International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/IJSRCSEIT

Secured Erasure Code For Distributed Storage System

*[1]G Geetha Devi, [2]A. Harika, [3]M. Kalyani Rao
[1]Assistant Professor, Department of Information Technology, Bhoj Reddy Engineering College for Women, Hyderabad, India
[2,3]Students, Department of Information Technology, Bhoj Reddy Engineering College for Women, Hyderabad, India

ARTICLEINFO

Article History:

Accepted: 10 April 2023
Published: 30 April 2023

Publication Issue
Volume 9, Issue 2
March-April-2023

Page Number
661-665

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.

Keywords: Encryption, Threshold Proxy Re-encryption, Severs, Cloud storage, Storage sever, Decentralized

I. INTRODUCTION

This paper describes Farsite, a serverless distributed file system that logically functions as a centralized file server but whose physical realization is dispersed among a network of untrusted desktop workstations. Farsite is intended to provide both the benefits of a central file s access, and reliable data storage) and the benefits of local desktop file systems (low cost, privacy from nosy sysadmins, and resistance to geographically localized faults). Farsite replaces the physical security of a server in a locked room with the virtual security of

cryptography, randomized replication, and Byzantine fault-tolerance Farsite is designed to support typical desktop file-I/O workloads in academic and corporate environment. 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 designing Farsite, our goal has been to harness the collective resources of loosely coupled, insecure, and unreliable machines to provide logically centralized, secure, and reliable file-storage service. Our system protects and preserves file data and directory metadata primarily through the techniques of cryptography and replication. Since file data is large and opaque to the system, the techniques of encryption, one-way hashing, and raw replication provide means to ensure its privacy, integrity, and durability, respectively. By contrast, directory metadata is relatively small, but it must be comprehensible and revisable directly by the system; therefore, it is maintained by Byzantine replicated state-machines [8] and specialized cryptographic techniques that permit metadata syntax enforcement without compromising privacy. One of Farsite's key design objectives is to provide the benefits of Byzantine agreement in the common case, by using signed and dated certificates to cache the authorization granted through Byzantine operations. Both Farsite's intended workload and its expected machine characteristics are those typically observed on desktop machines in academic and corporate settings. These workloads exhibit high access locality, a low persistent update rate, and a pattern of read/write sharing that is usually sequential and rarely concurrent. The expected machine characteristics include a high fail-stop rate (often just a user turning a machine off for a while) [6] and a low but significant rate of malicious or opportunistic subversion. In our

design, analysis, evaluation, and discussion, we focus on this environment, but we note that corporate administrators might choose to supplement Farsite's reliability and security by adding user less machines to the system or even running entirely on machines in locked rooms. Farsite requires no central administration beyond that needed to initially configure a minimal system and to authenticate new users and machines as they join the system. Administration is mainly an issue of signing certificates: Machine certificates bind machines to their public keys; user certificates bind users to their public keys; and namespace certificates bind namespace roots to their managing machines. Beyond initially signing the namespace certificate and subsequently signing certificates for new machines and users, no effort is required from a central administrator.

II. RELATED WORK

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.

III.PROPOSED SYSTEM

Our system model consists of users, n storage servers S1;S2;.....;Sn and m key servers K1;K2;.....;Km. Storage servers offers storage services and key servers provide key management services. They work independently. Our distributed storage system comprises 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 selects the system parameters and publishes them. Every user A is assigned a public-secret key pair PUKA; SEKA. User A distributes his secret key SEKA to key servers such that each key server Ki holds a key share SEKA. The key is shared with a threshold t. In 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 cipher text Ci and sends it to v randomly chosen storage servers. Upon receiving cipher texts from a user, each storage server linearly incorporates them with randomly chosen coefficients into a code word symbol and stores it.

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, A forwards his encrypted message with identifier ID stored in storage servers to user B so that B can decrypt the forwarded message by his secret key, To achieve this, A uses his secret key (SEKA) and B's public key (PUKB) to compute a re-encryption key RKID A!B and then sends this key to all storage servers.

Upon receiving the key, each storage server uses the re-encryption key to re-encrypt its code word symbol. The re-encrypted code word symbol is the combination of cipher texts under B's public key PUKB.

To distinguish re-encrypted code word symbols from untouched ones, we call them original code word symbols and re-encrypted code word symbols, respectively.

