ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

A Secure Client Side Replication Big File Cloud Storage Environments

S.Md Ismail, M.Tech
Associate Professor,
Department of CSE,
Al-Habeeb College of Engineering &Technology, Chevella, Hyderabad.

Mr.Mohd Anwar Ali, M.Tech
Associate Professor & HOD,
Department of CSE,
Al-Habeeb College of Engineering &Technology, Chevella, Hyderabad.

Eragam Niranjan Reddy
Post Graduate Student,
Department of CSE,
Al-Habeeb College of Engineering &Technology, Chevella, Hyderabad.

ABSTRACT:

Cloud-based storage services are swiftly emergent and becoming a budding in data storage meadow. Present be several problems when scheming an efficient storage engine for cloud-based systems with some necessities such as big-file processing, lightweight meta-data, low latency, parallel I/O, reduplication, disseminated, high scalability. Security and privacy are among top concerns for the public cloud environments. Towards these security challenges, we propose and implement, on Open Stack Swift, a new client-side reduplication scheme for securely storing and sharing outsourced data via the public cloud. The originality of our proposal is twofold. First, it ensures better confidentiality towards unauthorized users. That is, every client computes as per data key to encrypt the data that he intends to store in the cloud. As such, the data access is managed by the data owner. Second, by integrating access rights in metadata file, an authorized user can decipher an encrypted file only with his private key. This research applied the advantages of ZDB - an in-house key value store which was optimized with auto-increment integer keys for solving big-file storage problems efficiently. The results can be used for building scalable distributed data cloud storage that support big-file with size up to several terabytes.

Keywords:

Cloud Storage, Key-Value, Big File, Distributed Storage, Proof of Ownership.

1.INTRODUCTION :

The use of remote storage systems is gaining an expanding interest, namely the cloud storage based services, since it provides cost efficient architectures.

These architectures support the transmission, storage in a multi-tenant environment, and intensive computation of outsourced data in a pay per use business model. . For saving resources consumption in network bandwidth and storage capacities, many cloud services, namely Drop box and Memo pal, apply client side reduplication. This concept avoids the storage of redundant data in cloud servers and reduces network bandwidth consumption associated to transmitting the same contents several times.Despite these significant advantages in saving resources, client data reduplication brings many security issues, considerably due to the multi-owner data possession challenges. For instance, several attacks target either the bandwidth consumption or the confidentiality and the privacy of legitimate cloud users. For example, a user may check whether another user has already uploaded a file, by trying to outsource the same file to the cloud. Recently, to mitigate these concerns, many efforts have been proposed under different security models. These schemes are called Proof of Ownership systems (PoW). They allow the storage server check a user data ownership, based on a static and short value.Understanding the characteristics of data types especially the type of key in key-value pair is important to design the scalable store system for that data. There are several popular key types: variable-length string, fixed-size binary, random integers, auto-incremental integer... In popular applications, incremental integer keys are used widely in database design.

2.ASSOCIATED WORKS AND SECURITY ANALYSIS :

The Proof of Ownership (PoW) is introduced It is challenge-response protocol enabling a storage server to check whether a requesting entity is the data owner, based on a short value.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

That is, when a user wants to upload a data file (D) to the cloud, he first computes and sends a hash value hash = H(D) to the storage server. This latter maintains a database of hash values of all received files, and looks up hash. If there is a match found, then D is already outsourced to cloud servers. As such, the cloud tags the cloud user as an owner of data with no need to upload the file to remote storage servers. If there is no math, then the user has to send the file data (D) to the cloud. This client side reduplication, referred to as hash-as-a proof presents several security challenges, mainly due to the trust of cloud user's assumption.

2.1 Security Analysis:

Despite the significant resource saving advantages, PoW schemes bring several security challenges that may lead to sensitive data.

• **Data confidentiality disclosure –** hash-as-a-proof schemes introduce an important data confidentiality concern, mainly due to the static proof client side generation. For instance, if a malicious user has the short hash value of an outsourced data file, he could fool the storage server as an owner trying to upload the requested data file. Then, he gains access to data, by presenting the hash proof. As such, an efficient PoW scheme requires the use of unpredictable values of verifications.

