International Journal of Computer Science and Engineering Communications,
Volume.5, Issue.6 (2017): Page.1784-1791
www.ijcsec.com

A Secure Shared Access in Cloud Computing Using Identity Group Key Based Encryption

S.Revathi[1] and V.Gowri[2]
[1]PG Scholar, [2]Assistant Professor,
[1, 2] Department of Computer Science and Engineering,
[1, 2] Ranippettai Engineering College, Walaja, India.
revathimerec@gmail.com[1], gowrimerec@gmail.com[2]

Abstract: *Cloud storage is an application of clouds that free from confinement of organizations from establishing in-house data storage systems. However, cloud storage gives apply to security reasons. In case of group-shared data, the data deals with both cloud-specific and conventional insider threats. Secure data sharing among a group that counters insider threats of conforming the rules yet malicious users is an important research issue. In this paper, we propose the Secure Data Sharing in Clouds methodology that provides: 1) Data being secret and the quality of being honest; 2) access control; 3) data sharing without using compute-intensive re-encryption; and 4) insider threat security. The Secure Data Sharing in Clouds methodology encrypts a file with a single encryption key. Two different key shares for each of the users are generated, with the user only getting one share. The possession of a single share of a key allows the Secure Data Sharing in Clouds methodology to counter the insider threats. The other key share is stored by a trusted third party, which is called the cryptographic server. The Secure Data Sharing in Clouds methodology is applicable to conventional and mobile cloud computing environments.*

Keywords: *Access Control, Cloud Computing, Cryptographic Server, encryption key, Group Shared Data, Re-encryption, Threat Security.*

Citation: S.Revathi, and V.Gowri.(2017). A Secure Shared Access in Cloud Computing Using Identity Group Key Based Encryption. *International Journal of Computer Science and Engineering Communications,* 5(6), 1784-1791. Article ID 5617841791.

I. Introduction

Cloud service providers offer users efficient and scalable data storage services with a much lower marginal cost than traditional approaches. It is routine for users to leverage cloud storage services to share data with others in a group, as data sharing becomes a standard feature in most cloud storage offerings, including Drop box, iCloud and Google Drive. The integrity of data in cloud storage, however, is subject to skepticism and scrutiny, as data stored in the cloud can easily be lost or corrupted due to the inevitable hardware/software failures and human errors. To make this matter even worse, cloud service providers may be reluctant to inform users about these data errors in order to maintain the reputation of their services and avoid losing profits. Therefore, the integrity of cloud data should be verified before any data utilization, such as search or computation over cloud data. Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the "software as a service" (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. Meanwhile, the increasing network bandwidth and reliable yet flexible network connections make it even possible that clients can now subscribe high quality services from data and software that reside solely on remote data centers.

One of the biggest concerns with cloud data storage is that of data integrity verification at untrusted servers. For example, the storage service provider, which experiences Byzantine failures occasionally, may decide to hide the data errors from the clients for the benefit of their own. What is more serious is that for saving money and storage space the service provider might neglect to keep or deliberately delete rarely accessed data files which belong to an ordinary client. Consider the large size of the outsourced electronic data and the client's constrained resource capability, the core of the problem can be generalized as how can the client find an efficient way to perform periodical integrity verifications without the local copy of data files.

In order to solve the problem of data integrity checking, many schemes are proposed under different systems and security models in all these works, great efforts are made to design solutions that meet various requirements: high scheme efficiency, stateless verification, unbounded use of queries and irretrievability of data, etc. Considering the role of the verifier in the model, all the schemes presented before fall into two categories: private auditability and public auditability. Although schemes with private auditability can achieve higher scheme efficiency, public auditability allows anyone, not just the client (data owner), to challenge the cloud server for correctness of data storage while keeping no private information. Then, clients are able to delegate the evaluation of the service performance to an independent third party auditor (TPA), without devotion of their computation resources. In the cloud, the clients themselves are unreliable or may not be able to afford the overhead of performing frequent integrity checks. Thus, for practical use, it seems more rational to equip the verification protocol with public audit ability, which is expected to play a more important role in achieving economies

of scale for Cloud Computing. Moreover, for efficiency consideration, the outsourced data themselves should not be required by the verifier for the verification purpose.

