I J C T A, 7(2) December 2014, pp. 215-224

Highly Secured Cloud Environment with Self Destructing Data System (SeDas) Triggered by Time Parameter

S. Savitha[*] and D. Thilagavathy[**]

ABSTRACT

Evolution of cloud computing and advancement of Internet with mobile technology has become more and more frequent in people's everyday routine where they are subjected to give personal details to access the resources present on cloud. Cloud Service providers maintain these information at various location by storing, caching with multiple copies for easier access of the data. When users' personal details are at different locations securing it is the first and foremost thing to be done. Self-destructing data system solves this by providing an important technique for safe guarding the data which self-destructs it after the triggered time specified by the user. The data along with its copies becomes unreadable after certain period when it moves out of control. Self destructing mechanism meets this challenge by the combination of cryptographic techniques and active storage framework based on T10 OSD standard. These security procedures and its functionalities make sure that SeDas is easy for practical use with all privacy-preserving policies. "In comparison with the system without SeDas (native system) it is shown that the performance for uploading/downloading files has been achieved well, thus SeDas performs higher in terms of overhead and throughput".

Keywords: cloud computing, cloud service providers, cryptographic techniques, object storage device standard, self destructing data system.

1. INTRODUCTION

Cloud computing has taken a huge leap in people's day to day life that delivers various services over the Internet. When it comes to the usage part of cloud, it is mainly viewed a storage medium that stores, processes and transforms data by providing on-demand services to the users. There are various service providers in cloud namely Amazon, Google, IBM etc., Cloud computing has taken various forms for providing services such as Iaas (Infrastructure-as-a-Service), Paas (Platform-as-a-service), Saas (Software-as-a-Service).The cloud service providers (CSPs) are the ones who acts as an intermediate between the users and the cloud server. The cloud usage can be viewed in various forms. They are Private Cloud, Public Cloud, Hybrid Cloud and Community Cloud. Based on each type the users receive the services with different conditions.

Figure 1 shows the outer view of these services and interaction between the communicating parties. CSPs maintain the stored, processed and cached data on the cloud on behalf of the users. The users have no knowledge about these data copies as where it is stored or from where it is controlled. In such situations there are high chances for the data leakage that disturbs data privacy. These problems present inspiring challenges for protecting privacy for the data stored on cloud.

The new idea for this is provided by Vanish [1] system for protecting privacy and sharing the data securely. Figure 2 shows Vanish system architecture and its functions as how the keys are divided and stored in the P2P system with DHTs (Distributed Hash Tables).

[*] PG Scholar, Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, Tamil Nadu, India, *E-mail: savithasclick@gmail.com*

[**] Professor, Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, Tamil Nadu, India, *E-mail: thilagakarthick@yahoo.co.in*