• **Privacy violation –** sensitive data leakage is a critical challenge that has not been addressed by Halevi. That is, cloud users should have an efficient way to ensure that remote servers are unable to access outsourced data or to build user profiles.

• **Poison attack –** when a data file D is encrypted on the client side, relying on a randomly chosen encryption key, the cloud server is unable to verify consistency between the uploaded file and the proof value hash. In fact, given the storage server can not verify, if there is an original data file D that provides a hash value hash. As such, a malicious user can replace a valid enciphered file with a poisoned file. So, a subsequent user loses his original copy of file, while retrieving the poisoned version.

Proposed the concept of Proof of Ownership (PoW), while introducing three different constructions, in terms of security and performances.

These schemes involve the server challenging the client to present valid sibling paths for a subset of a Merkle tree leaves. The first scheme applies erasure coding on the content of the original file. This encoded version is the input for construction of the Merkle tree. The second purpose pre-possesses the data file with a universal hash function instead of erasure coding. The third construction is the most practical approach. Design an efficient hash family, under several security assumptions. Unfortunately, the proof assumes that the data file is sampled from a particular type of distribution. In addition, this construction is given in random oracle model, where SHA256 is considered as a random function.They use the projection of the file into selected bit-position as a proof of ownership. The main disadvantage of this construction is the privacy violation against honest but curious storage server. The confidentiality preservation concern in cross-user client side reduplication of encrypted data files. They used the convergent encryption approach, for providing reduplication under a weak leakage model. Unfortunately, their paper does not support a malicious storage server adversary.

2.2 Threat Model:

For designing a secure client-side reduplication scheme, we consider two adversaries: malicious cloud user and honest but curious cloud server.

• **Malicious user adversary –** the objective of a malicious user is to convince the cloud server that he is a legitimate data owner. That is, we suppose that the adversary successes to gain knowledge of an arbitrary part of D. This information is then used as a challenging input to the POW protocol.

• **Curious cloud server adversary –** this storage server honestly performs the operations defined by our proposed scheme, but it may actively attempt to gain the knowledge of the outsourced sensitive data. In addition, he may try to build links between user profiles and accessed data files.

3.SYSTEM MODEL

It relies on the following entities for the good management of client data:

• **Cloud Service Provider (CSP):** a CSP has significant resources to govern distributed cloud storage servers and to manage its database servers

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

It also provides virtual infrastructure to host application services. These services can be used by the client to manage his data stored in the cloud servers.

• **Client:** a client makes use of provider's resources to store, retrieve and share data with multiple users. A client can be either an individual or an enterprise.

• **Users:** the users are able to access the content stored in the cloud, depending on their access rights which are authorizations granted by the client, like the rights to read, write or re-store the modified data in the cloud. These access rights serve to specify several groups of users. Each group is characterized by an identifier IDG and a set of access rights. In practice, the CSP provides a web interface for the client to store data into a set of cloud servers, which are running in a cooperated and distributed manner. In addition, the web interface is used by the users to retrieve, modify and restore data from the cloud, depending on their access rights.Moreover, the CSP relies on database servers to map client identities to their stored data identifiers and group identifiers.

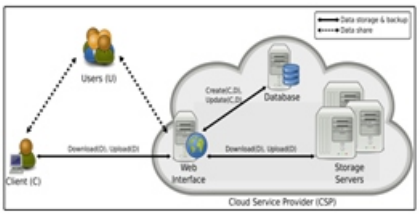

Fig1: Architecture of cloud data storage

Clients do not directly request to Cloud Apps Service, but through a dispatcher which provides public APIs for clients. The dispatcher checks user session before forwarding the client request to Cloud Apps Service. Moreover, the dispatcher also checks the number of connections from a client, if there are too many concurrent connections from a client, the dispatcher can block that client's requests. Storage Logical Layer stores and retrieves data from Object Store Layer.

Object Store Layer is the most important layer which has responsibility for storing and caching objects. This layer manages information of all objects in the system including user data, file information data, and especially meta-data. In BFC system, meta-data describes a file and how it is organized as a list of small chunks. We implemented some optimizations to make low-complicated Meta-data.

