An Efficient Approach to Enhance Data Security in Cloud Storage

S. Preethi
B.E-CSE

J. G. Prithvikha Roshini
B. E-CSE

J. Sulthan Alikhan
Assistant Professor (Sr. Gr)

Department of Computer Science and Engineering
K.L.N. College of Engineering
Sivagangai District, Tamilnadu, India

Abstract - **Large amount of data in the cloud may contain account numbers, passwords, notes and other information. This information could be misused by a miscreant or a competitor. Cloud Service Providers (CSPs) will cache, copy and archive the data often without users' authorization and control. Self-destructing data mainly aims at the user data's privacy. Data, their copies and the decryption keys will be destructed after the user- specified time. In this paper, the user's data is divided into two halves. Each contains its own private keys. The data cannot be decrypted without enough parts of the key. By using the self-destructing data method, the information needed to reconstruct the key will be permanently lost after a time period. Hence, the results demonstrate that it is practical to use and meets all the privacy-preserving goals.**

Index Terms- Self-destructing data, Data privacy.

1. INTRODUCTION

Cloud computing is considered as the next step in the evolution of on-demand information technology which combines a set of existing and new techniques from research areas such as service-oriented architecture and virtualization. The data in cloud servers, however, usually contains users' sensitive information (e.g., personal profile, financial data, health records, etc.) and needs to be well protected. Security of data privacy has become more challenging since people rely more on the Internet and Cloud Computing. On the one hand, when data is being processed, transformed and stored by the current computer system or network, they must cache, copy or archive it. The systems and the network need these copies as they are essential. However, people do not have any knowledge about these copies and cannot control them, so these copies may leak their privacy. On the other hand, their privacy can also be leaked by Cloud Service Providers (CSPs') negligence, hackers' intrusion or some illegal actions. In considering these disadvantages, this paper demonstrates a solution to implement a self-destructing data system, or SeDas, which is based on an active storage framework. The SeDas system defines two new modules. One is a self-destruct method object that is associated with each secret key part and the other is survival time parameter for each secret key part.

2. RELATED WORK

A. Self-Destruction of Data

The SeDas system in the Cloud environment should meet the following requirements: **i)** The clear data should become permanently unreadable due to the loss of the encryption key, even if an attacker can obtain a copy of that data; **ii)** Explicit delete actions by the user, or any third-party storing that data are not allowed; **iii)** No need to modify any of the stored or archived copies of that data; **iv)** No use of secure hardware but support to completely erase data in HDD and SSD respectively.

B. Active Storage

Today, the active storage system has become one of the most important research branches in the domain of intelligent storage systems. The data can be processed in storage devices; people attempt to add more functions into a storage device and make it more intelligent and referred to it as "Intelligent Storage" or "Active Storage". For instance, there is a model of load-managed active storage, which strives to integrate computation with storage access in a way that the system can predict the effects of offloading computation to Active Storage Units (ASU). Hence, applications can be configured to meet the requirements of the hardware capabilities and load conditions.

C. Completely Erase Bits of Encryption Key

In SeDas, erasing files, which include bits (Shamir Secret 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. Software methods typically involve overwriting all or part of the drive multiple times with patterns specifically designed to obscure any remnant data. For instance, different from erasing files which simply marks files 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, the overwritten data could be zeros ("zero-fill") or could be various random patterns. Extensive experiments show that the proposed SeDas does not affect the usual storage system and can meet the requirements of self-destructing data under a survival time by user controllable key.

