ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 12, December 2013

Declaring Time Parameter to Data in Active Storage Framework

Backya S, Palraj K

***Abstract*— Cloud computing is the notion of outsourcing on-site available services and data storage to an off-site. Data stored in cloud contains personal information and that could be used by unauthorized person. This is due to the cached copy available at the cloud service provider cache memory. Self destruction method protects data's privacy by sanitizing the data after its usage. Both the confidential data and its copies are destroyed and unreadable after certain user's specified time. Key used for encryption and decryption also gets vanished. In this paper, active storage framework provides virtualization environment to run client's application and data is treated as objects to increase the throughput and decrease the latency.**

***Index Terms*— confidential data privacy, object based active storage, self-destructing data, Vanishing data.**

I. INTRODUCTION

Cloud computing is mainly used to solve the storage and maintenance problem. Cloud storage offers online storage and accesses it from anywhere and at any time. Few of the services in cloud render by service providers. In private cloud they have their own storage area. File lost problem is solved by the file backup process in peer to peer system. In the distributed system archived, cached, copies of the file is available at many peers. Any of the peers can act as a server to the service providers. So the copies are available forever after downloaded the required confidential file. The copies resides in the peer are in readable form. Here, limited the excessive amount or replication is required. Users are unaware of those copies available at the cache memory of the service providers. They cannot have control over the data. Such copies of the file are maintained by the service provider against accidental, legal and malicious attacks. In P2P system secret key is stored with distributed hash table (DHT). In distributed hash table it must be ensure that key actually stores the data associated with the key in each node. Routing attacks, storage and retrieval attacks violates data privacy in DHT. Sybil attacks create the fake entities and gains reputation from the honest entities.

Manuscript received Nov, 2013.

***Backya S**, Department of Computer Science and Engineering, Sri Vidya College of Engineering and Technology, Virudhunagar, India, +91 9487576079*

***Palraj K**, Department of Computer Science and Engineering, Sri Vidya Collge of Engineering and Technology, Virudhunagar, India, +91 8508259397.*

For sharing the files and protecting privacy the concept called vanish is introduced. Vanish encapsulates the file with the pre-defined timeout. It is resistant to the attacks in vuzeDHT which is a centralized system. This is achieved by encrypting data with the random symmetric key and the key is not revealed to the user. Key is broken into multiple pieces and sprinkled across random nodes; distribute the key across the randomly chosen node in peer-to-peer system and finally the needed information for key retrieval is gathered to retrieve the key pieces with the encrypted data. Vanish system prevents the key pieces to be retrieved after the specified timeout.

The proposed concept in this paper is self-destructing data. There is an extensible framework for integrating multiple key-storage mechanism into a single self-destructing data system. It has the different key storage approaches to provide security against the attacks. In self-destructing data system all copies of the data permanently unreadable at the user specified time. Goals for self-destructing data are an attacker retroactively obtains a copy of the data and any relevant cryptographic keys from before the timeout there is no use because the specific data and its key are destroyed automatically, without the use of any explicit delete action by any parties involved the data disappear by its own, without need to modify any of the copies of the data, without the use of secure hardware and without relying on new and trusted external services, it provides the receiver with the minimum knowledge needed to consume the data. Our prototype attempts to meet these goals through the use of various cryptographic techniques. To increase the processor performance and to reduce the cost, object based storage system is needed. Object storage disk provides interaction between the operating system and the storage system in the abstraction level. It separates the data and metadata to increase the throughput. It performs computation in parallel using distributed storage nodes. In active storage portion of the application is run directly on the hard disk. It reduces the data traffic and increases the processing speed.

I. RELATED WORK

In this section, we discuss related works on self-destruction of data and object based active storage. DHT [1] implements the services of the remote hash table and provides internal coordination among the nodes in peer-to-peer network. Some of the properties are scalable, distributed, decentralized and self-managing. Adeona a privacy preserving method for mobile devices and vanish a method for creating self-destructing data. DHT [7] is exposed to attacks such as

byzantine attacks, node crawling attacks and information harvesting attacks. Active Storage unit increase the process capability, reduces the data movement while searching and possible to read, write and execute the data directly on active storage. It improves host processing performance. It offers local control over the data. It checks the availability of resources and maintains the load. Computational process depends on hardware capabilities and load condition in active storage. The disadvantages are; large scale storage system aggregates many disks [2]. It has the problem of bandwidth limitation.