II. Literature of Survey

R.Curtmola, O.Khan: As storage-outsourcing services and resource-sharing networks have become popular, the problem of efficiently proving the integrity of data stored at untrusted servers has received increased attention. In the provable data possession (PDP) model the client preprocesses the data and then sends it to an untrusted server for storage, while keeping a small amount of meta-data. A definitional framework and efficient constructions for dynamic provable data possession (DPDP), which extends the PDP model to support provable updates to stored data. In cloud storage systems, the server (or peer) that stores the client's data is not necessarily trusted. The main contribution of this are provide the first efficient fully dynamic PDP solution next present a rank-based authenticated dictionary built over a skip list. This construction yields a DPDP scheme with logarithmic computation and communication and the same detection probability as the original PDP scheme. We give an alternative construction of a rank-based authenticated dictionary using an RSA tree. This construction results in a DPDP scheme with improved detection probability but higher server computation. To evaluate the performance of our DPDP scheme interms of communication and computational overhead, in order to determine the price of dynamism over static PDP. In version control to evaluate an application that suits our scheme's ability to efficiently handle and prove updates to versioned, hierarchical resources.[1]

M. A. Shah, R. Swaminathan: A growing number of online services such as google, yahoo! are starting to charge users for their storage. Customers often use these services to store valuable data such as email, family photos and videos, and disk backups. To make storage services accountable for data loss that allows a protocol for third party auditor to periodically verify the data stored by a service and assist in returning the data intact to the customer. The storage services must export hooks for challenges to the response queries and compute extensive functions for responses. To avoid these overheads we can batch many files together into single key and check that fill all at once. Hybrid as mentioned but is still imposes the end and overheads that the storage service and customer experience with our protocols. It will eliminate the encryption techniques and extend the formal definition of provable data possession and proof of retrievability. Auditor to be trusted and hat collude with either party.[2]

A. Oprea, M. K. Reiter: A new methods to provide block-level integrity in encrypted storage systems, i.e., so that a client will detect the modification of data blocks by an untrusted storage server. A trusted client component maintains state with which it can authenthicate blocks returned by the storage server, and we explore techinques for minimizing the size of this state. A scheme that implements integrity resistance to replay attacks. To authenticate data without changing the block size or the number of sectors accessed, clients need to keep themselves additional integrity information and it minimize the size of the integrity information, are provably secure, and storage-efficient.[3]

III. System Analysis

Cloud computing has been envisioned as the next-generation architecture of IT enterprise. We consider the task of allowing a third party auditor (TPA) to verify the integrity of dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of the client through the auditing of whether his data stored in the cloud is indeed intact. First of all, applying the traditional solution of key revocation to cloud storage auditing is not practical. This is because whenever the client secret key is exposed, the client needs to produce a new pair of public key and secret key and regenerate the authenticators for the client's data previously stored in cloud In our system, Merkle Hash Tree construction is used to authenticate the values of data blocks with the RSA Algorithm. Our main scheme to support batch auditing for TPA upon delegations from multiusers.

