IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 6, Ver. VII (Nov – Dec. 2014), PP 48-54
www.iosrjournals.org

Time Controlled Cloud Environment with Self destructing Data system (SeDas) for Data Confidentiality

S. Savitha[1], Dr. D. Thilagavathy[2]
[1]*Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, Tamil Nadu, India*
[2]*Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, Tamil Nadu, India*

Abstract: *Popularization of cloud technology and day to day usage of mobile Internet has become very common in today's fast moving world where people are subjected to post their personal information like account numbers, passwords, notes, and different vital data. This information are cached, copied, and archived by Cloud Service Providers (CSPs), typically for users' authorization and management. There are high chances for the data to fall into wrong side usage where it could be accessed illegally without the users' knowledge. In such situations security for the data on cloud has to be increased. Self-destructing information principally aims at protecting the users' data confidentiality. All the data and their copies become destructed or unclear after a user-specified time, with no intervention from the user part. SeDas system meets this challenge by a new combination of cryptographic techniques and active storage framework based on T10 OSD standard. These security procedures and its functionalities make sure that SeDas meets all privacy-preserving policies and is easy for practical use. Compared to the system without SeDas the performance for uploading/downloading files has been achieved better.*

Keywords: *active storage, cloud computing, cloud service providers, cryptographic techniques, data confidentiality, self-destruction data system.*

I. Introduction

With development of Cloud computing and popularization of mobile web, Cloud services are getting a lot of vital for people's life. People are more or less requested to submit or post some personal non-public data to the Cloud over Internet. Once people try this, they subjectively hope service suppliers can give security policy to safeguard their knowledge from leakage, thus others people won't invade their privacy. As people trust a lot on the net and Cloud technology, security of their privacy is more at risk. On the one hand, once knowledge is being processed, reworked and keep by the present automatic data processing system or network, systems or network should cache, copy or archive it. These copies are essential for systems and therefore the network. However, people don't have any data regarding these copies and can't manage them, thus these copies could leak their privacy. On the opposite hand, their privacy can also be leaked via Cloud Service suppliers (CSPs') negligence, hackers' intrusion or some legal actions. These issues provide formidable challenges to safeguard people's privacy. Fig. 1 shows the outer view of these services and interaction between the communicating parties.

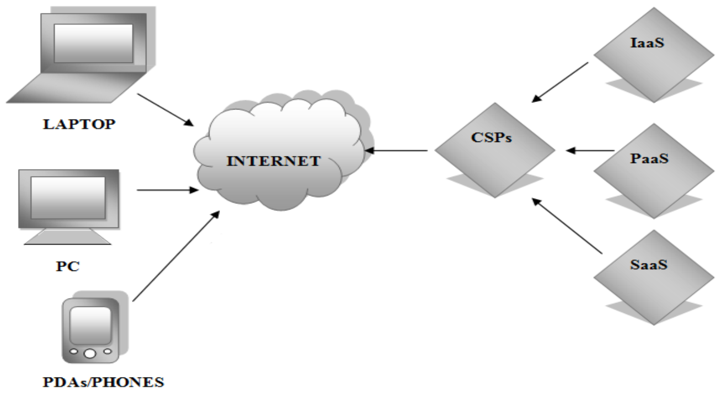


Fig. 1. Overview of Cloud Services

Vanish [1] provides a replacement plan for sharing and protecting privacy. Fig. 2 shows Vanish system architecture and its functionalities. Within the Vanish system, a secret is divided and stored in a P2P system with distributed hash tables (DHTs). With connection and exiting of the P2P node, the system will maintain secret keys. Consistent with characteristics of P2P, after eight hours the DHT can refresh each node. With Shamir Secret Sharing algorithm [2], once one cannot get enough components of a key, he won't decipher knowledge encrypted with this key, which suggests the secret is destroyed.

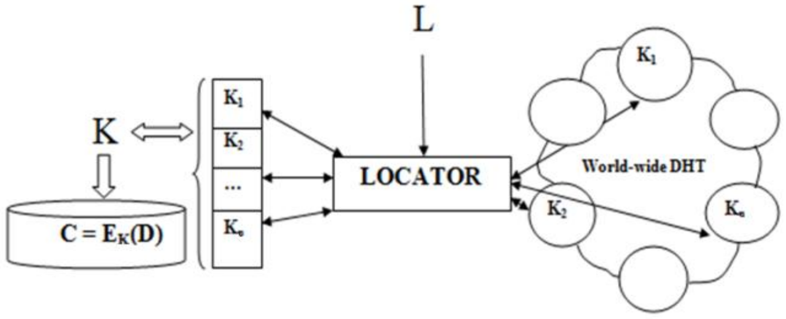


Fig. 2. Vanish system Architecture

Some special attacks to characteristics of P2P are challenges of Vanish [3], uncontrolled in how long the key will survive is additionally one in every of the disadvantages for Vanish. In considering these disadvantages, this paper presents an answer to implement a self-destructing system, or SeDas, that is predicated on a full of active storage framework [5]–[6]-[7]. The SeDas system defines two new modules, a destroy methodology object that's related to every secret key and survival time parameter for every secret key. During this case, SeDas will meet the necessities of self-destructing knowledge with manageable survival time whereas users will use this method as a general object storage system. Our contributions are summarized as follows

