently large value of k. In practice, small co-efficients, reduce the storage cost in each storage server. In practice, small coefficients reduce the storage cost in each storage server

Computation cost:

We measure the computation cost by the number of pairing operations, modular exponentiation in G_1 and G_2, modular multiplications in G_1 and G_2, and arithmetic operations over GF(p). The cost is summarized in Table 1.

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 re-encryption and decryption for B can be seen. 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.

Table 1
The Computation Cost of Each Algorithm in Our Secure Cloud Storage System

Operation	Computation cost
Enc	$k$pairing + k Exp_1 + k $Mult_2$
Encode (for each storage server)	k Exp_1 + k Exp_2 + (k-1) $Mult_1$ + (k-1) $Mult_2$
Key Recover	$O(t^2)$ F_p
ReKeyGen	1 Exp_1
ReEnc (for each storage server)	1 Pairing + 1 $Mult_2$
Share Dec(for t key servers)	t Exp_1
Combine	K Pairing + t $Mult_1$+(t-1) Exp_1+ $O(t^2 + k^2)$ F_p + k^2 Exp_2 +(k-1)k $Mult_2$

- Pairing: a pairing computation of e.
- Exp_1 and Exp_2: a modular exponentiation computation in G_1 and G_2, respectively.
- $Mult_1$ and $Mult_2$: a modular multiplication computation in G_1 and G_2, respectively.
- F_p: an arithmetic operation in *GF (p).*

The difference between our system model and the one in is that our system model has key servers. A single user queries k distinct storage servers to retrieve the data. On the other hand, each key server in our system independently queries u storage servers. The use of distributed key servers increases the level of key protection but makes the analysis harder. Our generalization of parameter setting allows the number of storage servers be much greater than the number of blocks of a message. It gives a better flexibility for adjustment between the number of storage servers and robustness.

Security: The data confidentiality of our cloud storage system is guaranteed even if all storage servers, non target users, and up to (t-1) key servers are compromised by the attacker.

VI. CONCLUSION

We consider a cloud storage system that consists of storage servers and key servers. Our system provides both the storage service and key management service. A secure cloud storage system is constructed using the threshold proxy re-encryption scheme that provides secure data storage and secure data forwarding functionality in a decentralized structure. The threshold proxy re-encryption scheme supports encoding, forwarding, and partial decryption operations in a distributed way. Our construction is fully decentralized with storage server performing encoding and re-encryption process and each key server perform partial decryption. 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.

REFERENCES

[1.] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, 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.] 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.] 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.
[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.