ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 3, March 2017

Self Destructing Data System For Distributed Object Based Active Storage Framework

[1] Mrs. Bharati A. Raut, [2] Mr. Nitin S. Thakre, [3]rs. Rupali A. Fulse
[1][2][3] Assistant Professor
Department of Information Technology,
Priyadarshini Institute of Engineering and Technology,Nagpur,India

Abstract - **With the development of Cloud computing and popularization of mobile Internet, Cloud services are becoming more and more important for people's 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 service providers will provide security policy to protect their data from leaking, so other people will not invade their privacy. As people rely more and more on the Internet and Cloud technology, security of their privacy takes more and more risks. As all the concept and researches have been evolved on the self destructing data system on the cloud computing, this paper is all about the web site security. That means the same concept of self destructing data system is used and applied on the web site, as web site users are tremendously in large numbers than the cloud users.**

Keywords:--- **Active storage, Cloud computing, Data privacy, self-destructing data.**

I. INTRODUCTION

With development of fast network access, the popularization of Internet are becoming more and more important for people's life. People are more or less requested to submit or post some personal private information to the website byInternet. As people rely more and more on the Internet technology, security of their privacy takes more and more risks. On the one hand, when data is being processed, transformed and stored by the current computer system or network, systems or network must cache, copy or archive it. These copies are essential for systems and the network. A goal of creating data that self-destructs or vanishes automatically after it is no longer useful. More-over, it should do so without any explicit action by the users or any party storing or archiving that data, in such a way that all copies of the data vanish simultaneously from all storage sites, online or offline. Vanish [1] supplies a new idea for sharing and protecting privacy. In the Vanish system, a secret key is divided and stored in a P2P system with distributed hash tables (DHTs). With joining and exiting of the P2P node, the system can maintain secret keys. According to characteristics of P2P, after about eight hours the DHT will refresh every node. With Secret key Sharing Algorithm [2], when one cannot get enough parts of a key, he will not decrypt data encrypted with this key, which means the key is destroyed. A self-destructing data system, or SeDas, which is based on an active storage framework [5]–[10]. 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.

In the key distribution algorithm, algorithm [2], which is used as the core algorithm to implement client (users) distributing keys in the object storage system. These methods to implement a safety destruct with equal divided key. Based on active storage framework, we use an object-based storage interface to store and manage the equally divided key. Through functionality and security properties evaluation of the SeDas prototype, the results demonstrate that SeDas is practical to use and meets all the privacy-preserving goals. The prototype system imposes reasonably low runtime overhead. SeDas supports security erasing files and random encryption keys stored in a hard disk drive (HDD) or solid state drive (SSD), respectively.

II. DETAILS EXPERIMENTAL

2.1. Related Work A new scheme that supports dynamic grouping and sharing system, where any member is allowed to join or his/her participation in the group. Revocation can be simply performed without making trouble in updating secret keys of other group members. A cryptographic storage system which enables secure sharing of files on untrusted server called Plutus. The large files are divided into no of small files each then encrypted with leave the group at any point in time.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 3, March 2017

Group signature identifies and authenticates each member in the group to ensure the quality of the secret. User revocation performed by using a revocation list generated by cloud server. Each newly joined member can decrypt files without contacting data owners before a secret key. Relevant users receives corresponding key to decrypt each small file blocks. Even though the system is flexible enough, heavy key distribution overhead experienced for large scale file sharing. Moreover, inorder to perform user revocation entire key pairs has to be updated. Based on Encryption method ,the data owner usually the group manager selects a key randomly to encrypt files and the chosen random key given to corresponding users for decoding the secret key. Decryption only performed if and only if it satisfies predefined access structure of the file. This system also requires updating secret keys of all users to perform a user revocation operation.

2.2. Data Self Destruction Self-Destruction data is implemented by encrypting data with a key and that information is needed to reconstruct the decryption key with one or more third parties. A local data destruction approach will not work in the Cloud storage because the number of backups or archives of the data that is stored in the Cloud is unknown, and some nodes preserving the backup data have been offline. The clear data should become permanently unreadable because of the loss of encryption key,(1) even if an attacker obtains a copy of the encrypted data and the user's cryptographic keys and passphrases after the timeout, (2) without the user or user's agent taking any explicit action to erase it, (3) without needing to modify any saved copies of that data, and (4) without the user relying on secure hardware. The self-destructing data system in the web site environment should meet the following requirements:

i) How to destruct all copies of the data.
ii) No explicit delete actions by the user, or any third-party storing that data.
iii) No compelling reason to adjust any of the saved or documented copies of that data.
iv) No use of secure hardware but support to completely erase data in HDD and SSD, respectively.