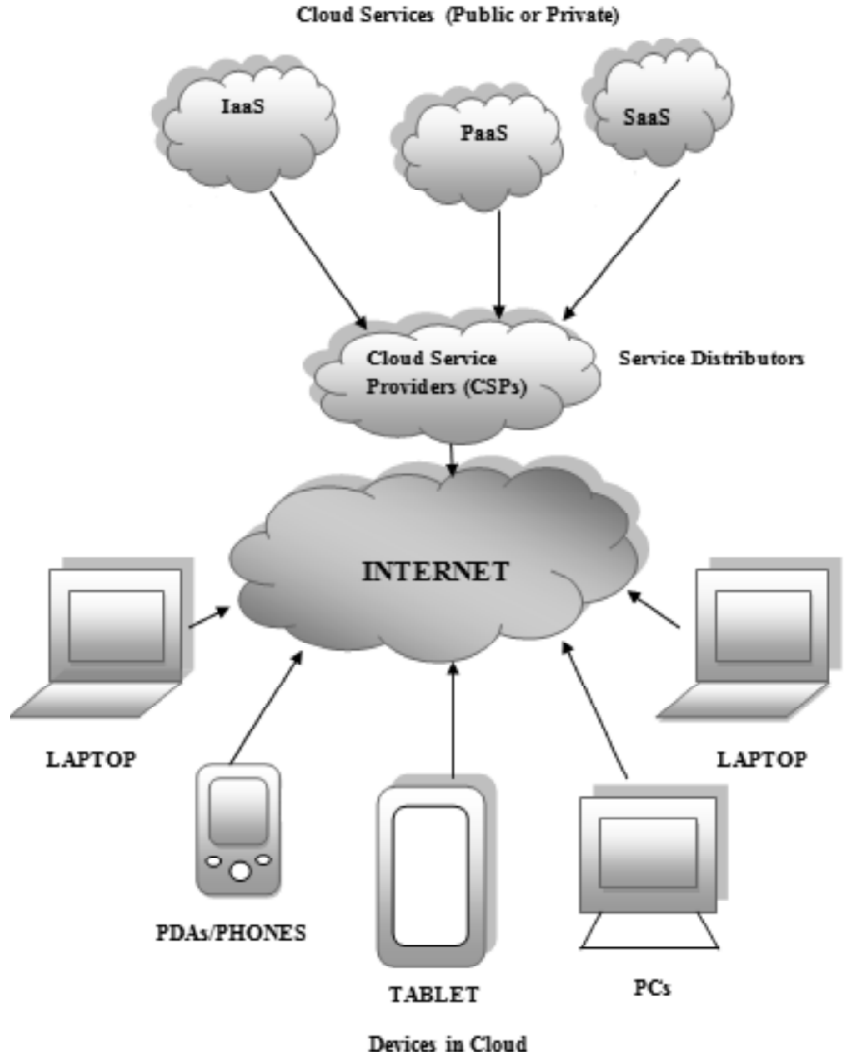

Figure 1: Overview of Cloud Services

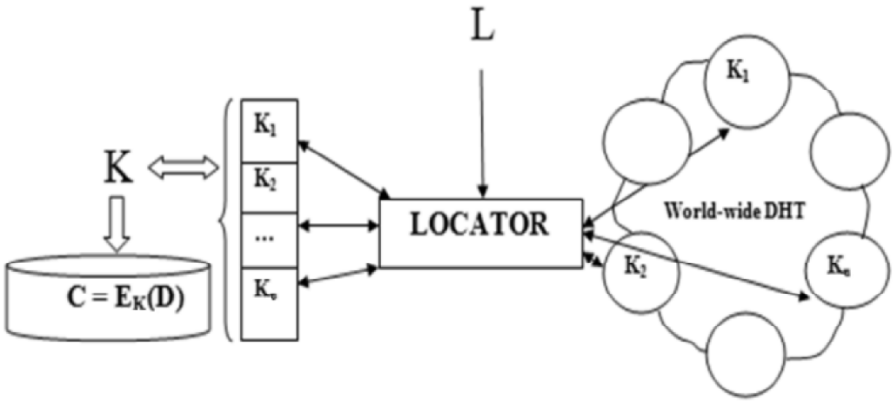

Figure 2: Vanish System Architecture

The secret keys are maintained by joining and exiting the P2P nodes. According to Shamir secret sharing [2] data decryption cannot happen when one cannot get enough portions of the key of the encrypted data which ultimately leads to destruction of the key. Moreover, DHTs maintain the secret keys which are sprinkled globally. The data in DHTs becomes unavailable over a certain period of time due to its cleansing property which permanently destroys the protected data.

Vanish is subjected to various types of attacks such as hopping attacks (the Sybil attack) [3] and also survival period of the key is also a major drawback over here. Having gone through these shortcomings this paper presents a solution by self-destructing the data with the SeDas system. SeDas functions by introducing a controllable survival time parameter with the secret key part that is stored as objects which is based on active storage framework [5]-[6]-[7].Hence the underlying work are as follows

1. The major focus is on Shamir Secret Sharing [2] algorithm which is an important key distribution algorithm that is being used as a core algorithm for implementing client side key distribution in object storage system. The equally divided keys are safely destructed using this method.
2. In order to manage the equally divided keys we go for active storage concept that uses object based storage interface to store and manage the keys.
3. The results of SeDas system shows that it is practical to use and implements all privacy-preserving goals with low runtime overhead.
4. SeDas provides secured erasing of file and random key encryption in both Hard Disk Drive (HDD) and Solid State Drive (SDD).

The remaining portion of the paper proceeds as follows. The actual problem is identified in the section II. All the related works are reviewed in section III. SeDas system architecture, implementation and design are described in the section IV. Then the extension of the paper evaluation is presented in section V. Finally the paper concludes in the Section VI followed by work on further enhancement.