Object based active storage framework is built upon iSCSI OSD standards. Many data-intensive applications have small CPU and memory requirements and are attractive for execution across Active Disks. Implementing this in the form of objects allows for better portability, reusability of components, extensibility and other such advantages of object oriented programming. Object-based Storage devices have the capability of managing their storage capacity and shows file-like storage ”objects” to their hosts. These objects behave exactly like files. They can be created and destroyed and can grow and shrink their size during their lifetimes. The idea of moving the portion of application and make it to run directly on the disk. It has to be done to reduce data traffic. The client side of our application was designed to have APIs for multiple operations on an object that can exploit the parallelism offered by such a framework. The problem here is framework which does not supports any kind of objects. Currently the application supports only the list objects. User objects are placed in partitions that are represented by partition objects [3]. The average sort time is the average time required on each target node to do the read from disk and then sort. Storage administration costs will be higher than the cost of the storage systems themselves.

Vanish a system for creating messages that automatically destroy data after a period of time. It encrypts the message using the random key. Key shares are stored at the public distributed hash table. The hash table contains name, value pair. DHT deletes data and its key shares. Data is permanently available in encrypted form. Vanish encapsulates the data object so the data is destroyed automatically. Length of the key shares depends on the key length [4]. The problem in the existing system is deployed vanish implementation is insecure. Here is a possibility of Sybil attacks. Attack occurs in the distributed hash table. Attacker able to hack the key before it ages out. In Sybil attacks the attacker pretends like user. Sybil attack is more expensive. In public DHT any peers act as a server. User trusted that vanish deletes data. So they are not deleting the sensitive data periodically [5]. Deployment of vanish does not provide the assured automatic deletion. Attacker cannot able to understand the user’s traffic towards the DHT. Vuze nodes replicate nearby 20 nodes. The solution to the above mentioned problems are using Shamir secret key it breaks the key into many n shares. If we recover the n-1 part of the key then only we can recover the data. It stores the key in the random indices. It avoids hacker to hack the entire part of the data. Vanishing data object retrieves the original plain text before expiration. It reduces the replication in peer to peer system. So the attack cost is more expensive in Sybil with the new implementation of vanish. Open DHT is introduced to run the private data. Third party is used to prevent the threat. Attacker cannot able to read more than the small fraction of the data.

MVSS is a storage system for active storage framework. MVSS offers a single framework for supporting various services at the device level. It provides a flexible interface for associating services to the file through multiple views of the file. Views in MVSS are generated dynamically and not stored in the physical storage. MVSS represents each view of an underlying file through the separate entry in the file system namespace. [6] MVSS separates the deployment of services from file system implementation and thus allows services to be migrated to the storage devices. Improve the storage devices performance, functions and characteristics by migrating services to disks. Direct network attachment is suggested to enable data transfer from device directly to the client rather than the server. The main advantage of disk storage is parallelism among disks and more aggregate CPU power at the disk than at the server. Active disks have the ability to reduce the bandwidth demands. The most difficult problem to implement the active storage is the migration problem. File system access data by block level. The solution to the above mentioned problem is different host level interfaces to accommodate high level services. It supports the heterogeneous platform, reusing of the existing file system and the operating system technology. It allows the application to access the new services transparently. It minimizes the changes to the operating system. MVSS supports the virtual file. Virtual file is the combination and its associated services. Each virtual file is stored in different virtual disk. Virtual disk facilitates the namespace distinctions of different views of the file, provides the solution for the caching problem.

