SECURE DATA ACCESS USING BIOMETRIC RECOGNITION IN CLOUD COMPUTING

[1]P.NIVETHA , [2]V.SARALA

Abstract-The personal data stored in the cloud may contain account numbers, passwords, notes, and other important information that could be used and misused by a miscreant, a competitor, or a court of law. These data are cached, copied, and archived by Cloud Service Providers (CSPs), often without user's authorization and control. Self-destructing data mainly aims at protecting the user data's privacy. All the data and their copies become destructed or unreadable after a user-specified time, without any user intervention. In addition, the decryption key is destructed after the user-specified time. In the proposed system, SeDas, a system that meets this challenge through a novel integration of cryptographic techniques with active storage techniques based on T10 OSD standard. In the proposed system, user's biometric information are encrypted and stored in a database. Batch homomorphic encryption technique is used to encrypt the biometric information provided by user biometric information and are compared with the database where the authenticated user's encrypted information are stored. User's fingerprints are encrypted by batch homomorphic encryption technique. The biometric information is decrypted and the user is subscribed to access the cloud. This decides whether the user is approved or denied. Once the user is recognized as an authorized user, the user can view, upload or download the data stored in cloud.

Keywords –Cloud computing, Active storage, Self destruction data, Data privacy

---

1 INTRODUCTION

The personal data stored in the cloud may contain account numbers, passwords, notes, and other important information. These data are cached, copied and archived by cloud service providers, often without user's authorization and control. Self-destructing data mainly aims at protecting the user data's privacy. By using biometric recognition, the data can be secured by encryption and decryption of data.

- ***Nivetha.P*** *is currently pursuing masters degree program in computer science engineering. Ph.9952854091*
- ***Sarala.V****, M.E., Assistant Professor of CSE Dept, ph-9444194449*

A cloud computing system is divided into two sections: the front end and the back end. They connect to each other through a network, usually the Internet. The front end is the computer user, or client[1]. The back end is the "cloud" section of the system. The front end includes the client's computer (or computer network) and the application required to access the cloud computing system. On the back end of the system, there are various computers, servers and data storage systems that create the "cloud" of computing services. A cloud computing system is any computer program that can be imagined from data processing to video games. Usually, each application will have its own dedicated server.

2 RELATED WORK

The increasing performance and decreasing cost of the processors and memory are causing system intelligence to move from the CPU to peripherals such as disk drives. Storage system designers are using this trend towards excessive computation capability to perform complex processing and optimizations directly inside the storage devices. Such kind of optimizations has been performed only at low levels of the storage protocol. At some point, additional processing power can be had at negligible cost[4]. The question then becomes simple where to place this computation power in a system to support the widest range of tasks efficiently. The contention of this work is that processing power is already moving into peripheral devices and that applications can achieve significant performance gains by taking advantage of this trend.

Storage Class Memory (SCM) has become increasingly popular in enterprise systems as well as embedded and mobile systems. However, replacing hard drives with SCMs in current storage systems often forces either major changes in file systems or suboptimal performance, because the current block-based interface does not deliver enough information to the device to allow it to optimize data management for specific device characteristics such as the out-of-place update. To alleviate this problem and fully

utilize different characteristics of SCMs, the use of an object-based model[7] has been proposed. This provides the hardware and firmware and the ability to optimize performance for the underlying implementation, and allows drop-in replacement for devices based on new types of SCM. The design of object-based SCMs was developed to implement an object-based flash memory prototype.

A storage system for active storage devices: MVSS offers a single framework for supporting various services at the device level. It provides a flexible interface for associating services to file through multiple views of the file. Similar to views of a database in a multi view database system, views in MVSS[10] are generated dynamically and are not stored on physical storage devices. MVSS represents each view of an underlying file through a separate entry in the file system namespace. MVSS separates the deployment of services from file system implementations and, thus, allows services to be migrated to storage devices. The paper presents the design of MVSS and how different services can be supported in MVSS at the device level.

Disk storage densities have been growing and storage systems are accumulating larger amounts of data. Storage is network-based and shared by many applications including parallel applications accessing storage through high-speed interconnects such as SANs, Infinite band, or Ethernet networks at gigabit speeds. These interconnects allow shared access to distributed storage across large network sites, or even across multiple sites in computational grids spanning administrative domains[2]. These large-scale decentralized storage systems are a key to address big Data computational challenges in the future. A key goal of the approach is to enable load managed active storage. This approach exposes primitive computation units and their costs to the system, enabling the system to control the mapping of computational workload to processing units in order to maximize global system performance. Functions perform bounded computation as a side effect of I/O access; the mapping of functions to ASUs and hosts is configurable and potentially dynamic.