2. PROBLEM IDENTIFICATION

Cloud services via social networking sites such as Facebook and twitter are being in use in our day to day life. As more and more enterprises and people are moving to cloud, it is mandatory for all cloud service providers to provide security to all data stored on it. Moreover the data is also cached, where the possibilities of security vulnerabilities are high that the data might be easily hacked. Services have to be ensured as hacking of our accounts can breach the security features. So some highly trusted technique has to be used so that the data remains in the hands of the user. A measure to solve this is provided ahead in this paper by starting with some discussions on the previous works.

3. RELATED WORK

3.1. Self destructing Data System

In cloud environment, the self destructing data system must allow the following requirements

1. The data with all its copies has to be self-destructed simultaneously and make them unreadable when it goes out of control.
2. The system should also make sure that self-destruction of the data with some local destructing mechanism will not work in case of cloud storage because the copies are cached over various locations for immediate access that is unknown to the user and the nodes having those backups might be offline.
3. The system should ensure that the copies even it is pristine should become unreadable after certain period of time in case it goes into illegal access.
4. There should not be any explicit data deletion by the user or any third party administrator.
5. There should not be any need for modifying those cached or archived of the data.
6. The system must support to completely erase data in HDD and SDD respectively.

Vanish [1] is a system that self-destructs the data after a period of time. The keys are divided and stored in P2P system with the DHTs (Distributed Hash Tables). DHTs are reliable distributed storage where data is available in desired interval of time and discards the data older than a certain age. Thus vanish functions by encrypting the user's data locally with the a random encryption key that is not known to the user, destroys the local copy of the key and sprinkles the key bits (Shamir Secret Share [2]) over the DHT nodes that are randomly located. Since the keys are lost Sybil attack [3] helps by crawling each node and saving the keys. This can recover keys more than 99% of vanish messages. Vanish system uses VuzeDHT which is a public DHT for storing the keys.Vanish seems to be insecure in its current form because of some security issues in both OpenDHT as well as publicDHT. Hence the hybridDHT will also lead to security breaches and are very much susceptible to attacks.

To solve the problems faced by the Vanish system a new scheme is proposed called SafeVanish [4] which prevents hopping attack, a kind of Sybil attack. SafeVanish performs by extending the key length to some random range with some improvements in Shamir Secret Sharing [2] used in vanish system.

The use of P2P system is a major weakness for both Vanish as well as SafeVanish that leads to hopping attack (Sybil attack). Moreover in Vanish the survival time parameter is not in the hands of the user rather it is totally controlled by the DHT system. This paper introduces active storage system with object based storage structure with time constrained parameter that self-destructs the data when it goes out of control. Here the user is fully allowed to control the life cycle of the key by specifying the survival time of the key that in turn decides the life cycle of the private data on cloud.

FADE, proposed by Tang et al. [8] assuredly deletes the file on the cloud server and makes it unrecoverable and inaccessible to any third party. FADE implements time-based file assured deletion with fine-grained approach called policy-based file assured deletion.

3.2. Storing Data as Objects with Active Storage Technique

Active storage is an intelligent storage system that has become very popular in today's research area. For instance Wickremesinghe *et al.* [9] describes a new model for managing the load in the form of active storage units (ASU) which maximizes the processing capabilities by controlling the mapping of computational workload to the processing units. Similarly, MVSS (Multiview Storage System) [10], is a storage system for active storage devices that functions under a single framework for providing flexible migration of application code to storage devices. MVSS provides multiple ways for viewing a file similar to multiview in a database system.

Objects are primitive units of storage that can be directly accessed without passing through a server. This type of immediate and direct access offers maximization in performance. Devices that store objects are referred are called as object storage devices (OSD) [11]. Object based storage [12] offers great advancement in both storage devices as well as applications by increasing the functionalities of storage devices which seems to be far better compared to block based storage. Recently, many a system has entered into object storage environment such as, Panasas [13] and Ceph [14] that was developed and deployed under object-based technology. As it is easy to store and process data in object storage devices (OSD), people add more features in it that made these type of storage intelligent referred as "Intelligent storage" or "Active storage" [5]-[7].

3.3. Erasing Encryption Key bits

When a file is deleted or erased in SeDas, the bits of the encryption keys of those files are not totally gone until the area in the disk is overwritten or used by any other file. This situation gets even worse and complex in case of solid state drives (SSDs) because of its weird internal architecture [15].

In order to overcome such situations various methods are being employed for erasing the files reliably from hard disks like ATA or SCSI command, software tools and government standards. All these techniques have better capabilities to erase or delete files either single type or a drive full in an efficient manner. The ATA and SCSI have instructions that securely erase the files by sanitizing the whole disk.

