ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 2, Issue 12, December 2013

Self-Demolition Information Framework For Securing Distributed Storage

S.P.Sharmila
PG Scholar
Kalasalingam Institute of Technology

V.Ramesh
Assistant professor
Kalasalingam Institute of Technology

***ABSTRACT*—As people rely more and more on the Internet and Cloud technology, security of their privacy takes more and more risks, They subjectively hope service providers will provide security policy to protect their data from leaking Individual information saved in the Cloud that could be utilized and abused by a fraud, a contender, or a court of law. These information are cached, copied, and archived by Cloud Service Providers (CSPs), often without users' authorization and control. Self-demolition data mainly aims at securing the user data's privacy. All the information and their duplicates get demolished or unreadable after a client-specified time, without any client mediation. In addition, the decryption key is demolished after the client-specified time. In this paper we present a solution Self-demolition, a system that meets this challenge through a novel integration of cryptographic techniques with an active storage framework. It hopely reduces hopping attack and sniffing attack. Some enhancements are added into the current version a file system with the help of the object storage mechanism, it make device more intelligent by enabling computation inside storage device. Which provides a hybrid approach to combine request driven model and policy-driven model. Extensive security and performance analysis show the proposed schemes are provably secure and highly efficient.**

***Index Terms*—Active storage, Cloud computing, Data privacy, Distributed storage, Vanishing data.**

INTRODUCTION

With 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 others people will not invade their privacy. The Self Demolition system defines two new modules, a self-demolition method object that is associated with each secret key part and survival time parameter for each secret key part[4].

A key observation in this system is that users need to keep certain data for only a limited period of time. After that time, access to that data should be revoked for everyone — including the legitimate users of that data, the known or unknown entities holding copies of it, and the attackers[13][14]. This mechanism will not be universally applicable to all users or data types; instead, we focus in particular on sensitive data that a user would prefer to see destroyed early rather than fall into the wrong hands

DATA SELF-DEMOLITION

Self-demolition is implemented by encrypting data with a key and then escrowing the information needed to reconstruct the decryption key with one or more third parties. Assuming that the key reconstruction information disappears from the escrowing third parties at the intended time, encrypted data will become permanently unreadable: (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. Once the key-reconstruction information disappears, data owners can be confident that their data will remain inaccessible to powerful attacks, whether from hackers who obtain copies of backup archives and passphrases or through legal means . The self-demolition data system 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.

Our system combines a proactive approach in the object storage techniques and method object, using data processing capabilities of OSD to achieve data self-demolition. User can specify the key survival time of distribution key and use the settings of expanded interface to export the life cycle of a key, allowing the user to control the subjective life-cycle of private data.

SYSTEM ARCHITECTURE

fig.1 shows the architecture of Self-demolition System. There are three parties based on the active storage framework. **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 Self-demolition system. **iii) Storage node**: Each storage node is an OSD. It contains two core subsystems: key value store subsystem and active storage object (ASO) runtime sub- system.

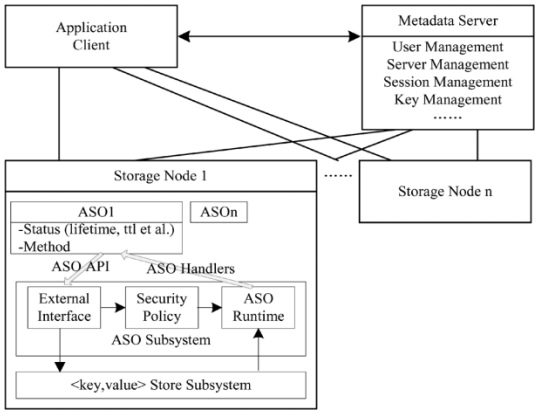
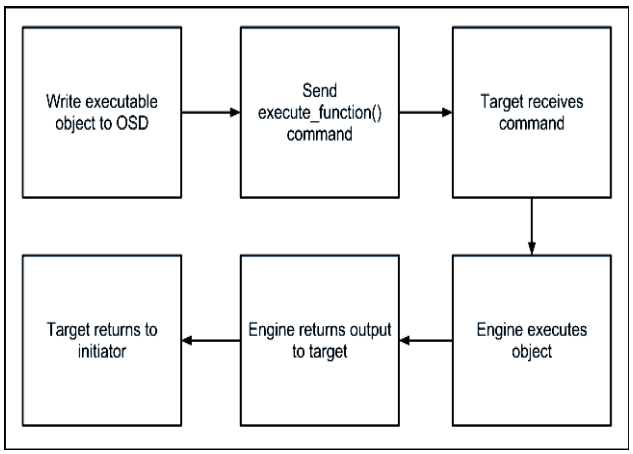


Fig1. system architecture

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.