2.1 PROPOSED WORK

Novel approaches for biometric systems (NABS) framework design, typically characterize any biometric application setting: 1) the biometry set: to exemplify a possible instantiation of NABS framework, combine face, ear, and fingerprint biometrics. 2) the normalization method: each system may return results using different dimensionalities and scales; a normalization function is proposed that provides good results even when the maximum value to normalize is not known; 3) the integration schema: present systems follow three possible design choices, i.e., parallel, serial, or hierarchic; A new architectural schema called N-cross testing protocol (NCTP) have been proposed and present the results obtained by a multimodal system implemented according to it; 4) a reliability measure: each subsystem in a multimodal architecture should return a reliability measure, to express how much its response can be trusted; Two alternative measures have been proposed based on the composition of the stored gallery; the fusion process: the integration of information by different biometrics is possible in three moments, i.e., during feature extraction, matching, or decision . The sooner the fusion is performed, the higher amount of information can be saved.

The biometric information is decrypted and the user is subscribed to access the cloud. This decides whether the user is approved or denied. The accelerated comparison is done in milliseconds. Once the user is recognized as an authorized user, the user can view, upload or download the data stored in cloud. The active storage object holds the ttl value (time to live). Ttl value is used to trigger the self destruct operation.

3 SYSTEM ARCHITECTURE

Biometric information are gathered and by using the batch homomorphic encryption technique we encrypt the biometric information. If new user enter means they want to register their details and it will save on database. Afterwards biometric information are compared with the database. If it matches then decryption process are done. Then using the server IP address we have connect to the server and message send to the cloud server. Server replied you are a authorized person we can access the cloud otherwise not. Now in cloud their one login and registration process afterwards we can upload or download any type of file. While uploading or downloading one local copy will be generated. By

using the Shamir secret key we will delete that local copy for secure storage.

In cloud we can upload and download any type of files. While uploading or downloading a secret key is generated. Using the secret key only we done further modifications. Whenever enter into the cloud secret key is generated and send to their mail id. Each time secret key is changed.

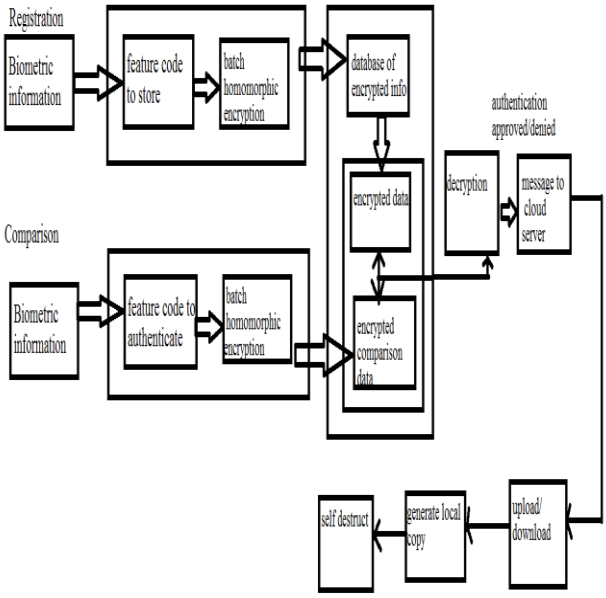

Fig .3.1 Architecture

4 ALGORITHM

Homomorphic authenticated encryption:

A HAE is a tuple Π = (Gen, Enc, Eval, Dec) of the following ,

• (ek, sk) ← Gen(1^{λ}): given a security parameter λ, Gen(1^{λ}) outputs a public evaluation key ek and a secret key sk.

• c ← Enc(sk, τ,m): given a secret key sk, a label τ →L and a plaintext m → M, Enc(sk, τ,m) outputs a ciphertext c →C.

• c ← Eval(ek, f, c_1, …, c_l): given an evaluation key ek, an arity-l admissible function f :M_l → M in F and l ciphertexts c_1, … , c_l →C, the deterministic algorithm Eval outputs
a ciphertext ˜c →C.

• m or M ← Dec(sk, (f, τ_1, …… , τ_l), c): given a secret key sk, a labeled program (f, τ_1,…. , τ_l)and a ciphertext c → C, the deterministic algorithm Dec outputs a message m →M.

Fingerprint matching algorithm using BLPOC Function

The user's biometric information is compared with the database where the authenticated user's encrypted information is stored. The user's fingerprints are encrypted by batch homomorphic encryption technique and then sent to the comparison server.