As per the previous works there is no common use of self-destructing data system rather than some specific applications like multimedia, database etc., SeDas implements a fully functional prototype where series of experiments are carried out to look into its functionalities. Usage of SeDas has experimentally shown that it doesn't affect the normal storage of a system rather it allows self-destruction of data with user controlled survival time.

4. IMPLEMENTATION AND DESIGN OF SEDAS

4.1. SeDas Architectural Design

The architectural design of SeDas basically consists of three divisions based on active storage technique. Figure 3 describes the detailed view of SeDas. It mainly consists of i) Application client: The application client is nothing but the node that stores the services of SeDas system. This part in SeDas design has all the properties needed for a client node to interact with the metadata server in order it get its job done ii) Metadata Server: Metadata server is wholly responsible various managing services such as, User management, Server Management, Session Management, Key Management and File Management. iii) Storage node: The data in the storage node are actually stored in the OSD (Object Storage Device). The storage node basically has two subsystem for storing the data namely, <key,value> and active storage object (ASO) runtime. Data in the storage node are stored in <key,value> manner where key is the object ID on which one can perform operations like read/write object, lookup object and so on and the value denotes the associated data with its attributes. The ASO runtime subsystem manages active storage request for the user by mapping the appropriately.

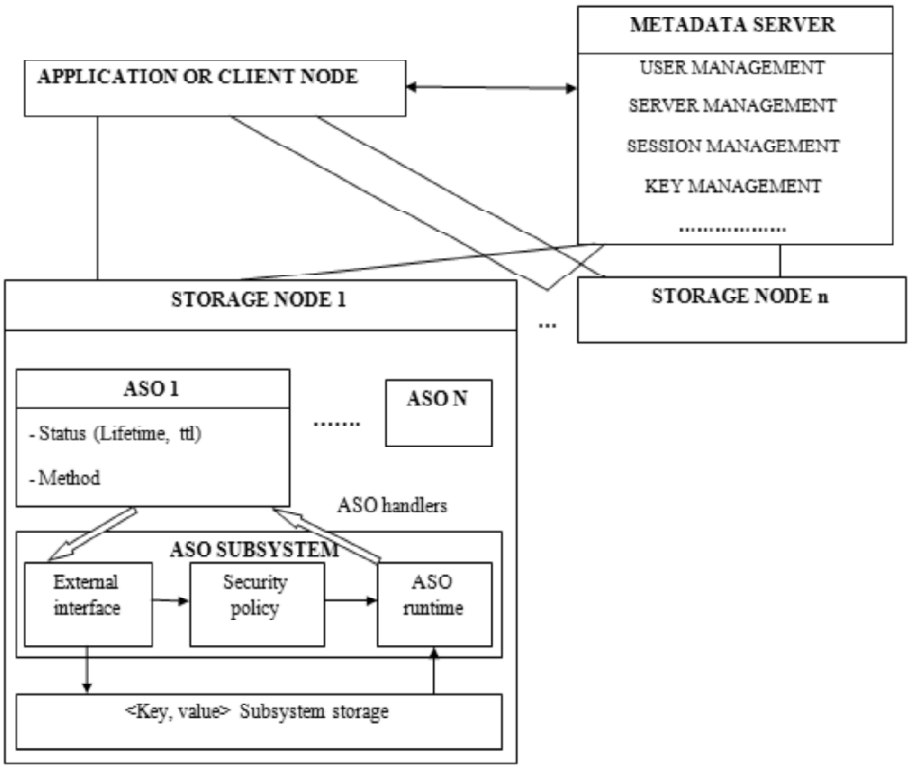

Figure 3: SeDas system Architecture

4.2. Active Storage Object

Active storage object (ASO) has ttl value associated with each data that is stored as user objects. The self-destruction mechanism is triggered by the ttl (Time-to-Live) parameter. This ttl value is decided by the user and it is activated based on the users wish. ie, the deletion of file is completely based on the value of the ttl which controls the time, until which the file/data can exist. After the ttl value expires the file/data is manually deleted as the user suggested.

4.3. Sharing the Keys in Storage Nodes

The client must first register itself with the server after which storage node also continues the registration with the metadata server. As a part of the client Advanced Encryption algorithm (AES) is used to generate keys for encryption and decryption of the file before uploading or downloading it .Once the keys are generated it has to be shared between the user application and the metadata server using Shamir Secret Sharing technique which enables distribution of data to N nodes.

