www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume - 3 Issue -9 *September*, 2014 Page No. 8155-8158

Extended self destructing data system with data recovery

Vinayaka R H,   Dayananda P , Shwetha S

PG student, Department of Information Science & Engineering MSRIT College, Bangalore, Karnataka, India
vinayak.rh7@gmail.com

Assistant Professor, Department of Information Science & Engineering MSRIT College, Bangalore, Karnataka, India
dayanandap@msrit.edu

Assistant Professor, Department of Information Science & Engineering RVCE College,
Bangalore, Karnataka, India
shwetha.ise@rvce.edu.in

Abstract*:* ***Personal data stored in cloud may contain account numbers or passwords or notes or any other important information that could be used and misused by a hacker, a competitor, or a court of law. These data are cached, copied, and used by Cloud Service Providers (CSPs), often without users authorization and control. SeDas mainly aims at protecting the user data's privacy. All these data and their copies become destructed or unreadable after a user-specified time, without any user intervention. In addition to that the decryption key will be destroyed after the user-specified time. The presence of SeDas, a system that meets this challenge through a novel integration of cryptographic techniques with active storage techniques based on T10 OSD (Object Storage Device) standard provides a recovery mechanism to the legitimate users to obtain their data back by requesting to the cloud admin. A new key is be sent to the legitimate user either to the Email or to Mobile using this key he has to login to the SeDas platform to get back their data. So this approach is more efficient to use and possible to achieve all the privacy preserving goals.***

Keywords: ***Cloud computing, Data privacy, Self Destructing data, Active Storage***

1. INTRODUCTION

Recent advances of Cloud computing and popularization of mobile Internet Cloud services are becoming 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 that others people will not invade their privacy.

As people rely more on the Internet and Cloud technology security of their privacy takes more risks. On the one hand, when data is being processed or transformed and stored by the current computer system or network systems The copies provided are essential for systems and the network. However, these people have no knowledge about these copies and cannot control them, so these copies can leak their privacy.

On the other hand, privacy also can be leaked via Cloud Service Providers (CSPs) negligence, hackers or some legal actions. These problems represents formidable challenges to protect people's privacy. A pioneering study of Vanish[1] supplies a new idea for sharing and protecting privacy. In 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 key. According to characteristics of P2P, after eight hours the DHT will refresh every node. With Shamir Secret Sharing Algorithm, when user cannot get enough parts of a key, he/she will not decrypt data encrypted with the key, which means the key is destroyed.

Some special attacks to characteristics of P2P are a challenge of Vanish, uncontrolled in how long the key can survive is also one of the disadvantages for Vanish. Considering the disadvantages, this paper has a solution to implement a self-destructing data system, or SeDas, which is based on an active storage framework. The SeDas system defines two modules, a self-destruct method object that is associated with each secret key part and survival time parameter for each secret key part. If this is the case, SeDas can meet the requirements of self-destructing data with controllable survival time while users can use the system as a general object storage system. Our contributions are summarized as follows.

1) The focus on the related key distribution algorithm ie., Shamir Sceret key algorithm, which is used as the core algorithm to implement clients distributing keys in the object storage system. The use of these

methods implement a safety destruct with equally divided key (Shamir Secret Shares)

2) By functionality and security properties evaluation of the SeDas prototype, the results demonstrate that SeDas is good to use and meets all the privacy-preserving goals.

3) SeDas supports security erasing files and random encryption keys stored in hard disk drive (HDD) or solid state drive (SSD), respectively.

2. RELATED WORK

The following section gives the related work so far carried out in the scope of self destructing data. Vanish[2] is the system that provides the basic idea of self destructing data. The system developed is a prototype which is implemented using Distributed Hash Table (DHT). It used bittorrents Vuze DHT that can support eight hours timeout or PlanetLab hostedOpenDHT that can support one week timeout. This system provides a plug-in for Firefox browser that creates a message which automatically disappears after a specified period of time. Here the expiry time for the data is controlled by the DHT and not by the user. Later many extensions are been implemented on the Vanish system. Another system called FADE[4] , proposed by Tang et al provides a contribution for the self destructing data by integrating cryptographic techniques. These data will be encrypted before sending it. This system will delete the files and makes them unrecoverable by revoking the file access permissions. A system called File System Design with assured delete proposes three types of file delete. 1) Is the expiration known at the time of file creation, 2)Is on demand deletion of individual files and 3) Is the usage of custom keys for classes of data. As given above, many systems have been proposed to implement a self destructing system among which only some provide promising results. The system don't have a user controllable data expiration time. They rather have a fixed time for file expiration which is not an efficient approach for the self destructing scenario Zeng et al came up with the solution for user controllable expiry time with the integration of cryptography techniques along with active storage techniques based on T10 OSD(Object Storage Device) standards. In this system all the data will be self destructed after a specified time without user intervention. The system mainly concentrates on encrypting the data before encrypting and using the Shamir's secret key [3] distribution for the safeguarding the key. The key will be retroactively deleted once the expiry time reached, makes the data to unreadable.