3. IMPLEMENTATION

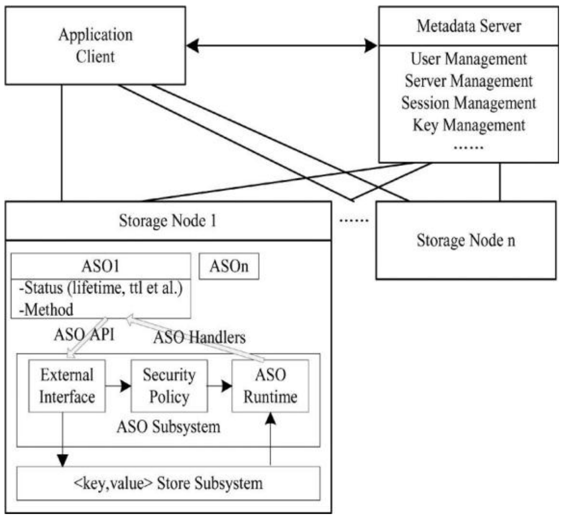

Fig. 1. SeDas system architecture

A. SeDas Architecture

SeDas architecture consists of three parties based on the active storage framework. **i)** Metadata server is responsible for user management, server management, session management and file metadata management. **ii)** Application node is a client to use storage service of the SeDas. **iii)** Storage node: Each storage node is in 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.

B. Self-Destruct Method Object

A self-destruct method object is a service method. A service method needs a long time to process a complicated task, so implementing code of a service method in user space can take favor of performance of the system. The system might collapse 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 needs three arguments. The lun argument indicates the device, the pid argument indicates 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.

i) Uploading file process: When a user uploads a file to a storage system and save his key in this SeDas system, he should specify the file, the key and ttl as arguments for the uploading procedure. We assume data and key has been read from the file in the code. The ENCRYPT procedure uses a common encrypt algorithm or user-defined encrypt algorithm. After uploading data to storage server, key shares generated by ShamirSecretSharing algorithm will be used to create active storage object 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 prior to use.

D. Data Security Erasing In Disk

Our implementation method is as follows: **i)** The system pre-specifies a directory in a special area to store sensitive files, **ii)** Monitor the file allocation table and acquire and maintain a risk 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 synchronization manages the list of LBA, the LBA data in the list updates, **v)** For ordinary LBA, the system uses the regular update method.

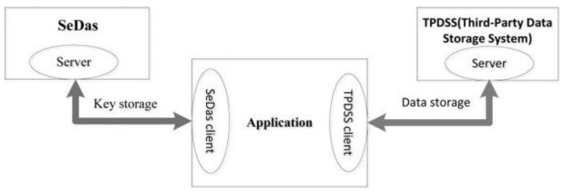

Fig. 2. Structure of user application program realizing storage process.

4. EVALUATION AND DISCUSSION

The difference of I/O process between SeDas and Native system is the additional encryption/decryption process which needs help from the computation resource of SeDas' client. We compare two systems: **i)** a self-destructing data system based on active storage framework, and **ii)** a conventional system without self-destructing data function.

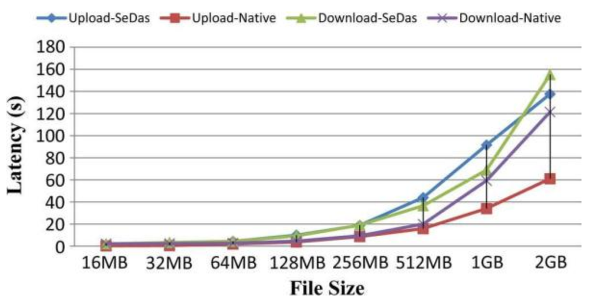

Fig. 3. Comparisons of latency in the upload and download operations.

Fig. 3 shows the latency of different schemes. We observe that SeDas increases the average latency of the Native system by 59.06% and 25.69% for the upload and download, respectively.

ACKNOWLEDGEMENT:

This Research work was supported by the KLN College of Engineering, (approved research centre by Anna University). The authors are grateful to the anonymous reviewers for their valuable comments that helped in improving the paper.