II. SECURED KEY STORAGE AND AUTOMATIC DELETION

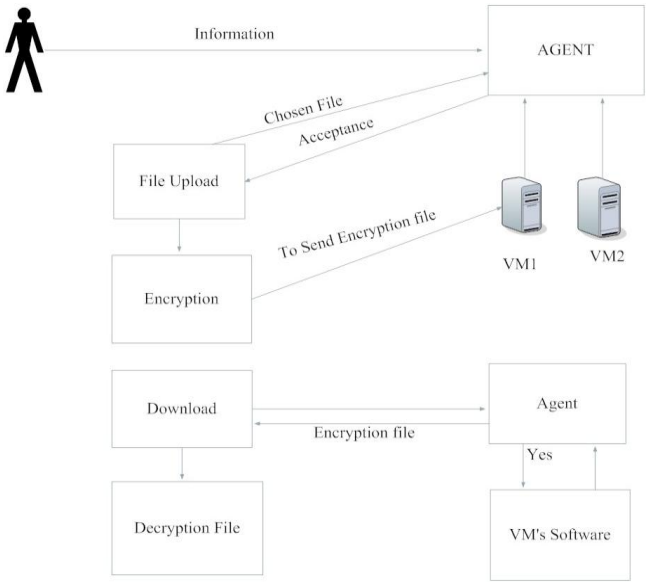

Fig. 1. Data Expiration Process

From the Fig.1 confidential file has chosen by the uploader of the file. The file chosen by the uploader alone stored. in the cloud because in the private cloud service providers offers few services which are needed by the users. The file is divided and stores each portion of the file in different virtual machine. Virtual machine acts as an active storage and randomly generated key for each partition. The key and file portion is associated with the corresponding uploader using the overlay network. Now the encrypted file is stored in the virtual machine. There is no storage in the agent. So it is not necessary to trust the third party. It just verifies the authorized access by the user or not. At the time of downloading the file verification procedure is done by the agent and guarantees the user to access the service. Otherwise service is denied. Along with the key the survival parameter of the key is defined. It provides the lifespan for the file. Until the key is alive user downloads the file. It facilitates the file to be more confidential. Procedure to upload the file is first to create the account and obtain the unique identity from the service provider. There is no need to convert the format of the file in cloud storage.

A. Active storage framework

In active storage framework object based interface to store and manage the equally divided key. Evaluation of self-destructing data based on its functionality and security policies. The result demonstrates that self destructing data meets all the privacy preserving goals. Object based interface achieves coordination between application server and storage devices.

B. Self data manipulation

There is no need of explicit delete actions by the user, or any other third- party storing that data. No need to modify any of the stored or archived copies of that data. The time of key attainment is determined by DHT system and not controllable for the user. In distributed object-based storage system self destructing data function is used. Extensive experiments show that the proposed Self-destructing data system does not affect the normal use of storage system. The requirements of self-destructing data under a survival time by user controllable key.

C. Time parameter for secret key

A service method needs a long time to process a complicated task. A service method in the user space can take advantage of performance of the system. All the data and their copies become destructed or unreadable after a user-specified time. A self-destruct method object is associated with each secret key part and survival time parameter for each secret key part. A system creates messages that automatically self- destruct after a period of time. The user's applications should implement logic of data process and act as a client node. Once it meets the time constraints the data is deleted.

III. DESIGN

In object based storage when client application request metadata for one file from the metadata server it provides the mapping functions and it is cached for future reference. It improves the throughput and reduces its latency. Mapping function consists of data associated to the particular file. Object based storage designed to achieve cost effective scalable bandwidth. Objects are stored on disk location. Metadata server performs prefetching so that the related objects are available in the cache memory. In metadata server client application gets permission to perform any operation. MAC is used to avoid replay attack. The response from the metadata is in the form of array. It reduces the prefetching overhead. File length is updated to the metadata server and hence it reflects the existing object based storage.