1) We concentrate on the connected key distribution rule, Shamir's rule [2], that is employed because the core rule to implement users distributing keys within the object storage system. We tend to use these ways to implement a security with equal divided key (Shamir Secret Shares [2]).
2) Based on active storage framework, we tend to use associate object-based storage interface to store and manage the equally divided key. We tend to enforce a proof-of-concept SeDas paradigm.
3) Through practicality and security properties analysis of the SeDas paradigm, the results demonstrate that SeDas is sensible to use and meets all the privacy-preserving goals. The paradigm system imposes moderately low runtime overhead.
4) SeDas supports security erasing files and random coding keys keep in a magnetic disk drive (HDD) or solid state drive (SSD), severally.

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

II. Related work

2.1 Data Self-Destruction

The self-destructing information system within the Cloud setting ought to meet the subsequent requirements: i) the way to destruct all copies of knowledge at the same time and build them undecipherable just in case the information is out of control. ii) A local data destruction approach won't help the Cloud storage because of the amount of backups or archives of the information that is hold on within the Cloud is unknown, and a few nodes conserving the backup information are offline. iii) No explicit delete actions by the user, or any third-party storing that information should occur. iv) No need to modify any of the data or archived copies of that data; iv) Support for securely erasing data in both HDD and SSD.

Tang et al. [8] proposed FADE that is made upon common cryptographic techniques and assuredly deletes files to make them forgotten to anyone upon revocations of file access policies. Wang et al. [9] utilized the general public key based mostly homomorphism critic with random mask technique to deliver a privacy-preserving public auditing system for Cloud knowledge storage security and uses the technique of an additive combination signature to support handling of multiple auditing tasks. Perlman et al. [10] presents three kinds of assured delete: expiration time noted at file creation, on-demand deletion of individual files, and custom keys for categories of knowledge.

Vanish [1] may be a system for making messages that mechanically self-destroy when an amount of given time. It integrates special techniques with global-scale, P2P, distributed hash tables (DHTs): DHTs discard information older than a definite age. Vanish works by encrypting every message with a random key and storing shares of the key during a period of time, public DHT. However, Sybil attacks [3] might compromise the system by endlessly crawling the DHT and saving every key with data on worth before it ages out. This will efficiently recover keys for over ninety nine of Vanish messages. Wolchok et al. [3] concludes that public DHTs like VuzeDHT in all probability cannot give robust enough security for Vanish. So, Geambasu et al. [11] proposes two main countermeasures.

Although using each OpenDHT [12] and VuzeDHT may raise the bar for a wrongdoer, at the best it will offer the most security derived from either system: if each DHTs are insecure, then the hybrid also will be insecure. Vanish is a remarkable approach to a crucial privacy downside, but, in its current type, its insecure [3].

To address the matter of Vanish mentioned higher than, in the previous work, tendency to project a brand new theme has been done known as SafeVanish [4], to avoid hopping attack, that is one reasonably the Sybil attacks, by extending the length of the key shares, and did some improvement on the Shamir Secret Sharing algorithm [2] enforced within the Vanish system. Also, the tendency to give an improved approach against sniffing attacks by method of using the general public key cryptosystem to avoid sniffing operations.

However, the employment of P2P options still is a fatal weakness each for Vanish and SafeVanish, because there is a specific attack against P2P ways (e.g., hopping attacks and Sybil attacks [3]).

In addition, for the Vanish system, the survival time of key attainment is set by DHT system and not governable by the user. Supported active storage framework, this paper proposes a distributed object-based storage system with self-destructing information operation. The system combines a proactive approach within the object storage techniques and methodology object, exploitation processing capabilities of OSD to attain information self-destruction. User will specify the key survival time of distribution key and use the settings of expanded interface to export the life cycle of a key, permitting the user to manage the subjective life-cycle of personal information.

2.2 Object-based storage 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. [13] 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) [14], 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) [15]. Object based storage [16] 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 [17] and Ceph [18] 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].

2.3 Erasing total bits of encryption key

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 [19].

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.

III. Design and Implementation of SeDas

3.1 System Architecture of SeDas

Fig. 3 shows the architecture of SeDas. 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 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. The object ID is used as a key. The associated data and attribute are stored as values. The ASO runtime subsystem based on the active storage agent module in the object-based storage system is used to process active storage request from users and manage method objects and policy objects.

