ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

IJARCCE

nCORETech

LBS College of Engineering, Kasaragod

Vol. 5, Special Issue 1, Feburary 2016

Secured Self Destructing Data System Based on Active Storage Framework

Swathi J[1], Anagha Achuthan[2]

Student, M Tech, Computer Science, Malabar Institute of Technology, Kannur, India [1]

Assistant Professor, Computer Science, Malabar Institute of Technology, Kannur, India [2]

Abstract: Self-destructing data mainly aims at protecting the user data's privacy. Personal data stored in the Cloud may contain account numbers, passwords, notes, and other important information. These data are cached, copied, and archived by Cloud Service Providers, often without users authorization and control. All the data and their copies become destructed or unreadable after a user-specified time, without any user intervention. In addition, the decryption key is destructed after the user-specified time. In this paper, we present SeDas, a system that meets this challenge through a novel integration of symmetric key techniques with active storage techniques by successfully deleting even if the server goes down. Compared to the system without self-destructing data mechanism, throughput for uploading and downloading with the proposed SeDas acceptably decreases by less than 72%, while latency for upload/download operations with self-destructing data mechanism increases by less than 60%.

Keywords: Active storage, Cloud computing, Self destruction data system.

I. INTRODUCTION

With the development of cloud computing and popularization of mobile internet cloud services are becoming more and more important in peoples life. People are more or less requested to submit or post some personal private information to the cloud by the internet. When people do this, they subjectively hope that service providers will provide security policy to their data from leaking to the other people. As people rely more and more on the internet and cloud technology, security of their privacy takes more and more risk. Most of the individuals prefer using the internet as the primary medium to transfer data from one end to another across the internet. There are many possible ways to transmit data using the internet like: via e-mails, sending text and images, etc. However, one of the main problems with sending data over the Internet is the security and authenticity. Data security basically means protection of data from unauthorized users or attackers. No one can access the content without knowing the encryption key. On the one hand, when data is being processed, transformed and stored by current system or network, systems or network must cache , copy or archive it because these copies are essential for systems and network. However, people have no knowledge about these copies and cannot control them, thus leaking the privacy of these copies to the public. On the other hand, their privacy can also be leaked attributing to cloud service providers negligence, hackers intrusion or some legal As people rely more and more on the internet and cloud technology, security of their privacy takes more and more risk. Most of the individuals prefer using the internet as the primary medium to transfer data from one end to another across the internet. There are many possible ways to transmit data using the internet like: via e-mails, sending text and images, etc. However, one of the main problems with sending data over the Internet is the security and authenticity. Data security basically means protection of data from unauthorized users or attackers. These problems present formidable challenges to protect people's privacy in the cloud environment. Earlier many systems were implemented for sharing and protecting privacy. Vanish [4] provides a replacement plan for sharing and protecting privacy. Within the Vanish system, a secret is divided and stored in a P2P system with distributed hash tables (DHTs). With connection and exiting of the P2P node, the system will maintain secret keys. Consistent with characteristics of P2P, after eight hours the DHT can refresh each node. With Shamir Secret Sharing algorithm [5], once one cannot get enough components of a key, he won't decipher knowledge encrypted with this key, which suggests the secret is destroyed. Some special attacks to characteristics of P2P are challenges of Vanish [3], uncontrolled in how long the key will survive is additionally one in every of the disadvantages for Vanish. Aiming at many disadvantages in the earlier systems we proposed a system self-destructing data system, or SeDas, which is based on an active storage framework. Based on active storage framework, we use object based storage interface to store and manage the equal divided key. We implemented a proof of concept SeDas prototype. The SeDas system defines two new modules, a self-destruct method object that is associated with each secret key part and survival time parameter for each secret key part. SeDas supports security erasing files and random encryption keys stored in hard disk drive and solid state drive respectively. In this case, SeDas can meet the requirements of self destructing data with controllable survival time while users can use this system as a general object storage system. Here we focus on the key distribution algorithm, Shamir's algorithm, which is used as the core algorithm to implement client (users) distributing keys in the object storage system. We use these methods to implement a safety destruct with equal divided key. Through functionality and security properties evaluation of SeDas prototype, the results demonstrate that SeDas is practical to use and meets all the privacy preserving goals. The prototype imposes reasonably low run time overhead.