Vanish is a system for creating messages that automatically self-destruct after a period of time. It integrates cryptographic techniques with global-scale, P2P, distributed hash tables (DHTs): DHTs discard data older than a certain age.The key is permanently lost, and the encrypted data is permanently unreadable after data expiration. Vanish works by encrypting each message with a random key and storing shares of the key in a large, public DHT. Based on active storage framework, this paper proposes a distributed object-based storage system with self-destructing data function. Our system combines a proactive approach in the object storage techniques and method object, using data processing capabilities of OSD to achieve data self-destruction. User can specify the key survival time of distribution key and use the settings of expanded interface toexport the life cycle of a key, allowing the user to control the subjective life-cycle of private data.

2.3. Object - Based Active Storage Abbreviated as OSD, an Object-Based Storage Device is a device that implements the standard in which data is organized and accessed as objects, where object means an ordered set of bytes (within the OSD) that is associated with a unique identifier. Objects are allocated and placed on the media by the OSD logical unit. With an OSD interface, metadata is associated directly with each data object and can be carried between layers and across storage device files and records are no longer abstractions, but actual storage objects that are understood, managed and secured at the device level. Object-based storage (OBS) uses an object-based storage device (OSD) as the underlying storage device. The T10 OSD standard is being developed by the Storage Networking Industry Association (SNIA) and the INCITS T10 Technical Committee. 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. 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.

2.4. Completely Erase Bits Of Encryption Key

In SeDas, erasing files, which include bits (Secret Key Shares) of the encryption key, is not enough when we erase/ delete a file from their storage media; it is not really gone until the areas of the disk it used are overwritten by new information. With flash-based solid state drives (SSDs), the erased file situation is even more complex due to SSDs having a very different internal architecture. For instance, different from erasing files which simply marks file space as available for reuse, data wiping overwrites all data space on a storage device, replacing useful data with garbage data. Depending upon the method used, the overwrite data could be zeros (also known as "zero-fill") or could be various random patterns [41]. The ATA and SCSI command sets include "secure erase" commands that should sanitize an entire disk. Physical destruction and degaussing are also effective. SSDs work differently than

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 3, March 2017

platter-based HDDs, especially when it comes to read and write processes on the drive. The most effective way to securely delete platter-based HDDs (overwriting space with data) becomes unusable on SSDs because of their design. Data on platter-based hard disks can be deleted by overwriting it. This ensures that the data is not recoverable by data recovery tools. This method is not working on SSDs as SSDs differ from HDDs in both the technology they use to store data and the algorithms they use to manage and access that data.

2.5. System Architecture

Fig. 1 shows the architecture of SeDas. There are three parties based on the active storage framework.Metadata server (MDS): MDS is responsible for user management, server management, session management and file metadata management.

i) Application node: The application node is a client to use storage service of the SeDas.

ii) 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 is used to process active storage request from users and manage method objects and policy objects.

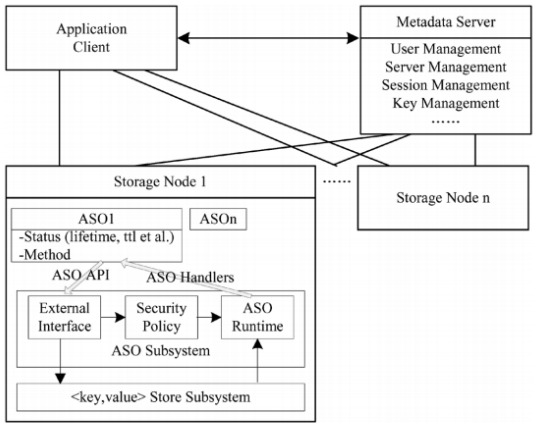

Fig. 1. SeDas system architecture.

2.6. Active Storage Object

An active storage object derives from a user object and has a time-to-live (ttl) value property. The ttlvalue is used to trigger the self-destruct operation. The tllvalue of a user object is infinite so that a user object will not be deleted until a user deletes it manually. The ttlvalue of an active storage object is limited so an active object will be deleted when the value of the associated policy object is true.Interfaces extended by ActiveStorageObjectclass are used to manage ttlvalue. 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, ActiveStorageObject::create will call UserObject::create first and associate it with the self-destruct method object and a self-destruct policy object with the ttlvalue. The getTTLmember function is based on the read_attrfunction and returns the ttlvalue of the active storage object. The setTTL, addTimeand decTimemember function is based on the write_attrfunction and can be used to modify the ttlvalue. 2.7. 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 libccan 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 lunargument specifies the device, the pid argument specifies the partition and the obj_idargument specifies the object to be destructed. 2.8. Data Process To use the SeDas system, user's applications should implement logic of data process and act as a client node. There are two different logics: uploading and downloading. i) Uploading file process (Fig. 2):When a user uploads a file to a storage system and stores his key in this SeDas system, he should specify the file, the key and ttlas arguments for the uploading procedure. The ENCRYPT procedure uses a common encrypt algorithm or 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. ii) 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.