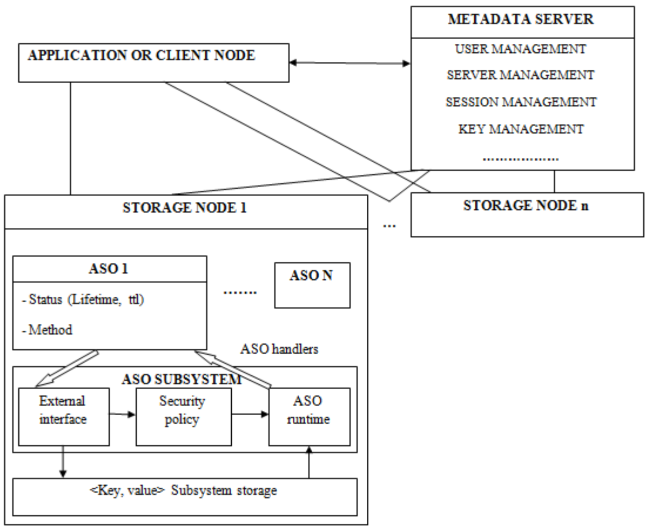


Fig. 3. SeDas system Architecture

3.2 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.

3.3 Self-Destruct Method Object

Generally, kernel code can be executed efficiently; however, a 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 can be used to develop software in user space. It is much 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 code of a service method in user space can take advantage of performance of the system. The system might crash 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 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 destructed.

3.4 Key Sharing in storage node

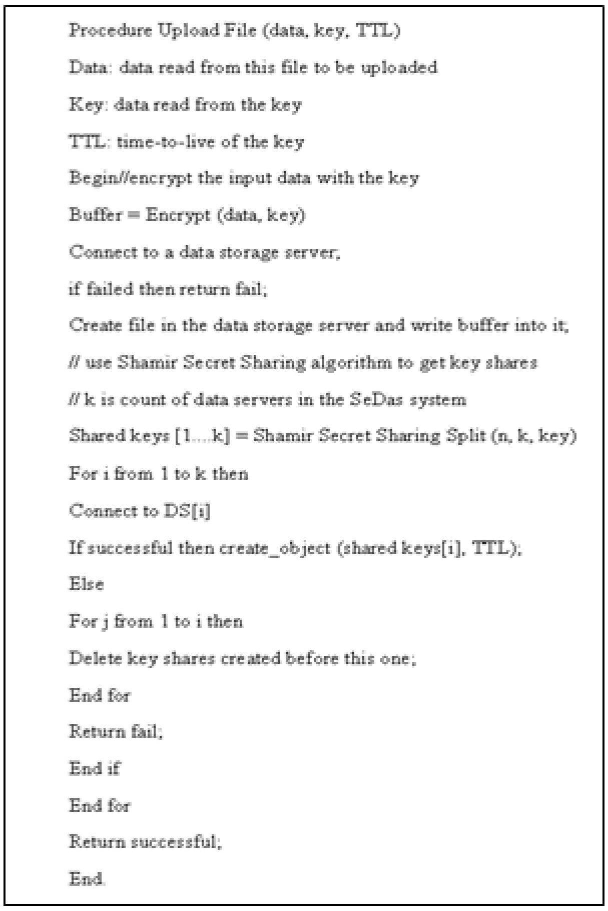
The client must first register itself with the server after which storage node also continues the registration with the metadata server. As a part of the client Advanced Encryption algorithm (AES) is used to generate keys for encryption and decryption of the file before uploading or downloading it .Once 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.

3.5 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 processes: uploading and downloading.

3.5.1 Process of uploading files: 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 arguments for the uploading procedure. Fig. 4 presents its pseudo-code. In these 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 ShamirSecretSharing algorithm will be used to create active storage object (ASO) in storage node in the SeDas system.

3.5.2 Process of downloading files: 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.



```
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
```

Fig. 4. Pseudo-code for file Upload

IV. Evaluation and Discussion

In this section, we tend to discuss take a look at methodology and implementation for SeDas then offer analysis on the take a look at result. We tend to place up an information storage file system supported pNFS in virtual machine surroundings to implement the take a look at for file uploading, downloading and sharing.

4.1 Methodology

There are multiple storage services for a user to store knowledge. Meanwhile, to avoid the matter made by the centralized "trusted" third party, the responsibility of SeDas is to guard the user key and supply the function of self-destructing knowledge. Fig. 5 shows the brief structure of the user application realizing storage method. During this structure, the user application node contains two system clients: any third-party knowledge storage system (TPDSS) and SeDas. The user application interacts with the SeDas server through SeDas' client, obtaining knowledge storage service.

