Increasing Data Privacy in Cloud by Self-Destruction Mechanism

Kshama Bothra
Department of Information Technology,
MIT College of Engineering,
Pune, India
kshama.bothra@gmail.com

Prof. Sudipta Giri
Department of Information Technology,
MIT College of Engineering,
Pune, India
sudipta.giri@mitcoe.edu.in

***Abstract*— Cloud computing technology entirely changed IT industry. There are many security issues related to cloud computing which falls into two broad categories: security issues faced by cloud providers and security issues faced by their customers. Immense data is stored in cloud storage system. Security is primary constraint for this immense data. This data include passwords, notes, account number. Self-destructing data aims at protecting user data's privacy. All the data and their copies become self-destructed after user specified time, without any user intervention. Decryption key is destructed after user-specified time. Shamir secret sharing algorithm is used, which generates a pair of keys. Self-destruction method is consociated with time to live (TTL) property to specify the life time of the keys. After user specified time (TTL) data and its keys becomes destructed or unreadable. Self-destruction mechanism helps reducing overhead during upload and download process in cloud.**

Keywords- Cloud computing; self-destruction; Active Storage Object; Time to live(ttl); data privacy.

I. INTRODUCTION

Cloud computing is a technique to increase capacity or capabilities dynamically without licensing new software, investing in a new framework, or training new personnel. Cloud services are becoming necessity in people's life. Advances of Cloud computing and popularization of mobile Internet attract people and they submit or post some personal private information to cloud by internet. People hope that service provider will provide security to data from leaking and third party won't invade their privacy.

People rely on cloud to store their data and this has become prime concern. Copies of data provided on the cloud environment are important for computer system and network. However users who store data don't have information about storage of these copies and can't control them. Copies stored in cloud environment may leak the privacy of user.

Privacy of the data stored in cloud can be leaked by other sources, like Cloud Service Provider's (CSPs) negligence, hackers or through some other legal actions.
Y. Tang [3] designs an approach to achieve flexible access control and large-scale dynamic data management in a high secure and efficient way. For large scale enterprise Fade is suitable option since they need large files with large amount of data. Three types of assured deletion: on-demand deletion of individual files, custom keys for classes of data, and. expiration time known at file creation

New idea for sharing and protecting data privacy is supplied by Vanish. Secret key is divided and stored in a P2P system with distributed hash tables (DHTs). P2P nodes refreshes after every eight hours. To decrypt data user will require enough parts of a key, unless he/she won't be able to decrypt data encrypted by the key. Keys will be destroyed after eight hours. Keys are stored in million-node Vuze Bit Torrent DHT in Vanish which causes two Sybil attack was proposed by S. Wolchok. Attacker continuously crawl DHT and saves each stored value before it ages out. This attack can efficiently recover about 99% of Vanish messages.

Safe-Vanish proposed by Linfang Zeng prevent hopping attack [2]. Length range of key shares is increased which increase the attack cost substantially. Improvements were done on Shamir Secret sharing algorithm. Using public key cryptosystem improved approach against sniffing attack was presented.

Attacks to the characteristics of P2P are a challenge of Vanish. Duration of key survival is also one of the disadvantages of Vanish. Considering the disadvantages Lingfang Zeng, proposed SeDas. SeDas is based on active storage framework. SeDas system has two modules, self-destruct method object that is associated with each secret key part and survival time parameter for each secret key part[4]. SeDas can meet the requirement of self-destructing data with controllable survival time. Self-Destructing data should meet the following requirement –

1) SeDas system focuses on Shamir's algorithm. Shamir's algorithm is used as a core algorithm to implement client distributing keys in an object storage system.
2) SeDas a novel system meets all the privacy preserving goals.
3) SeDas should support to completely erase data in HDD and SD.

II. LITERATURE SURVEY

Self-destructing data system in a cloud environment aims to meet the following requirements-
i) It should be able to destruct all the copies of data simultaneously and make them unreadable. The number of backups stored in cloud environment is not known hence local self-destruction mechanism will not work in cloud storage, also sometime it happens the nodes preserving backup data are offline. Hence data should become permanently unreadable due to loss of encryption keys;

ii) Data should be destructed after specified time period there should be no need to carry explicit action from user end;
iii) No modification is to be done on stored and archived copies of data.