3. SYSTEM DESIGN AND IMPLEMENTATON

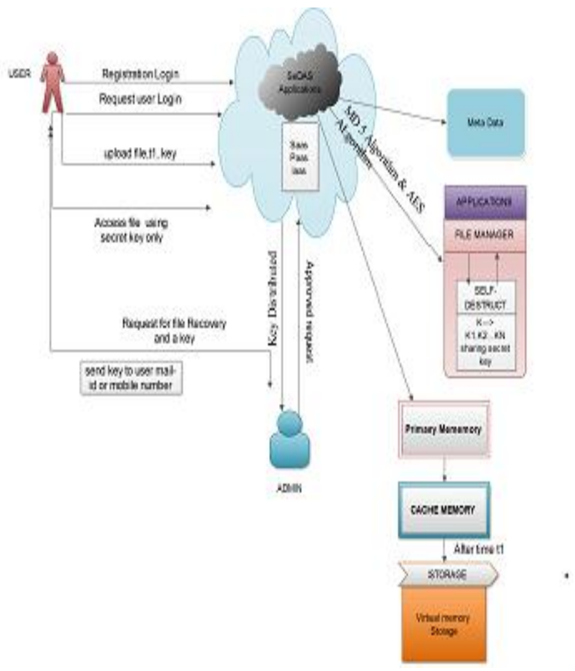

Figure 1: SeDas Architecture

There are three parties based on the active storage framework as shown in figure 1 **i**) **Metadata server** (MDS): MDS is responsible for user management, server management, session management and file metadata management. **ii**) **Application node**: The application node is a client to use storage service of the SeDas. **iii**) **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. object ID is used as a key. The associated data and attribute are stored as values.

A. 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 *tll* value of a user object is infinite so that a user object will not be deleted until a user deletes manually. The *ttl* value of an active storage object is limited so an active object will be deleted when the value of the associated policy object are true. 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, User Object:: create will be called to create a user object, else, ActiveStorageObject:: create will call User Object::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 the 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.

B. Self-Destruct Method Object

Generally, kernel code can be executed efficiently; however, the 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 are used to develop software in user

space. It is 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 the code of a service method in user space can take advantage of performance of the system. The system might crash with error in kernel code, but this will not happen if error occurs in code of user space.
A self-destruct method object is a service method. It needs these three arguments, the lun argument specifies the device and the pid argument specifies the partition and the obj_id argument specifies the object to be destructed.

C. 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. The logic for uploading and downloading is as below:

- **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 key and *ttl* as the arguments for uploading procedure. Algorithm 1 presents its pseudo-code. In these codes, our assumption is that the data and key has been read from the file. The ENCRYPT procedure uses common encrypt algorithm or user-defined encrypt algorithm. After uploading data to storage servers, key shares generated by algorithm will be used to create active storage object (ASO) in storage node in the SeDas system.
- **Downloading file process**:
  Any user who has relevant permission can download data stored in the data storage system. The data are decrypted before use. The whole logic are implemented in code of user's application.

D. Data Security Erasing in Disk:

To 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 that great, so if this part of the file updates operation changes, then OSD performance will be impacted greatly. Our implementation method are as follows: **i**) The system pre specifies a directory in a special area to store sensitive files. **ii**) Monitors the file allocation table and acquire and maintains list of all sensitive documents, the logical block address (LBA). **iii**) LBA list of sensitive documents appear to increase or decrease, the update is sent to the OSD. **iv**) OSD internal synchronizations maintain the lists of LBA, the LBA data in the list updates. For example, for SSD, the old data page writes 0, and then another writes the new data page. When the LBA list is shorter than the corresponding file, size is shrinking. **v**) For ordinary LBA, system uses the regular update method. **vi**) By using ordinary data erasure API, safety is assured to delete sensitive files of the specified directory.

