ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Accumulating Data into Cloud Storage System with Security Code-Based and Secure Data Forwarding

B.Mohan
M.Tech (CSE),
Department of Computer Science & Engineering, Nagole Institute of Technology & Science.

Dr. P. Venkateswarlu
Professor & HOD,
Department of Computer Science & Engineering, Nagole Institute of Technology & Science.

M. Naresh Choudary
Assistant Professor,
Department of Computer Science & Engineering, Nagole Institute of Technology & Science.

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, threshold cryptography, secure storage system.

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.

2. PROBLEM STATEMENT:

Data robustness is a major requirement for storage systems. There have been many proposals of storing data over storage servers. 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 trade-off 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 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. 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 code-word 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 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.

2.1 PROBLEM DEFINITION:

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. Moreover, we consider the system in a more general setting than previous works. This setting allows more flexible adjustment between the number of storage servers and robustness. 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 mes-sages and key servers independently perform partial decryption.

2 we present a general setting parameter for our secure cloud storage system. Our parameter setting of $n=akc$ supersedes the previous one one of $n=ak\sqrt{k}$, where $c \geq 1.5$ and $a>\sqrt{2}$ [6]. Our result $n=akc$ allows the number of storage severs be much greater than the number of blocks of a message. In practical systems, the number of storage servers is much more than k. The sacrifice is to slightly increase the total copies of an encrypted message symbol sent to storage servers. 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.

3.A SECURE DECENTRALIZED ERASURE CODE FOR DISTRIBUTED NETWORK STORAGE:

We consider the problem of constructing an erasure code for storage over a network when the data sources are distributed. Specifically, we assume that there are n storage nodes with limited memory and k < n sources generating the data. We want a data collector, who can appear anywhere in the network, to query any k storage nodes and be able to retrieve the data. We introduce Decentralized Erasure Codes, which are linear codes with a specific randomized structure inspired by network coding on random bipartite graphs. We show that decentralized erasure codes are optimally sparse, and lead to reduced communication, storage and computation cost over random linear coding.

3.1 PLUTUS:

Plutus is a cryptographic storage system that enables secure file sharing without placing much trust on the file servers. In particular, it makes novel use of cryptographic primitives to protect and share files. Plutus features highly scalable key management while allowing individual users to retain direct control over who gets access to their files. We explain the mechanisms in Plutus to reduce the number of cryptographic keys exchanged between users by using file groups, distinguish file read and write access, handle user revocation efficiently, and allow an un trusted server to authorize file writes. We have built a prototype of Plutus on OpenAFS. Measurements of this prototype show that Plutus achieves strong security with overhead comparable to systems that encrypt all network traffic.

3.2.TOTALRECALL:

Availability is a storage system property that is both highly desired and yet minimally engineered. While many systems provide mechanisms to improve availability– such as redundancy and failure recovery – how to best configure these mechanisms is typically left to the system manager. Unfortunately, few individuals have the skills to properly manage the trade-offs involved, let alone the time to adapt these decisions to changing conditions. Instead, most systems are configured statically and with only a cursory understanding of how the configuration will impact overall performance or availability. While this issue can be problematic even for individual storage arrays, it becomes increasingly important as systems are distributed – and absolutely critical for the wide area peer-to-peer storage infrastructures being explored. This paper describes the motivation, architecture and implementation for a new peer-to-peer storage system, called Total Recall that automates the task of availability management. In particular, the Total Recall system automatically measures and estimates the availability of its constituent host components predicts their future availability based on past behaviour, calculates the appropriate redundancy mechanisms and repair policies, and delivers user-specified availability while maximizing efficiency.

3.3.PAST:

This paper sketches the design of PAST, a large-scale, Internet-based, global storage utility that provides scalability, high availability, persistence and security. PAST is a peer-to-peer Internet application and is entirely self organizing. PAST nodes serve as access points for clients, participate in the routing of client requests, and contribute storage to the system. Nodes are not trusted, they may join the system at any time and may silently leave the system without warning. Yet, the system is able to provide strong assurances, efficient storage access, load balancing and scalability.

4. EXISTING SYSTEM:

In Existing System we use a straightforward integration method. In straightforward integration method Storing data in a third party's cloud system causes serious concern on data confidentiality. In order to provide strong confidentiality for messages in storage servers, a user can encrypt messages by a cryptographic method before applying an erasure code method to encode and store messages. When he wants to use a message, he needs to retrieve the Codeword symbols from storage servers, decode them, and then decrypt them by using cryptographic keys. General encryption schemes protect data confidentiality, but also limit the functionality of the storage system because a few operations are supported over encrypted data. A decentralized architecture for storage systems offers good scalability, because a storage server can join or leave without control of a central authority.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

DISADVATAGES OF EXISTING SYSTEM:

* The user can perform more computation and communication traffic between the user and storage servers is high.

* The user has to manage his cryptographic keys otherwise the security has to be broken.

* The data storing and retrieving, it is hard for storage servers to directly support other functions.

5. PROPOSED SYSTEM:

In our proposed system 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.

The distributed systems require independent servers to perform all operations. We propose a new threshold proxy re-encryption scheme and integrate it with a secure decentralized code to form a secure distributed storage system. The encryption scheme supports encoding operations over encrypted messages and forwarding operations over encrypted and encoded messages.