II. DATA SELF DESTRUCT

The self-destructing information system within the Cloud setting ought to meet the subsequent requirements:
1. How to destruct all copies of the data.
2. No explicit delete actions by the user, or any third-party storing that data.
3. No compelling reason to adjust any of the saved or documented copies of that data.
4. No use of secure hardware but support to completely erase data in HDD and SSD, respectively.

III. OBJECT BASED STORAGE

Objects are primitive units of storage that can be directly accessed without passing through a server.Object-based storage (OBS) uses an object-based storage device as the underlying storage device. Devices that store objects are referred are called as object storage devices (OSD) [2]. Object based storage offers great advancement in both storage devices as well as applications by increasing the functionalities of storage devices which seems to be far better compared to block based storage. Each OSD consists of a CPU, network interface, ROM, RAM, and storage device (disk or RAID subsystem) and exports a high-level data object abstraction on the top of device block read/write interface. With the emergence of object-based interface, storage devices can take advantage of the expressive interface to achieve some intelligent collaborations between the application servers and the storage devices. A storage object can be a file consisting of a set of ordered logical data blocks, or a database containing many files, or just a single application record such as a database record of one transaction. As it is easy to store and process data in object storage devices (OSD), people add more features in it that made these type of storage intelligent referred as —Intelligent storage or —Active storage [5]-[7].

IV. DESIGN AND IMPLEMTATION OF SEDAS

A. The SeDas architecture

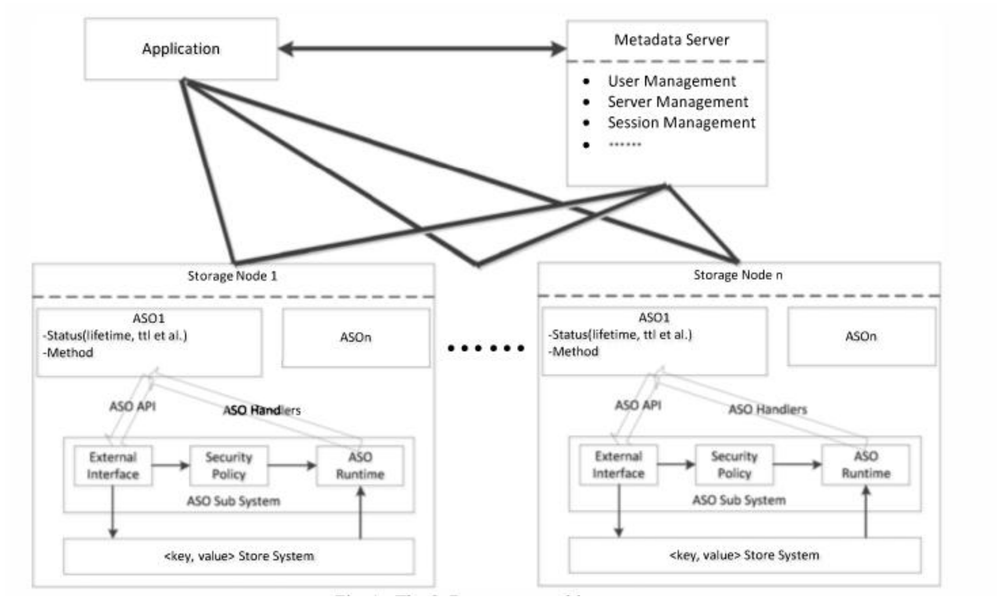

Fig. 1. : The SeDas system Architecture

Fig. 1 shows the architecture of SeDas. Three parties based on the active storage framework are:

1. Metadata server(MDS): MDS is responsible for user management, server management, file metadata management and session management.
2. Application node: The application node is a client to use storage service of the SeDas.
3. Storage node: Each storage node is an OSD. It contains two core subsystems: key value store subsystem and active storage object (ASO) runtime subsystem. The key value store subsystem that is based on the object storage component is used for managing objects stored in storage node: lookup object, read/write object and so on. The object ID is used as a key. The associated data and attribute are stored as values. The ASO runtime subsystem based on the active storage agent module in the object-based storage system was used to process active storage request from users and manage method objects and policy objects.

B. Active storage object

An active storage object derives from a user object and has a time-to-live (ttl) value property. The ttl value is used to trigger the self-destruct operation. The ttl value of user object is infinite so that a user object will not be deleted until a user manually deletes it. The ttl value of an active storage object is limited so when the value of the associated policy object is true an active object will be deleted. Interfaces extended by ActiveStorageObject class are used to manage ttl value. The create member function needs another argument for ttl. If the argument is 1, UserObject:: create will be called to create a user object, else, the ActiveStorageObject::create will call UserObject::create first and associate it with the self-destruct method object and a self-destruct policy object with the ttl value. The getTTL member function is based on the read_attr function and returns the ttl value of an active storage object. The setTTL, addTime and decTime member function is based on the write_attr function and can be used to modify the ttl value.