4.4. Process of Data Transformation

SeDas system allows two phases of data processing from the client point of view

4.4.1 File upload process: User uploads the file to the storage system with arguments such as the file, key, and the ttl parameter that is required by the SeDas system. Figure 4 shows the pseudo-code for uploading a file. Here we consider the data and the key have been read from the file. The encryption of the file is done using Advanced Encryption Standard (AES) algorithm and the data is now split according to the size of the storage node. Now the key share generated by the Shamir Secret Sharing is used to upload the data in the form of Active Storage Object (ASO) in the storage node of the SeDas system.

4.4.2 File download process: Users with appropriate permission can download the data from the storage system. The technique for decrypting the data before it is used is implemented in user's application code.

Considering that the data has been read from the downloaded file, the client gets key shares from the SeDas system before decrypting it and this happens successfully if the self-destruct operation is not triggered. In case if the operation is triggered the key is destructed and the client cannot reconstruct it and can only read the encrypted data.

4.5. Model of Self Destruction Data System

Time constrained data process automatically destroys the file of which ever type it may be after the time period expires. The storage nodes don't seek permission from either the client/user or any third party maintainer to destroy the file. File with all cached copies are destroyed and the storage nodes are refreshed to hold a new file. The details of those files are also removed from the metadata server thus making it unavailable to the intruder's or any illegal access.

5. EVALUATION AND DISCUSSION

5.1. Applied Methodology

Though there are multiple services available for the user to store the data it is the responsibility of the SeDas system to protect the key and solve the problems created by third-party maintainer or any illegal access. Figure 5 shows the structure of user application interacting with SeDas server via SeDas client.

```
Procedure Upload File (data, key, TTL)
Data: data read from this file to be uploaded
Key: data read from the key
TTL: time-to-live of the key
Begin//encrypt the input data with the key
Buffer = Encrypt (data, key)
Connect to a data storage server;
if failed then return fail;
Create file in the data storage server and write buffer into it;
// use Shamir Secret Sharing algorithm to get key shares
// k is count of data servers in the SeDas system
Shared keys [1....k] = Shamir Secret Sharing Split (n, k, key)
For i from 1 to k then
Connect to DS[i]
If successful then create_object (shared keys[i], TTL);
Else
For j from 1 to i then
Delete key shares created before this one;
End for
Return fail;
End if
End for
Return successful;
End.
```

Figure 4: Pseudo-code for file Upload

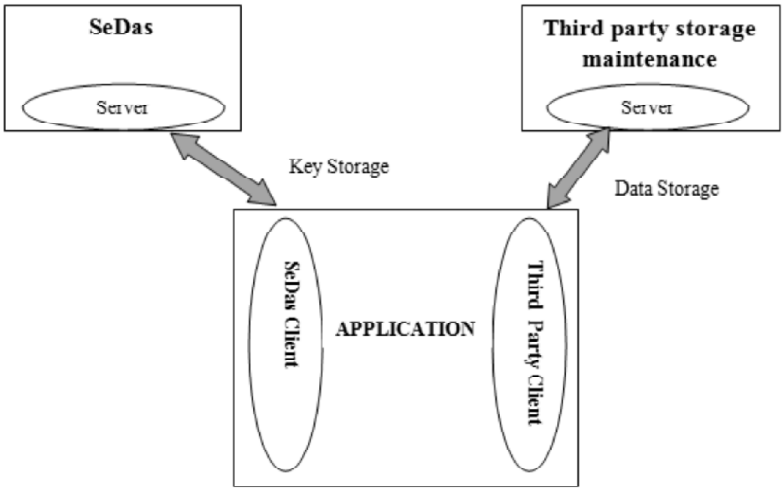

Figure 5: User Application Interacting with the Storage Server

Development of SeDas system is done using Spring framework which is basically an easy to use model for developing modern applications. Spring framework provides comprehensive platform developing software application. The Spring Framework consists of about 20 modules that are clustered into containers such as core container, data access/integration, web, AOP (Aspect Oriented Programming), Instrumentation, Messaging and Test. Code developed under spring framework and easily testable and are loosely coupled making it for easy maintenance. Data access/ integration layer consists of modules namely, JDBC Module, ORM Module, OXM Module where SeDas uses ORM for mapping the objects to the databases.