BLPOC Function

This section describes a finger-print matching algorithm using BLPOC function. The algorithm consists of the three steps:

(i) rotation and displacement alignment,
(ii) common region extraction and
(iii) matching score calculation with precise rotation.

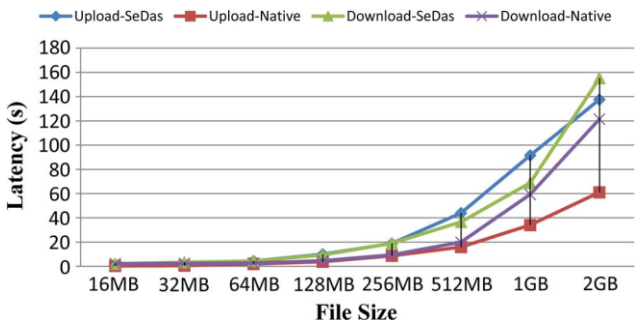

Fig 4.1: Latency in upload and download operations.

5 EXPERIMENTAL RESULTS

Decryption process are done and it show the matching score calculation. IP address is given to connect to the local server they decide authorized person or not. Then message send to cloud server to access the cloud.

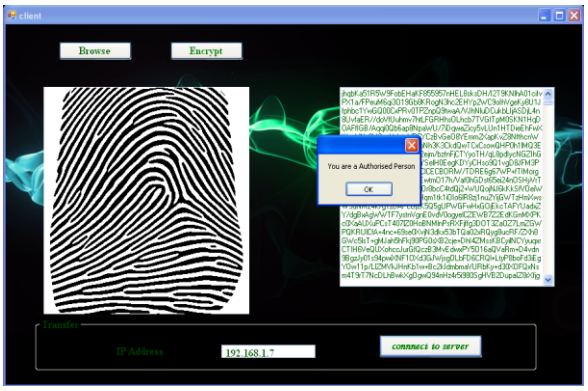

Fig: 5.1 IP address of the server

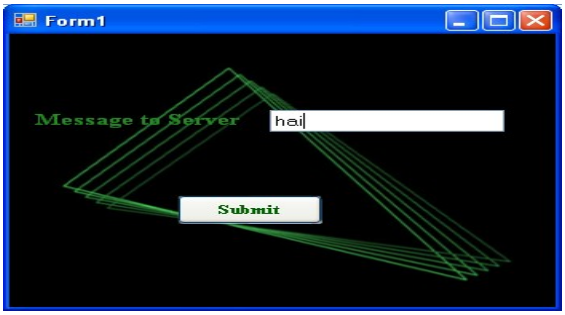

Fig 5.2 Message to cloud server

VI CONCLUSION AND FUTURE WORK

Secure data access in cloud using biometrical recognition is done by using batch homomorphic encryption technique. In this, privacy of data has become increasingly important in the cloud environment. A new approach for protecting the data from attackers who retroactively obtain, through legal or other means, a user's stored data and private decryption keys is implemented. It causes sensitive information, such as account numbers, passwords and notes to irreversibly self-destruct, without any action on the user's part.

The second phase of the system is focused on security. The reason for providing security is that the unknown users should not access the authorized data. The server provides access rights to access the data in the cloud.

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] 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. 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.

[4] 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.

[5] 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.

[6] Y. Kang, J. Yang, and E. L. Miller, "Object-based SCM: An efficient interface for storage class memories," in Proc. 27th IEEE Symp. Massive Storage Systems and Technologies (MSST), 2011.

[7] 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.

[8] 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.

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

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

[12] 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

[13] 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.
[14] J. R. Douceur, "The sybil attack," in Proc. IPTPS '01: Revised Papers From the First Int. Workshop on Peer-to-Peer Systems, 2002.
[15] 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.
[16] B. Poettering, 2006, SSSS: Shamir's Secret Sharing Scheme [Online].Available: http://point-at-infinity.org/ssss/
[17] M. Mesnier, G. Ganger, and E. Riedel, "Object-based storage," IEEE Commun. Mag., vol. 41, no. 8, pp. 84–90, Aug. 2003.

[1]Nivetha.P received degree B.E Computer Science and Engineering from Selvam College of technology, Anna university in 2012. Now pursuing M.E Computer Science and Engineering in Meenakshi College of Engineering, Anna university,

[2]Sarala.V received the M.Sc ., (computer science) from university of madras in 2004 and M.E degree in computer science engineering from anna university Chennai in 2007. Currently,she is an assistant professor in the computer science department, Meenakshi college of engineering Chennai, Anna university.