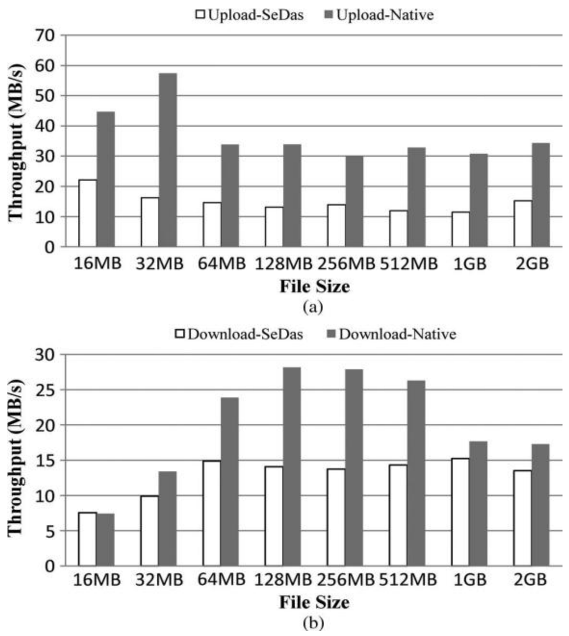

Fig. 4. Comparisons of throughput in the upload and download operations.

Fig. 4 shows the throughput results for the different schemes. We can see that SeDas reduces the throughput over the Native system by an average of 59.5% and upto 71.67% for the uploading and by an average of 30.5% and upto 50.75% for the downloading.

CONCLUSION:

Data Privacy in the Cloud environment has become increasingly significant. This paper introduced a new approach for protecting data privacy from attackers who obtain a user's stored data and private decryption keys through legal or other means. We demonstrated the feasibility of our approach by presenting SeDas that causes irreversible self-destruction of sensitive information such as passwords, account numbers and important details, without the involvement of user. The practicability of our approach is gained from our measurement and experimental security analysis.

REFERENCES:

[1] Z. Fu, X. Sun, Q. Liu, et al, "Achieving Efficient Cloud Search Services: Multi-Keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing," Ieice Trans. Communications, vol. E98.B, no. 1, pp. 190-200, 2015.

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

[3] T. M. John, A. T. Ramani, and J. A. Chandy, "Active storage using object-based devices," in Proc. IEEE Int. Conf. Cluster Computing, 2008, pp. 472-478.

[4] Z. Niu, K. Zhou, D. Feng, H. Chai, W. Xiao, and C. Li, "Implementing and evaluating security controls for an object-based storage system," in Proc. 24th IEEE Conf. Mass Storage Systems and Technologies (MSST), 2007.

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

[6] 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.

[7] E. Riedel, C. Faloutsos, G. Gibson, and D.Nagle, "Active disks for large scale data processing," IEEE Computer, vol. 34, no. 6, pp. 68-74, Jun. 2001.

[8] A. Shamir, "How to share a secret," Commun. ACM, vol. 22, no. 11, pp. 612-613, 1979.

[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.

[10] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Trans. Parallel and Distributed Systems, vol. 13, no. 3, pp. 260-274, 2002.

[11] V.Varadharajan, and U. Tupakula, "Security as a Service Model for Cloud Environment," IEEE Trans. Network and Service Management, vol. 11, no. 1, pp. 60-75, 2014.

[12] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for storage security in cloud computing," in Proc. IEEE INFOCOM, 2010.

[13] S. Wolchok, O. S. Hofmann, N. Heninger, E. W. Felten, J. A. Halderman, C. J. Rossbach, B. Waters, and E. Witchel, "Defeating vanish with low-cost Sybil attacks against large DHEs," in Proc. Network and Distributed System Security Symp., 2010.

[14] Y. Xie, K.-K. Muniswamy-Reddy, D. Feng, D. D. E. Long, Y. Kang, Z. Niu, and Z. Tan, "Design and evaluation of oasis:An active storage framework based on t10 osd standard," in Proc. 27th IEEE Symp.Massive Storage Systems and Technologies (MSST), 2011.

[15] L. Zeng, Z. Shi, S. Xu, and D. Feng, "Safevanish: An improved data self-destruction for protecting data privacy," in Proc. Second Int. Conf. Cloud Computing Technology and Science (CloudCom), Indianapolis, IN, USA, Dec. 2010, pp. 521-528.