C.Self -destruct method object

Generally, kernel code can be executed efficiently. A service method should be implemented in user space with these following considerations. Many libraries such as libc, can be used by code in user space but not in kernel space. Mature tools can be used to develop software in user space. It is much safer to debug code in user space than in kernel space. A service method needs a long time to process a complicated task, so implementing code of a service method in user space can take advantage of performance of the system. The system might crash with an error in kernel code, but this will not happen if the error occurs in code of user space. A self-destruct method object is a service method. It needs three arguments. The lun argument specifies the device, the pid argument specifies the partition and the obj_ id argument specifies the object to be destructed.

D. Key Sharing in storage node

The client must first register itself with the server after which storage node also continues the registration with the metadata server. As a part of the client Advanced Encryption algorithm (AES) is used to generate keys for encryption and decryption of the file before uploading or downloading it .Once the keys are generated it has to be shared between the user application and the metadata server using Shamir Secret Sharing technique which enables distribution of data to N nodes.

E. Data process

Here users applications should implement logic of data process and act as a client node to use this SeDas system. Two different logics are: uploading and downloading.

(1) Uploading file process: When a user uploads a file to a storage system and stores his key in this SeDas system, he should specify the file, the ttl and key as the arguments for the uploading procedure. The encrypt procedure uses a common encrypt algorithm or a user-defined encrypt algorithm. After uploading data to storage server, key shares generated by Shamir Secret Sharing algorithm will be used to create active storage object (ASO) in storage node in the SeDas system. So, here we can upload image files to a storage system and stores his key in this SeDas system so at first the image files must be converted into bit streams and then the encrypt procedure uses a common encrypt algorithm or user defined encrypt algorithm.

(2) Downloading file process: Any user who has relevant permission can download data stored in the data storage system. The data must be decrypted before use. The whole logic is implemented in code of user's application.

Before decrypting, client should try to get key shares from storage nodes in the SeDas system. If the self-destruct operation hasn't been triggered, the client can get enough key shares to reconstruct the key successfully. If the associated ASO of the key has been destructed , the client cannot reconstruct the key so he only read encrypted data.

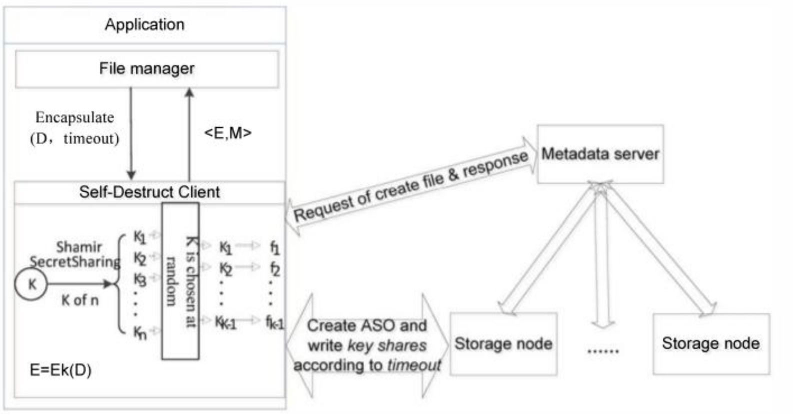

Fig. 2: Uploading file process

F. Data security erasing in disk
We must secure delete sensitive data and reduce the negative impact of OSD performance due to deleting operation. The proportion of required secure deletion of all the files is not great,so if this part of the file update operation changes, then the OSD performance will be impacted greatly.Our implentation method as follows:
(1) The system pre-specifies a directory in a special area to store sensitive files.
(2) Monitor the file allocation table, and acquire and maintain a list of all sensitive documents, the logical block address(LBA).
(3) OSD internal synchronization maintains the list of LBA.
(4) For ordinary LBA, the system uses the regular update method.
(5) By calling of ordinary data erasure API, we can safely delete sensitive files of the specified directory.