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-demolition operation. The ttl value of a user object is infinite so that a user object will not be deleted until a user deletes it 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 is true[5][7].



Addition of the EXECUTE_FUNCTION() command
Sent over iSCSI triggers execution of an executable object carries information including: Object id, Arguments to the function, Return data from function. Take better advantage of fast RAID arrays and SSDs. Drives bottle-necked by slow networks, run applications in parallel across multiple nodes[6][8][9]. It make use of unused processor
time, moves part of the file system to the disk

- Deals with objects instead of blocks
- Data and metadata are separated, Direct access to data once authorized
- Object attributes are adaptable and directly modifiable
- Objects accessible by name on nodes, not as bytes
- High throughput conceivable through striping data

SELF-DEMOLITION METHOD OBJECT

A self-demolition 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 demolish. As the name implies, DHTs are designed to implement a robust index-value database on a collection of P2P nodes [1]. Intuitively, Vanish encrypts a user's data locally with a random encryption key not known to the client, demolitions the neighbourhood duplicate of the key, and then sprinkles bits (Shamir secret shares [2]) of the key across random indices (thus random nodes) in the DHT.

In cryptography, **secret sharing** refers to a method for distributing a secret amongst a group of members, each of which is allotted a *share* of the secret. Goal is to divide some data *D* (e.g., the safe combination) into n pieces D1, D2….Dn in such a way that:

- Knowledge of any k or more D pieces makes D easily computable.

- Knowledge of any k -1 or fewer pieces leaves D completely undetermined (in the sense that all its possible values are equally likely).

This scheme is called (k, n) threshold scheme. Assuming that k=n then all participants are required together to reconstruct the secret.

- Gives tight control and removes single focus vulnerability.

- Individual key share holder can't change/access the data.

COMPLETELY ERASE BITS OF ENCRYPTION KEY

Our proposed Self-demolition, we have implemented a fully functional prototype system. Based on this prototype, we carry out a series of experiments to examine the functions of Demolition[11][12]. Extensive experiments show that the proposed System does not affect the normal use of storage system and can meet the requirements of self-demolition data under a survival time by user controllable key. 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 is true.

Interfaces extended by *ActiveStorageObject* class are used to manage *ttl* value[7][10]. The create member function needs alternate 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-demolition method object and a self-demolition policy object with the *ttl* value. The *getTTL* member function is based on the *read_attr* function and returns the *ttl* value of the ac- tive 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.

DATA PROCESS

There are two different logics: uploading and downloading.

i) Uploading file process: When a user uploads a file to a storage system and stores his key in this Self DEMOLITION system, he should specify the file, the key and ttl as arguments for the uploading procedure. In pseudo-codes, we assume data and key has been read from the file. 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 Self DEMOLITION system.

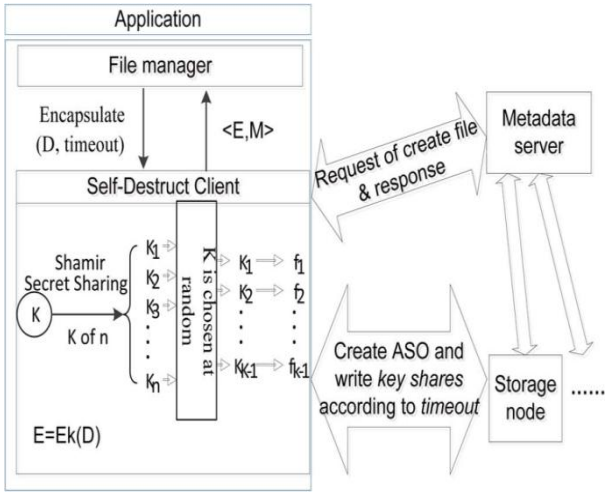


Fig 3.Uploading file process

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.

RESULT

There are multiple storage services for a user to store data. Meanwhile, to keep away from the issue generated by the centralized "trusted" third party, the responsibility of self-demolition system is to protect the user key and provide the function of self-demolition data. Fig. 4 shows the brief structure of the user application program realizing storage process. In this structure, the client application node contains two system clients: any third-party data storage system (TPDSS) and Self-demolition. The user application program interacts with the Self-demolition server through client, getting data storage service.