In the data retrieval phase, user A requests to fetch a message from storage servers. The message of user is either stored by him or forwarded to him. User A sends a retrieval request to key servers. Upon receiving the request and executing a appropriate authentication process with user A, each key server Ki requests randomly chosen storage servers to get codeword symbols and performs partial decryption on the received codeword symbols by using the key share SEKA;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 code word symbols as a new one and stores it. The system is then recovered.

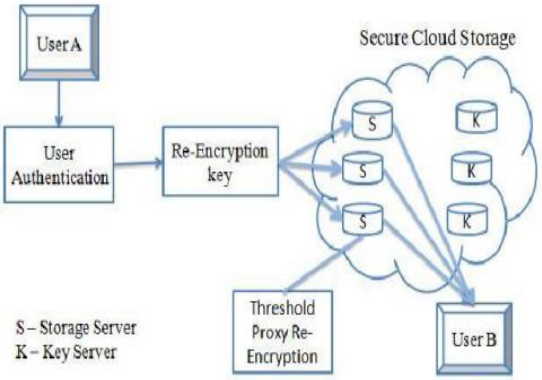

Fig 1: Data Forwarding

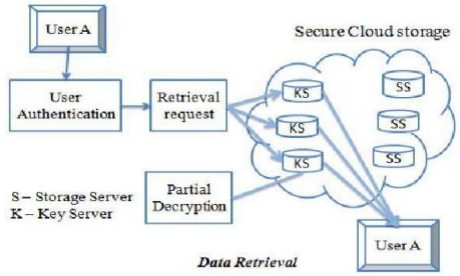

Fig 2: Data Retrieval

IV. RESULTS AND DISCUSSION

Many features like De-duplication, Compression, Thin Provisioning, Snapshots, Clones and RAID were implemented to make storage more efficient. In case of RAID, disk drives have become larger but their reliability has stayed the same. Recovering a failed disc takes more time; it is reconstructed using RAID parity information. There is also increasing probability of a second disk failure or other errors before reconstruction can complete. It's not uncommon for a 4TB disk to take days to rebuild. Implementation of Erasure Coding overcomes works by creating a mathematical function around a data set such that if a member of the data set is lost, the lost data can be recovered easily from the rest of the members of the set. Common industry uses Erasure coding for storage efficiency.

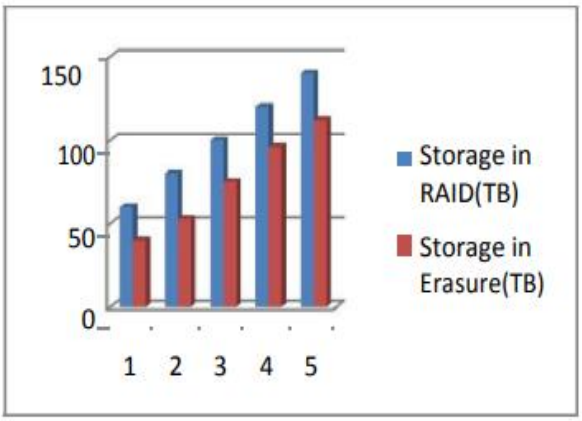

Fig 3: Graph

V. CONCLUSION AND FUTURE WORK

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 decentralized erasure codes. The threshold proxy reencryption scheme enhances encoding, forwarding, and partial decryption operations in a distributed way. To decrypt a message that are encrypted and encoded to 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 by erasure code mechanism with secure data forwarding functionality. Each storage server independently performs encoding and re-encryption

and partial decryption is performed by independent key servers. Our storage system and some newly proposed content addressable file systems and storage system are highly compatible. The storage servers which are storage nodes in a content addressable storage system are for storing content addressable blocks. Our key servers act as access nodes for providing a front-end layer such as traditional file system interface. We further enhance more flexibility in the implementation of Erasure Coding mechanism.

VI. REFERENCES

[1]. J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gumadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells and B. Zhao," Oceanstore: An Architecture for Global-Scale Persistent Storage," Proc. NinthInt' 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 Hottopics in Operating System (HotOS VIII), pp. 75-80,2001.

[3]. 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

[4]. Haeberlen, A. Mislove, and P. Druschel, "Glacier: HighlyDurable, 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 LeastAuthority 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

Cite this article as :

G Geetha Devi, A. Harika, M. Kalyani Rao, "Secured Erasure Code For Distributed Storage System", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 9, Issue 2, pp.661-665, March-April-2023.