III. EVALUATION

The evaluation platform supports simple file management, which includes some data process functions

such as file uploading, downloading and sharing. 1) Functional Testing:We input the full path of file, key file, and the life time for key parts. The system encrypts data and uploads encrypted data. The life time of key parts is 15 minutes for a sample text file with 101 bytes. System prompts creating active object are successful afterwards and that means the uploading file gets completed. The time output finally is the time to create active object. SeDas was checked and corresponded with changes on work directory of the storage node. The sample text file also was downloaded or shared successfully before key destruct. 2) Performance Evaluation:As mentioned, the difference of I/O process between SeDas and Native system is the additional encryption/decryption process which needs support from the computation resource of SeDas' client. We compare two systems: i) A self-destructing data system based on active storage framework (SeDas for short) ii) A conventional system without self-destructing data function. Compared with the Native 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%.

IV. RESULTS AND DISCUSSION

There are multiple storage services for a user to store data. Meanwhile, to avoid the problem produced by the centralized "trusted" third party, the responsibility of SeDas is to protect the user key and provide the function of self-destructing data. In this structure, the user application node contains two system clients: any third-party data storage system (TPDSS) and SeDas. The user application program interacts with the SeDas server through SeDas' client, getting data storage service.The process to store data has no change, but encryption is needed before uploading data and the decryption is needed after downloading data. In the process of encryption and decryption, the user application program interacts with SeDas. The client mainly runs in kernel mode, and it can mount a remote file system to local.

V. ACKNOWLEDGMENTS

Data privacy has become increasingly important in the Web Site 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 T10OSD standard. SeDas causes sensitive information, such as account 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.

REFERENCES

1. Lingfang Zeng, Shibin Chen , Qingsong Wei and DanFeng" SeDas: A Self-Destructing Data SystemBased on Active Storage Framework" IEEE TRANSACTIONS ON MAGNETICS, VOL. 49, NO. 6, JUNE 2013.

2. Karthik D U, Madhu C, Sushant M " A Systematic Approach to Cloud Security Using SeDas Platform," International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 3 Issue 5 may, 2014 Page No. 5940-5947.

3. R. C. Dharmik, Hemlata Dakhore, Vaishali Jadhao "Sedas: A Self destructive Active Storage Framework for Data Privacy" ISSN (Online): 2347-3878 Volume 2 Issue 3 March 2014.

4. P.Mani Priyadharsini, Mr.M.Gughan Raja, "Time Constrained Data Destruction in Cloud," International Journal of Innovative Research in Science, Engineering and Technology (An ISO 3297: 2007 Certified Organization) Vol. 3 , Issue 4 , April 2014.

5. N. RamaKalpana1, R. Santhosh2 " SeDas: SELF - DESTRUCTION DATA SYSTEM FOR DISTRIBUTED OBJECT BASED ACTIVE STORAGE FRAMEWORK," IJSWS 14-160; © 2014.

6. Backya S, Palraj K " Declaring Time Parameter to Data in Active Storage Framework," International Journal of Advanced Research in Computer engineering & Technology (IJARCET) Volume 2, Issue 12, December 2013.

7. IR.Ramachandran, IIM.P. Revathi, "SADDs – Self Annihilation and downloadable Data system in Cloud Storage Service," IJARCST Vol. 2 Issue Special 1 Jan-March 2014.

8. Lalitha K1, Sasi Devi J2, "SEDAS: A Self Destruction for Protecting Data Privacy in Cloud Storage As A Service Model," International Journal of Innovative Research in Science, Engineering and Technology , Volume 3, Special Issue 1, February 2014.

9. Backya S, Palraj K, "Declaring Time Parameter to Data in Active Storage Framework," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 12, December 2013.

10. Mohan Sadasivam1, Rajeeve Dharmaraj2, "SADS – Self Annihilating Data Storage system in Cloud Storage Service," International Journal of Information & Computation Technology. ISSN 0974-2239 Volume 4, Number 11 (2014), pp. 1035-1042.

11. Ranjith.K, P.G.Kathiravan, " A SELF-DESTRUCTION SYSTEM FOR DYNAMIC GROUP DATA SHARING IN CLOUD," IJRET: International Journal of Research in Engineering and Technology, Volume: 03 Special Issue: 07 | May-2014.

12. N.S.Jeyakarthikka, S.Bhaggiaraj, A.Abuthaheer, " Self Destructing Data System Based On Session Keys," INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 3, ISSUE 2, FEBRUARY 2014.