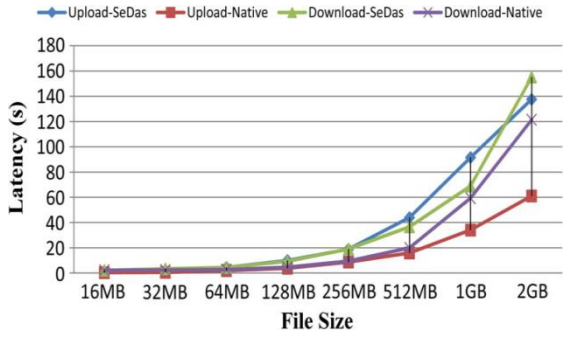


Fig. 4. Comparisons of latency in the upload and download operations.

EVALUATION:

The assessment stage based on pNFS supports simple file management, which includes some data process functions such as file uploading, downloading and sharing.

As mentioned, the difference of I/O process between self-demolition system and Native system (e.g. pNFS) is the additional encryption/decryption process which needs sup- port from the computation resource of client. We compare two systems: **i**) a self-demolition data system based on active storage framework, and **ii**) a conventional system without self-demolition data function (Native for short).

We evaluated the latency of upload and download with two schemes (*Self DEMOLITION* and *Native*) under different file sizes. Also, we evaluated the overhead of encryption and decryption with two schemes under different file sizes. Fig. 4 shows the latency of the system increases the average latency of the Native system by 59.06% and 25.69% for the upload and download, respectively.

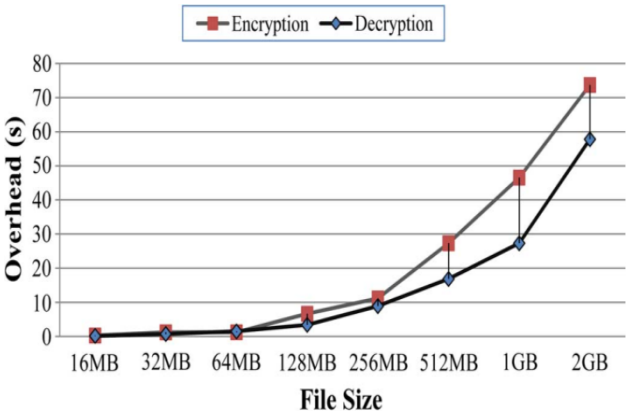


Fig. 5. Comparisons of overhead for encryption and decryption.

The reason for this performance degradation is the

encryption and decryption processes introduce the overhead To illustrate the encryption/decryption latency, Fig. 5 plots the overhead of both encryption and decryption processes under different file sizes in System.

CONCLUSION

Data privacy has become increasingly important in the 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 schema based on T10 OSD standard. We demonstrated the feasibility of our approach by presenting Self-Demolish system, a proof-of-concept prototype based on object-based storage techniques. It causes sensitive information, such as account numbers, passwords and notes to irreversibly self-demolition, without any action on the user's part. The fixed data timeout and large replication factor present challenges for a self-demolition data system.

REFERENCES

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

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

[3] S. Wolchok, O. S. Hofmann, N. Heninger, E. W. Felten, J. A. Hal- derman, 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.

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

[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 Networking 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,2008, pp. 472–478.

[8] A. Devulapalli, I. T. Murugandi, D. Xu, and P. Wyckoff, 2009, Design of an intelligent object-based storage device [Online]. Available: http://www.osc.edu/research/network_file/projects/object/papers/istor-tr.pdf

[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 stacks," in Proc. IEEE 26th Symp. Mass Storage Systems and Technologies (MSST), 2010.

[10] 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.

[11] Y. Tang, P. P. C. Lee, J. C. S. Lui, and R. Perlman, "FADE: Se- cure overlay cloud storage with file assured deletion," in Proc. SecureComm, 2010.

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

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

[14] R. Geambasu, J. Falkner, P. Gardner, T. Kohno, A. Krishnamurthy, and H. M. Levy, Experiences building security applications on DHTs UW-CSE-09-09-01, 2009, Tech. Rep..

[15] Azureus, 2010 [Online]. Available: http://www.vuze.com/99008004007

First Author S.P.Sharmila
The author is currently a ME Student in Computer Science and Engineering Department at Kalasalingam Institute of Technology. She had completed BE from Kalasalingam University. Her Research area is on Cloud computing, Network Security, and Data Mining.

Second Author Mr. V.Ramesh
The author is an Assistant Professor in Computer Science and Engineering Department at Kalasalingam Institute of Technology. He received his BE from Syed Ammal Engineering College; affiliated To Anna University, and M.Tech. Degree from Kalasalingam University. His Research interests are in the areas of Network Security, Data Mining and Cloud computing Security.