E. Recovery Mechanism:

A recovery mechanism is provided to the legitimate users to obtain their data back by requesting to the cloud admin. A new key will be sent to the legitimate user either to the Email or to Mobile using this key he has to login to the SeDas platform to get back data.

```
Algorithm1: Upload file (data, key, ttl)
//Input – data: Data which has to be uploaded
            Key: Security Key
            Ttl: time to leave (time to which the data on the
server should reside)
//Output – Success or Failure status
BEGIN
//encrypt the input data with the key
Buffer = ENCRYPT (data, key);
Connect to data storage network
If failed then return FAILURE
Create file in the data storage server
//use Shamir's algorithm to create data[2]
//k is count of data servers in SeDas systems
shared Keys[1…k]=Shamir Algorithm(n,k,key)
for I from 1 to k
connect to DS[i]
if successful then
create object(shared Keys[i],ttl)
else
for j from 1 to I then
    delete keys share  created before this one
end for
return FAILURE
endif
endfor
return SUCCESS
END
```

4. RESULTS

The experimental setup includes the usage of the open source cloud service provider such as Jelastic cloud platform. The cloud platform allows the user to use the basic functionalities and software's like TOMCAT, JAVA compilers. These services will be provided free for over a period of 15 days. This minimum support is enough to deploy application on cloud. The database provided on the cloud will help to store the files uploaded and the encrypt keys used for each file, user is not aware of where the data is stored on the cloud, a separate private cloud environment is created and used for the experimental purpose. The figure 2 shows the time taken for system to upload the file with different sizes and figure 3 shows the time taken for system to encrypt the file with different sizes

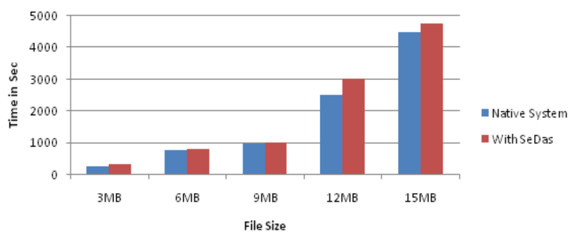

Figure 2: Comparison of time taken to upload a file with different sizes

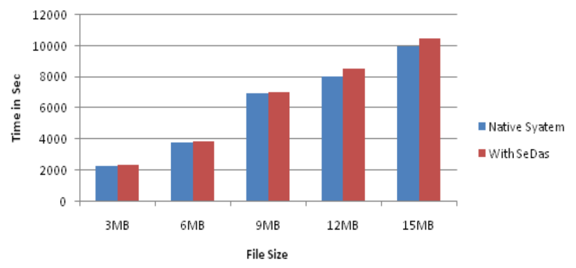

Figure 3: Comparison of time taken to encrypt a file with different size.

5. CONCLUSION

Data privacy has become increasingly important in the Cloud environment. A new approach is introduced for protecting the data privacy from attackers who may obtain user's stored data and private decryption keys. The novel aspect of our approach is the leveraging of the essential properties of active storage framework based on T10OSD standard. The demonstration shows the feasibility of our approach by presenting SeDas, a proof-of-concept prototype based on object-based storage techniques. A recovery mechanism is also provided to the legitimate users to obtain their data back by requesting to the cloud admin. Our measurement and experimental results demonstrates that SeDas is practical to use and provides security to user's data in cloud. The current SeDas system will help researchers to provide information for the design of cloud services with object-based storage system.

6. REFERENCES

[1] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy, "Vanish: Increasing data privacy with self-destructing data," in Proc. USENIX Security Symp., Montreal, Canada, Aug. 2009, pp. 299–315.

[2]R. Geambasu, T. Kohno, A. Levy, and H. M. Levy, 'Vanish: Increasing data privacy with self-destructing data,' in Proc. USENIXSecurity Symp., Montreal, Canada, Aug. 2009, pp. 299–315.

[3] A. Shamir, 'How to share a secret,' Commun. ACM, vol. 22, no. 11, pp. 612–613, 1979.

[4] R. Perlman, 'File system design with assured delete,' in Proc. Third IEEE Int. Security Storage Workshop (SISW), 2005.

[5] L. Qin and D. Feng, "Active storage framework for object-based storage device," in Proc. IEEE 20th Int. Conf. Advanced Information Networking and Applications (AINA), 2006.

[6] Y. Zhang and D. Feng, "An active storage system for high performance computing," in Proc. 22nd Int. Conf. Advanced Information and Applications (AINA), 2008, pp. 644–651.

[7] T. M. John, A. T. Ramani, and J. A. Chandy, "Active storage using object-based devices," in Proc. IEEE Int. Conf. Cluster Computing, , pp. 472–478.

[8] A. Devulapalli, I. T. Murugandi, D. Xu, and P. Wyckoff, 2009, Design of an intelligent object-based storage device.

[9] S. W. Son, S. Lang, P. Carns, R. Ross, R. Thakur, B. Ozisikyilmaz, W.-K. Liao, and A. Choudhary, "Enabling active storage on parallel I/O software stacks," in Proc. IEEE 26th Symp. Mass Storage Systems and Technologies (MSST), 2010.