5.2. Evaluating the Performance

The performance evaluation of SeDas system shows that time taken for uploading and downloading a file has been greatly reduced in terms of throughput. Overhead is an important factor to be taken into consideration which degrades the performance particularly when it comes to encryption/decryption. Figure 6 shows clearly the overhead in encryption and decryption of various file sizes. SeDas system achieves this by improving the performance in terms of throughput. The major comparison is between the SeDas System with Active storage framework and the other is the traditional system without self-destructing mechanism (Native system). Figure 7 shows comparison in uploading files of different sizes. In this case, it takes 45 seconds for a file of size 16 MB in the native system and this is done with 22 seconds in the SeDas system. This happens same for all the other file sizes where the time factor has been reduced. Similarly in Figure 8 the comparison of downloading various file sizes is carried out where it has taken 7 seconds for downloading a file of size 16 MB in the native system and this has been reduced to 6 seconds in the SeDas system. The I/O process between the SeDas and the native system tells SeDas performs higher. It is shown that the throughput ie. Time taken for completing the operation of uploading/downloading of different file sizes has been achieved with low time when compared to the native system.

6. CONCLUSION AND FURTHER ENHANCEMENT

Securing the data in cloud environment has become an important task. User's data along with it copies and private keys are protected from falling into wrong hands with the help of SeDas. The paper even introduced combination of active storage framework of T10 OSD standard that greatly reduces computational task. Time constrained self-destruction has paved way for safe-guarding user's personal details like account