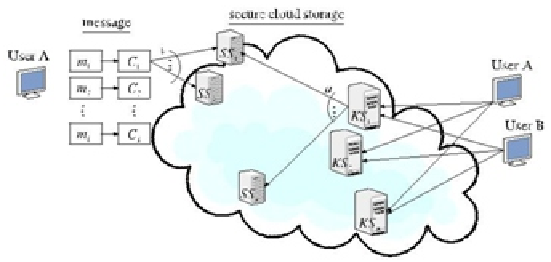
ADVANTAGES OF PROPOSED SYSTEM:

* Tight integration of encoding, encryption, and forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding.

* The storage servers independently perform encoding and re-encryption process and the key servers independently perform partial decryption process.

* More flexible adjustment between the number of storage servers and robustness.

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 robust-ness is strengthened.

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

* 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. The concept of provable data possession and the notion of proof of storage are proposed. Later, public auditability of stored data is addressed in. Nevertheless all of them consider the messages in the clear text form.

• 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.

• System Model

As shown in Fig. 1, our system model consists of users, n storage servers SS1; SS2; . . . ; SSn, and m key servers KS1; KS2; . . . ; KSm. Storage servers provide storage services and key servers provide key management services. They work independently. Our distributed storage system consists of four phases: system setup, data storage, data forwarding, and data retrieval. These four phases are described as follows. In the system setup phase, the system manager chooses system parameters and publishes them. Each user A is assigned a public-secret key pair ðPKA; SKAÞ. User A distributes his secret key SKA to key servers such that each key server KSi holds a key share SKA;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 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 and then sends 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. 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.

• 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 ðt _ 1Þ key servers. The attacker analyzes stored messages in storage servers, the secret keys of no 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.

6. FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

• ECONOMICAL FEASIBILITY

• ECHNICAL FEASIBILITY

• SOCIAL FEASIBILITY

6.1 ECONOMICAL FEASIBILITY:

This study is carried out to check the economic impact that the system will have on the organization.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

6.2 TECHNICAL FEASIBILITY:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

6.3 SOCIAL FEASIBILIT:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

MODULES:

Construction of Cloud Data Storage Module

• Data Encryption Module

• Data Forwarding Module

• Data Retrieval Module

MODULES DESCRIPTION:
Construction of Cloud Data Storage Module:

In Admin Module the admin can login to give his username and password. Then the server setup method can be opened. In server setup process the admin first set the remote servers IP-address for send that IP-address to the receiver. Then the server can skip the process to activate or Deactivate the process. For activating the process the storage server can display the IP-address. For De-activating the process the storage server cannot display the IP-address. These details can be viewed by clicking the key server. The activated IP-addresses are stored in available storage server. By clicking the available storage server button we can view the currently available IP-addresses.

Data Encryption Module:

In cloud login module the user can login his own details. If the user cannot have the account for that cloud system first the user can register his details for using and entering into the cloud system. The Registration process details are Username, E-mail, password, confirm password, date of birth, gender and also the location. After entering the registration process the details can be stored in database of the cloud system. Then the user has to login to give his corrected username and password the code has to be send his/her E-mail. Then the user will go to open his account and view the code that can be generated from the cloud system.

In Upload Module the new folder can be create for storing the files. In folder creation process the cloud system may ask one question for that user. The user should answer the question and must remember that answer for further usage. Then enter the folder name for create the folder for that user. In file upload process the user has to choose one file from browsing the system and enter the upload option. Now, the server from the cloud can give the encrypted form of the uploading file.

Data Forwarding Module:

In forward module first we can see the storage details for the uploaded files. When click the storage details option we can see the file name, question, answer, folder name, forward value (true or false), forward E-mail. If the forward column display the forwarded value is true the user cannot forward to another person If the forward column display the forwarded value is false the user can forward the file into another person..

In file forward processes contains the selected file name, E-mail address of the forwarder and enter the code to the forwarder. Now, another user can check his account properly and view the code forwarded from the previous user. Then the current user has login to the cloud system and to check the receive details. In receive details the forwarded file is present then the user will go to the download process.

Data Retrieval Module:

In Download module contains the following details. There are username and file name. First, the server process can be run which means the server can be connected with its particular client. Now, the client has to download the file to download the file key. In file key downloading process the fields are username, filename, question, answer and the code. Now clicking the download option the client can view the encrypted key. Then using that key the client can view the file and use that file appropriately.

SCOPE OF THE PROJECT:

* Designing a cloud storage system for robustness, confidentiality and functionality. The proxy re-encryption scheme supports encoding operations over encrypted messages as well as forwarding operations over encoded and encrypted messages. 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.

* 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. A decentralized erasure code is suitable for use in a distributed storage system.

* A storage server failure is modeled as an erasure error of the stored codeword symbol.

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.

FUTURE ENHANCEMENT:

In our system 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.

The distributed systems require independent servers to perform all operations. We propose a new threshold proxy re-encryption scheme and integrate it with a secure decentralized code to form a secure distributed storage system. The encryption scheme supports encoding operations over encrypted messages and forwarding operations over encrypted and encoded messages.

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 and erasure codes over exponents. The threshold proxy re-encryption scheme supports encoding, forwarding, and partial decryption operations in a distributed way. To decrypt a message of k blocks that are encrypted and encoded to n codeword symbols, each key server only has to partially decrypt two codeword symbols in our system. By using the threshold proxy re-encryption scheme, we present a secure cloud storage system that provides secure data storage and secure data forwarding functionality in a decentralized structure. Moreover, each storage server independently performs encoding and re-encryption and each key server independently perform partial decryption.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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.