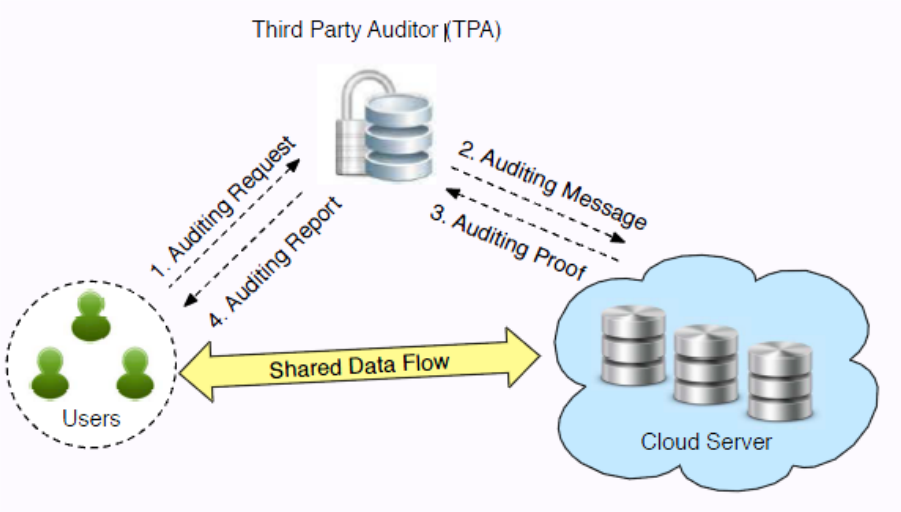

Figure 3.1 Architecture Diagram for Proposed System

IV. Software Description

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

The Java Virtual Machine

The Java Application Programming Interface

Microsoft Open Database Connectivity is a standard programming interface for application developers and database systems providers. Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC

application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMS. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or drivers.

V. Software Design

The data flow proceeds as that an owner will upload the specified file in the cloud server. That specified file store in DB. Then registered user make a request to the server, at that time data owner will check the user and then allow the file to download. Our project uses two tables through the MS-ACCESS. The tables are server details and user details. The server table is used to store the values to enable the third party auditor to verify the client's request. This table has fields like username, password, status, machine ip and port. The username table is used to stores the details about the user who are all the on-line users. It has many fields like username, password, status, port, start, increment, etc…

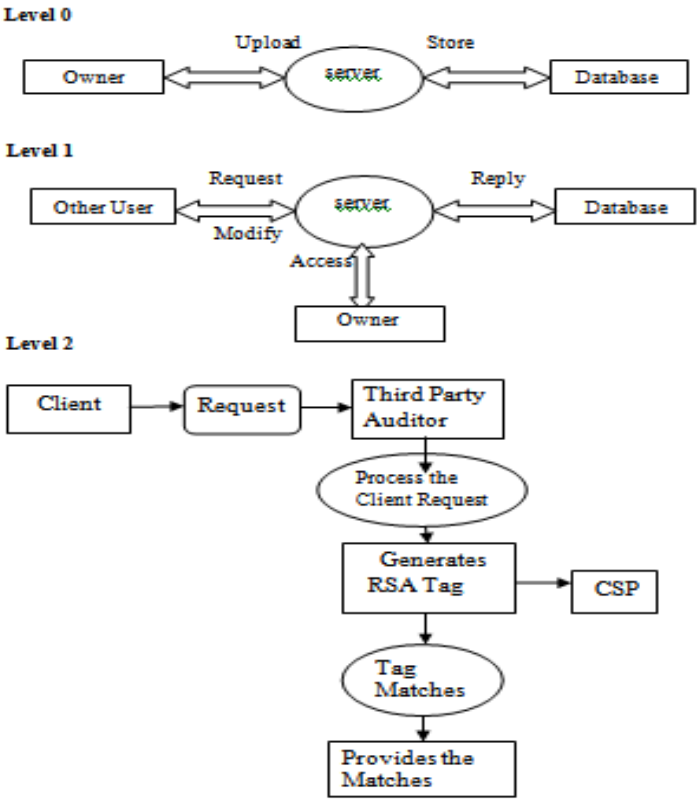

Figure 5.1 Dataflow Diagram for Provable Data Possession

Server				
username	**Password**	**Status**	**Machine ip**	**Port**
Server	Server	No	127.0.0.1	2001

Table 5.1 Server Details

Username	Password	status	Machine ip	port	start	Times	Public key	Private key	Common key
Bala	Bala	No	127.0.0.1	7006	61	2	11	25391	40301
Eve	Eve	Yes	127.0.0.1	7003	29	1	3	20491	31099
Alice	Alice	Yes	127.0.0.1	7000	19	1	5	22277	37523
Bob	Bob	Yes	127.0.0.1	7001	87	4	5	6605	33389
Durga	Durga	No	127.0.0.1	7010	55	3	7	2563	18209