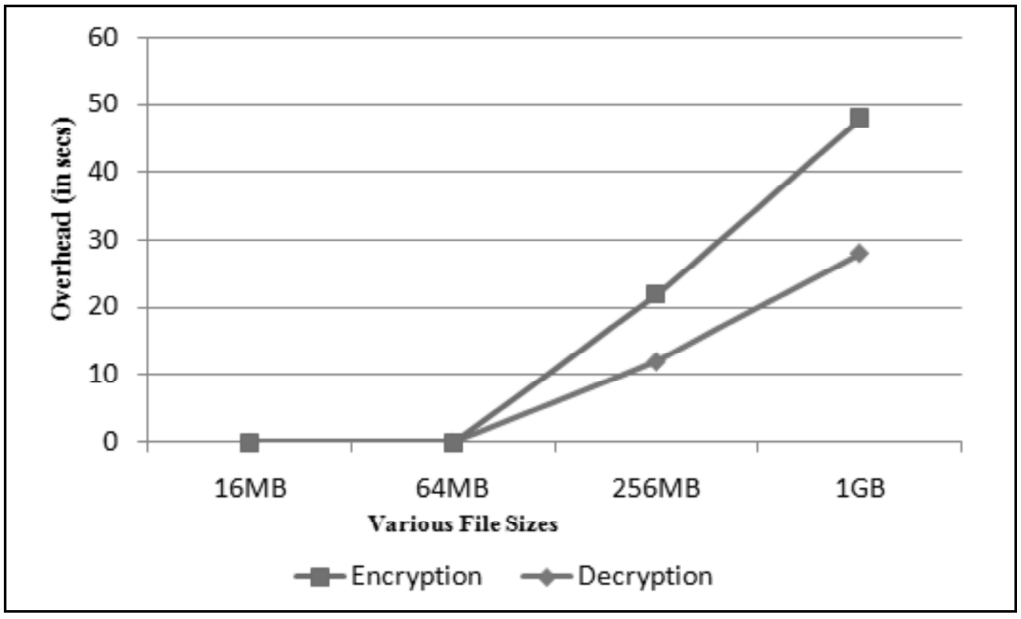

Figure 6: Overhead Comparison in Encryption and Decryption

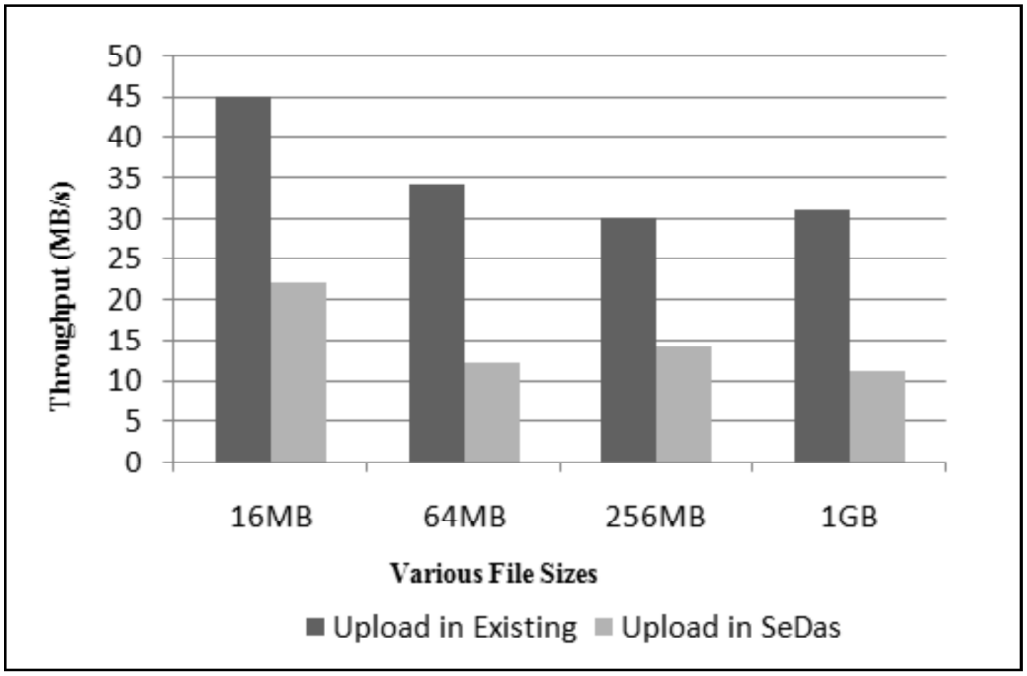

Figure 7: Throughput comparison in Uploading Various File Sizes

Figure 8: Throughput Comparison in Downloading Various File Sizes

number, password etc., by irreversibly self-destructing it with no action from the user point of view. Thus SeDas system makes sure that cloud environment is tied up with higher and uncompromisable security by offering reliable cloud services to its users.

Once the time to live factor expires the data with all the copies are destroyed. Sometimes the users are even forced to enter into situations where the files might be in need for their personal verification. In order to make this happen the file has to be recovered back. Though it is tedious to find the data back, with the help of some strong security procedures this could be enhanced by making the cloud services ahead in the near future.

REFERENCES

[1] R. Geambasu, T. Kohno, A. Levy, H. M. Levy, "Vanish: Increasing data privacy with self-destructing data," in *Proc. USENIX Security Symp.*, Montreal, Canada, pp. 299–315.2009.

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

[3] J. R. Douceur, "The sybil attack," in *Proc. IPTPS '01: Revised Papers From the First Int. Workshop on Peer-to-Peer Systems*. 2002.

[4] L. Zeng, Z. Shi, S. Xu, 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, pp. 521–528. 2010.

[5] L. Qin, 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, D. Feng, "An active storage system for high performance computing," in *Proc. 22nd Int. Conf. Advanced Information Networking and Applications (AINA)*, pp. 644–651.2008.

[7] Y. Xie, K. K. Muniswamy Reddy, D. Feng, D. D. E. Long, Y. Kang, Z.Niu, 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.

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

[9] R. Wickremesinghe, J. Chase, J. Vitter, "Distributed computing with load-managed active storage," in *Proc. 11th IEEE Int. Symp. High Performance Distributed Computing (HPDC)*, pp. 13–23. 2002.

[10] X. Ma, A. Reddy, "MVSS: An active storage architecture," *IEEE Trans. Parallel Distributed Syst.*, vol. 14, no. 10, pp. 993–1003. 2003.

[11] R. Weber, "Information Technology—SCSI object-based storage device commands (OSD)-2," Technical Committee T10, INCITS Std., Rev. 5. 2009.

[12] M. Mesnier, G. Ganger, E. Riedel, "Object-based storage," *IEEE Commun. Mag.*, vol. 41, no. 8, pp. 84–90. 2003.

[13] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, B. Zhou, "Scalable performance of the panasas parallel ûle system," in *Proc. 6th USENIX Conf. File and Storage Technologies (FAST)*. 2008.

[14] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, C. Maltzahn, "Ceph: A scalable, high-performance distributed ûle system," in *Proc. 7th Symp. Operating Systems Design and Implementation (OSDI)*. 2006.

[15] M. Wei, L. M. Grupp, F. E. Spada, and S. Swanson, "Reliably erasing data from flash- based solid state drives," in *Proc. 9th USENIX Conf. File and Storage Technologies (FAST)*, San Jose, CA, USA. 2011.