Tang et al. proposed FADE. FADE is built upon standard cryptographic technique which aims to provide access control assured deletion of files that are hosted by cloud storage [3]. Files are associated with file access policy that controls how files are to be accessed. Policy-based file assured deletion is presented in which files are deleted and are made unrecoverable to anyone.

Perlmen et al. present three type assured delete: custom keys for classes of data, expiration time known at file creation and on-demand deletion of individual files.

Proposed Vanish, in Vanish messages automatically self-destruct after a period of time. It integrates cryptographic approach with global-scale, P2P, distributed hash tables (DHTs) [1]. DHT have property of discarding data older than certain age. The key is permanently lost, and the encrypted data becomes permanently unreadable after data expiration. Vanish first encrypts each message with random key and stores each share of key in a large, public DHT. Sybil attacks are possible in Vanish, it weakens the system by continuously crawling the DHT and saving each stored value before it ages out [7]. More than 99% of Vanish keys can be recovered. Wolchok et al. infer that public DHTs cannot provide strong enough safety for Vanish messages. Geambasu et al. derives that using OpenDHT and VuzeDHT together can provide security; if both DHTs are insecure then the hybrid will also be insecure. Vanish is compelling approach to the privacy problem, but it is insecure. To direct the problem of Vanish, SafeVanish was proposed. SafeVanish prevents hopping attack, which is one kind of Sybil attack. SafeVanish works by extending the length range of the key shares which increases the attack cost to a great extent, some enhancement were also done on Shamir Secret Sharing algorithm which was implemented in Vanish. Also, enhanced approach to protect sniffing attack was presented by using public key cryptosystem. However, use of P2P is still a calamitous weakness for both Vanish and SafeVanish because they are susceptible to attack (e.g. Sybil attack and hopping attacks)

III. PROPOSED SYSTEM

Self-destructive system defines two new modules, a self-destruct method object that is associated with each secret key part and each secret key part has its own survival time parameter. In the proposed system we can control survival time of the data stored in object storage system and hence can meet the requirement of self-destruction.

A. Self-destructive architecture

Fig. 1 shows the architecture of self-destructive. There are four parties based on the active storage framework.

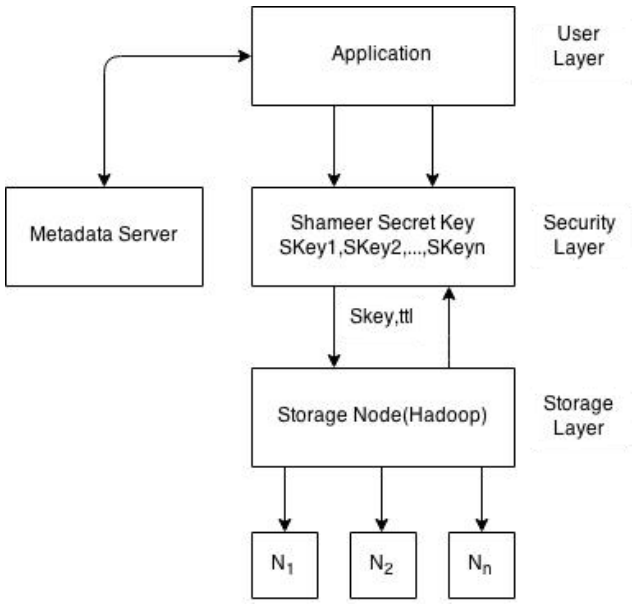

Figure 1: Proposed Architecture

i) Metadata server: Metadata server is responsible for session management, server management, user management and file metadata management.

ii) User layer: User layer is for the client that uses the storage services of SeDas. User layer is responsible for registration of new user and validation of existing user.

iii) Security layer: This layer uses Shamir Secret sharing algorithm. Key shares are generated. If a key is divided into n pieces than knowledge of at least k pieces of key is required to encrypt the data. If a user has less than k piece of key than data user won't be able to decrypt the data.

iv) Storage node: Each storage object is an OSD. It contains two core subsystem active object runtime system and key value store subsystem. Key value store system, which is small part, is based on object storage component and is used to govern objects stored in storage node: read/write object, lookup object and so on. The related data and attribute of the node are stored as values. Object ID is used as key.