Table 5.2 Username Details

Figure 5.2 Registration details

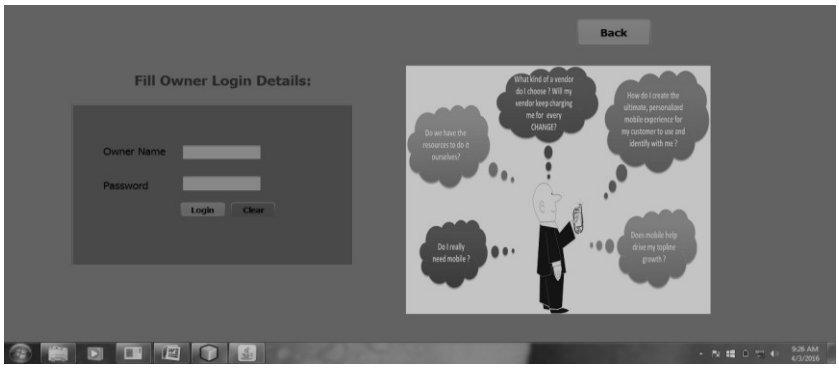

Figure 5.3 Owner login details

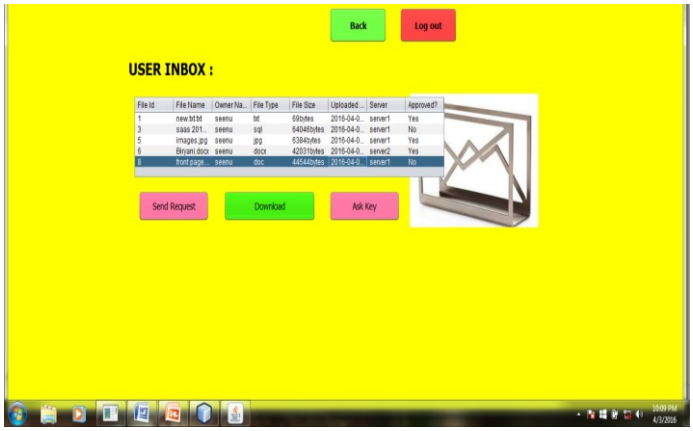

Figure 5.4 User inbox

IV. Conclusion

Ensure cloud data storage security; it is critical to enable a third party auditor to evaluated the service quality from an objective and independent perspective. Public auditability also allows client to delegate the integrity verification tasks to TPA while they themselves can be reliable or not be able to commit necessary computation resources performing continuous verifications. It supports batch auditing upon delegations from multi-users. Based on RSA Algorithm instantiation, computation cost of server and verifier will be reduced. Our construction is deliberately designed to meet an efficient security provided by TPA. Extensive security and performance analysis show that the proposed scheme is highly efficient and provably secure.

Acknowledgements

First of all I would like to express my deepest gratitude and sincere thanks to my guide, V. Gowri, Assistant Professor & Head of the Department, Ranippettai Engineering College,Walaja,Tamilnadu, India her valuable time and keep interest in my project work. Her intellectual advice has helping me in every step of my research work. I express my sincere thanks to our beloved Founder and Managing Director ‘’kalvi kavalar’’Thiru.B.Bose, Respected Principal Mr.N.C.Senthil Kumaran, for given valuable suggestions and Motivate this Efforts.

REFERENCES

[1] G. Ateniese, R. Burns, R. Curtmola "Provable data possession at untrusted stores,"in Proc. of CCS'07. New York, NY, USA: ACM, 2007.

[2] M. A. Shah, R. Swaminathan, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008.

[3] A. Oprea, M. K. Reiter, "Remote integrity check with dishonest storage server," in Proc. of ESORICS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 223–237.