Advantages of this scheme:
(1) This scheme provides for user-defined authorization period and ensures that the sensitive data cannot be read prior to the release time or after its expiration.
(2) It is able to implement fine-grained access control during authorization period and it also does not require any human intervention to make the sensitive data to get self-destructed after expiration time.
(3) It does not require the ideal assumption that ‘No attacks on the VDO(Vanishing Data Object) before it expires’.

V. PROPOSED SYSTEM

People are more or less requested to submit or post some personal private information to the Cloud by the Internet. As people rely more and more on the Internet and Cloud technology, security of their privacy takes more and more risks. Most of the individuals prefer using the internet as the primary medium to transfer data from one end to another across the internet. In the proposed system if the data is to be deleted from the node at a particular instant of time and if any one of the server goes down at that time then the data at that node will not be deleted until the server is up. The problem here is that the data remains in the node and cannot be deleted. As the user data is stored in different nodes in cloud system. And it will be removed or destructed automatically when user time limit occurred. For the self destruction we implemented a process with thread and time comparing process. If one node failed or down, then the user data in the node will not be destructed. So here we are adding an additional background process to resolve the problem mentioned above. After node is up, each node in the cloud will run the background process. In this process the node checks undeleted file and find the time limit of that files which the user has given and invoke a self destruction process for the files which has past time or previous time than the server/cloud node current time. For this purpose, user will not have an additional attention. All these process will happen automatically without any user interaction.

VI. CONCLUSION

Data privacy has become increasingly important in cloud environment. This paper introduced a new approach for protecting data privacy from attackers who retroactively obtain, through legal or other means, a user's stored data and private decryption keys. A novel aspect of our approach is the leveraging of the essential properties of active storage framework based on T10 OSD standard. We demonstrate the feasibility of our approach by presenting SeDas, a proof of concept prototype based on object-based storage techniques. SeDas causes sensitive information, such as account

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

IJARCCE

nCORETech

LBS College of Engineering, Kasaragod

Vol. 5, Special Issue 1, Feburary 2016

numbers, passwords and notes to irreversibly self-destruct, without any action on the user's part. The fixed data timeout and large replication factor present challenges for a self-destruction data system. Proposed approach is using active storage framework for maximum advantage and it is expected that after successful implementation, performance should be more than existing systems. With this privacy can be achieved in a public network.

VII. FUTURE WORK

As a part of future work, the users are provided with still more advanced features for managing the files on the cloud server even after destruction. Once the time to live factor expires the data with all the copies are destroyed. Sometimes the users are even forced to enter into situations where the files might be in need for their personal verification. In order to make this happen the file has to be recovered back. Though it is tedious to find the data back, with the help of some strong security procedures this could be hopefully made more easy and enhanced by making cloud services ahead in the near future.

ACKNOWLEDGMENT

We wish to thank the Department of Computer Engineering also we are thankful of Project Lab for providing all resources and their valuable support.

REFERENCES

[1] Linfang Zeng, Shibin Chen, Qingsong Wei and Dan Feng," SeDas: A Self destructing Data System Based onAcive Storage Framework".
[2] M. Mesnier, G. Ganger, and E. Riedel, "Object based storage," IEEE Communications Magazine, vol. 41, no. 8, pp.84, August 2003.
[3] R. Weber, Information Technology SCSI Object Based Storage Device Commands (OSD) , Technical Committee T10, INCITS Std., Rev. 5, Jan. 2009.
[4] S. Wolchok,O. S. Ho_man, and N. Henninger, Defeating vanish with low cost sybil attacks against large dhts , Proc. of the network and distributed system security Symposium, 2010.
[5] A. Shamir," How to share a Secret," communications of the ACM, vol.22,no.11,pp. 612,1979.
[6] P. Gutmann, " Secure deletion of data from magnetic and solid state memory "Proc. of the conference on USENIX Security Symbosium, focusing on Applications of Cryptography, Berkeley, CA, USA, pp.8, 1996.
[7] L. Xu,Z. Shi,S. Zenu,and D. Feng Safevanish an improved data self destruction Proc. of the second international conference on cloud computing and data technology and science pp.521-528, 2010.

BIOGRAPHY

Swathi J was born at Thalassery, Kerala, India. The author received Bachelor Degree in the field of Computer Science and Engineering, College of Engineering Thalassery, Kerala, India in the year 2014. She is currently pursuing her Master of Engineering in the field of Computer Science and Engineering, Malabar Institute of Technology, Kannur, Kerala. The author's areas of interest and research work involve Cloud Computing, Information Security, Network Security and Mobile Security.