B. Proposed plan of work

1. Development of Login System:-

Login page is created in this module. User registers himself/herself by filling the form. Validation of existing user is done on the basis of entries stored in database.

2. Development of Active Storage Framework:-

User derives active storage object and has time-to-live value property. Self-destruct operation is triggered by time-to-live value. In the earlier system time-to-live value of the user object was infinite i.e. user object is not deleted until user deletes it manually. The time-to-live value of an active object is restricted so the active object will be deleted once associated policy object is true.

3. Development of login tracking of the user:-

User application should implement the logic of data process and act as a client node. There are two different logics: uploading and downloading.

i) Uploading file process: User who wants to upload file to storage system, should first specify the file, the key and time-to-live as argument for uploading procedure. After uploading file to the storage server, key shares are generated using Shamir Secret Sharing algorithm.

Below are the steps for uploading file

```
Step1: UploadFile(data,key,ttl)
        data: data read from file to be uploaded
        key: data read from the key
        ttl: time-to-live of the key
Step2:  encrypt the input data with the key
        buffer=ENCRYPT(data,key)
Step3: connect to data storage server
        if failed then return fail;
Step4: create file in the data storage server and write
        buffer into it;
Step5: use ShamirSecretSharing algorithm to get key
       Shares
Step6: Connect to DS[i];
        if successful then
              create_object(sharedkeys[i],ttl);
        else
            for j from 1 to i then
                delete key shares created before this one
```

ii) Downloading file process: Any user who has appropriate permission can download data stored in data storage system. The data has to be decrypted before use.

IV. MATHEMATICAL MODEL

The system is modeled as $S = \{s, e, X, Y, \tau, F|\phi\}$
Where, s is the initial state
The user will input the data (*D*)
Provides key (*k*) and timespan (t_m)

e is the end state of the system which comprise of two states :

If $\tau < t_m$: Data will be retrieved from nodes (*N*)

If $\tau > t_m$: Data will be deleted from all nodes

X = Set of inputs in the system
$X = \{D, k, t_m\}$
Where, $D = \{d_1, d_2, d_3 ... d_n\}$
k = encryption key

t_m = time for which the data is present

Y = Set of outputs
$Y = \{R_D\}$
R_D = Retrieved data after decryption from various nodes using key (*k*)

τ = the time for which the data is present in the database.

Function $F = \{Enc_k(D), Decr_k(E)\}$
Where, $Enc_k(D)$ = Encryption of data using key (k) for storing data in encrypted format
$Dec_k(E)$ = Decryption of data for retrieving original data
Φ = Constraint on whole system.

V. EXPERIMENTAL SETUP

A. *Hardware configuration*

- Processor - Pentium- III
- Speed - 1.1 GHz
- RAM - 256 MB(min)
- Hard Disk - 20GB

B. *Software configuration*

- Operating System - Windows95/98/2000/XP
- Front End - HTML, Java Jsp
- Database - My sql 5.0
- Database Connectivity - JDBC

VI. CONCLUSION

Due to reliance of people to store data on cloud environment, it has become important to provide security to cloud environment. In this paper we introduced a new approach to provide security to data. Privacy to data is provided through the algorithm, as decryption of data cannot take place if the keys have been destructed with which data was encrypted. The property of active storage object is novel aspect of project.

ACKNOWLEDGEMENT

I am indeed thankful to my internal guide Prof. Sudipta Giri for his able guidance and assistance to complete this paper. I am grateful for their valued support and faith on me. I extend my special thanks to Head of Department of Information Technology, Prof. Anil S. Hiwale who extended the preparatory steps of this paper-work.

REFERENCES

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

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

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

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

[9] R. Geambasu, J. Falkner, P. Gardner, T. Kohno, A. Krishnamurthy, and H. M. Levy, "Experiences building security applications on DHTs", technical report, UWCSE- 09-09-01, 2009.