OSD represents files as the set of objects and distributed across various storage devices. Common object size is the size of the stripe unit size of the system. Object based file system improves the disk utilization. It works well on workloads of both small and large workloads. OSD provides object level interface to the files. Any client application can directly contact the OSD retrieve the objects regarding the related files. Providing direct data transfer between the storage device and client improves bandwidth. High level of security is achieved by certain cryptographic techniques and using security mechanism. Client interface manages the file system cache memory. File is divided into fixed size objects and in distributed manner. The asynchronous write operation in object based storage is cached. File lost problem due to power failure and hardware failure can be avoided by the cached copies. Object identifier used to retrieve object from disk.

In the disk location OBFS determines which in type of block object is to be stored. Object size is more than the disk utilization threshold of large blocks then large blocks are used or else small block size is used. It updates data structure asynchronously for better performance. From the user object active storage object is derived and time to live parameter is defined. This parameter checks the policy associated to the object and meets the constraints defined to those policy. The implementation of the data process includes two processes such as file uploading and downloading.

A. File upload

At the time user uploads the file they happen to store the file and key to encrypt in the object based storage system. Intermediate verifies the authentication of the user in the private cloud. To increase the privacy activation is banned for two user upload the file with same details. The input given by the user is the file, file size, time to live for the secret key and the number of parts the file to be divided. As specified by the user the virtual machine is created for each file partition. Same configuration of the virtual machine is created at each and every node. File is divided and stored in each virtual machine. Now the secret key is generated by the virtual machine. In the storage node Shamir secret sharing algorithm provides this splitting process.

B. File download

The user who has the access permission to access the file alone downloads the content of the file. Before downloading the decryption process is performed. For decryption client gets the key from the object storage. The storage system

reconstructs the key but the downloader cannot able to reconstruct the key. So the decryption is performed by the storage system. Next the file expiration condition verification begins.

C. Secure Erasable Function

Secure delete function is carried out in both read and write operation in OSD. It contains set of commands to completely overwrite all of the on the hard drive. Previous data cannot be accessible due to secure delete function. It performs sanitizing without affecting the disk capabilities.

IV. RESULT

Here it an evaluation of latency for the files of different sizes. The time taken for uploading and downloading file determines the latency. In this system during file access there is high speed from the first bit to last bit arrival rate of the file. It is evaluated with the file size. Downloaded time is divided according to the file size and makes it to the multiplication of the bytes. To the previous result overhead of sender and receiver is added.

V. CONCLUSION

A novel framework for automatic deletion of data to preserve the privacy of the confidential data is proposed in this paper. We demonstrated the approach of encrypted storage of data without explicit actions by user and provide the minimum knowledge to consume the data at the receiver side. It prevents the illegal use of the confidential data. Hence we conclude that less flexibility at the receiver side improves security in object based storage system.

VI. FUTURE WORK

Data and its associated key are destroyed after the expiration time of the key. Data such as file is treated as objects. When a single object is destroyed then any object referenced to that particular object gets free. Thus the reference count decreases. Instead of destroying an object as soon as its reference count falls to zero, it is added to the unreferenced list objects and periodically destroyed from the list.

VII. REFERENCES

[1] Lingfang Zeng, Shibin Chen, Qingsong Wei, and Dan Feng "SeDas: A Self-Destructing Data System Based on Active Storage Framework", *IEEE transactions on magnetics, vol. 49, no. 6, june 2013*

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

[4] 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.

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

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

[8] A. Acharya, M. Uysal, and J. Saltz, "Active disks: Programming model, algorithms and evaluation," in *Proc. 8th Conf. Architectural Support for Programming Languages and Operating System (ASPLOS)*, Oct. 1998, pp. 81–91.

[9] 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.

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

[11] T. Cholez, I. Chrisment, and O. Festor, "Evaluation of sybil attack protection schemes in kad," in *Proc. 3rd Int. Conf. Autonomous Infrastructure, Management and Security*, Berlin, Germany, 2009, pp. 70–82.

[12] Y. Lu,D.Du, andT.Ruwart, "QoS provisioning framework for an OSD based storage system," in *Proc. 22nd IEEE/13th NASA Goddard Conf. Mass Storage Systems and Technologies (MSST)*, 2005, pp. 28–35.

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

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

[15] 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.

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

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