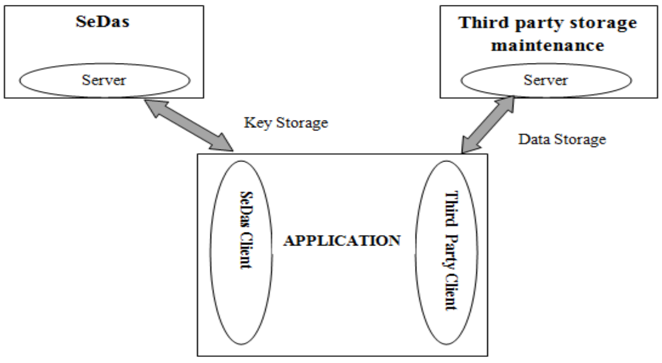
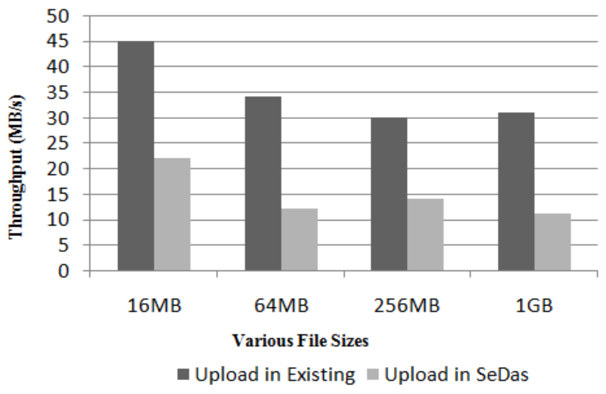


Fig. 5. User application interacting with the Storage server

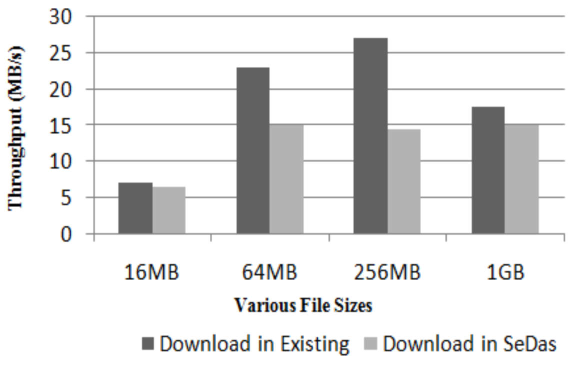
Development of SeDas system is done using Spring framework which is basically an easy to use model for developing modern applications. Spring framework provides comprehensive platform developing software application. The Spring Framework consists of about 20 modules that are clustered into containers 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.

4.2 Evaluation

The performance evaluation of SeDas system shows that time taken for uploading and downloading a file has been greatly reduced in terms of throughput. One is the SeDas System with Active storage framework and the other is the traditional system without self-destructing mechanism (Native system). As shown in the Fig. 6(a) comparison happens between two systems. In case of uploading a file 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 Fig. 6(b) 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.



(a)



(b)

Fig. 6. Comparisons of throughput in the upload and download operations.

V. Conclusion and Further Enhancement

Data privacy has become progressively vital within the Cloud surroundings. 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 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.

As a part of future work, the users are provided with still more advanced features for managing the files on the cloud server even after destruction. 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 hopefully made more easy and enhanced by making cloud services ahead in the near future.

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] 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, 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] Y. Xie, K. K. Muniswamy, D. Feng, D. D. E. Long, Y. Kang, Z.Niu, Z.Tan, Design and evaluationof 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 file assured deletion, in Proc. SecureComm, 2010.

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

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

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

[12] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, H. Yu, OpenDHT: A public DHT service and its uses,” in Proc. ACM SIGCOMM, 2005.

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

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

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

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

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

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

[19] M. Wei, L. M. Grupp, F. E. Spada, 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, Feb. 2011.

S. Savitha was born at Hosur, Tamil Nadu, India. The author received Bachelor Degree in the field of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, Tamil Nadu, India in the year 2013.

She is currently pursuing her Master of Engineering in the field of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, Tamil Nadu. The author’s areas of interest and research work involve Cloud Computing, Big Data, Network Security and Mobile Security. She is an active member in CSI

Dr. D. Thilagavathy M.E., Ph.D was born at Salem, Tamil Nadu, India. The author received Master degree in the field of Computer Science and Engineering, Sona College of Engineering, Salem, Tamil Nadu, India in the year 2004 and obtained the Ph.D degree from Anna University, Chennai, Tamil Nadu, India in the year 2012.

She is currently working as professor & head in the Department of Computer Science and Engineering in Adhiyamaan College of Engineering, Hosur, Tamil Nadu, India. She is having an experience of about 15 years and published about 20 papers in various international journals and has presented about 30 papers in national conferences. Her area of interest includes Key Agreement, Key Distribution in Network Security, Information Security and Mobile Security. She a life time member in ISTE.IE (I), CSI.