Fig2: BFC Main Backend Components

Object Store Layer contains many distributed backend services. Two important services of Object Store Layer are File Info Service and Chunk Store Service. Mall chunks can be stored efficiently in a key-value store. It is difficult to do this with a large file directly in local file system. In addition, this supports uploading and downloading file parallel and reusable.

3.1.Existing System:

People use cloud storage for the daily demands, for example backing-up data, sharing file to their friends via social networks such as Face book, Zing Me. Users also probably upload data from many different types of devices such as computer, mobile phone or tablet. After that, they can download or share them to others. System load in cloud storage is usually really heavy.Thus, to guarantee a good quality of service for users, the system has to face many difficult problems and requirements.

Disadvantages:

•Storing, retrieving and managing big-files in the system efficiently.
•Parallel and reusable uploading and downloading.
•Data reduplication to reduce the waste of storage space caused by storing the same static data from different users.

3.2 PROPOSED SYSTEM:

A common method for solving these problems which is used in many Distributed File Systems and Cloud Storages is splitting big file to multiple smaller chunks, storing them on disks or distributed nodes and then managing them using a meta-data system. Storing chunks and meta-data efficiently and designing a lightweight meta-data are significant problems that cloud storage providers have to face.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

After a long time of investigating, we realized that current cloud storage services have a complex meta-data system; at least the size of metadata is linear to the file size for every file. Therefore, the space complexity of these meta-data system is O (n) and it is not well scalable for big-file. In this research, we propose new big-file cloud storage architecture and a better solution to reduce the space complexity of meta-data.

Advantages:

•Propose a light-weight meta-data design for big file. Very file has nearly the same size of meta-data.
•Propose a logical contiguous chunk-id of chunk collection of files. That makes it easier to distribute data and scale-out the storage system.
•Bring the advantages of key-value store into big-file data store which is not default supported for big-value. ZDB is used for supporting sequential write, small memory-index overhead.

3.3 MODULES:

1.Application Layer
2.Storage Logical Layer
3.Object Store Layer
4.Persistent Layer

Module description:

Application Layer: It consists of native software on desktop computers, mobile devices and web-interface, which allow user to upload, download and share their own files.

Storage Logical Layer:

It consisted of many queuing services and worker services, ID-Generator services and all logical API for Cloud Storage System. This layer implements business logic part in BFC.

Object Store Layer:

It contains many distributed backend services. Two important services of object Store Layer are File Info Service and Chunk Store Service. File Info Service stores information of files. Chunk Store Service stores data chunks which are created by splitting from the original files that user uploaded.

Persistent Layer: it based on ZDB key-value store. There are many ZDB instances which are deployed as a distributed service and can be scaled when data growing.

3.4.Motivation:

The proposed system supports adaptive encryption methods for public cloud database propose a light-weight meta-data design for big file. Every file has nearly the same size of meta-data. BFC has space complexity of meta-data of a file, while size of meta-data of a file in Drop box, HDFS has space complexity of where n is size of original file. Propose a logical contiguous chunk-id of chunk collection of files. That makes it easier to distribute data and scale-out the storage system. Bring the advantages of key-value store into big-file data store which is not default supported for big-value.

3.5.Cloud Data Backup:

The data backup process starts when the client requests for retrieving the data previously stored in the cloud. The data backup process includes the following messages:

• **Client Request Backup:** it contains the URI of the requested data that the client wants to retrieve. Upon receiving this client request, the CSP verifies the client ownership of the claimed file and generates a Response Backup message.

• **Response Backup:** in his response, the CSP includes the encrypted outsourced data kf (f). Upon receiving the Response Backup message, the client. First retrieve the file metadata and deciphers the data decrypting key kf, using his secret key. Then, he uses the derived key to decrypt the request data file.

4. A SWIFT-CLIENT DATA DEDUPLICATION BASED SYSTEM

4.1 Context

In order to evaluate the performances of our proposal, we build a simulated cloud storage framework, based on Open Stack Storage system. Swift is a cloud based storage system, which stores data and allows write, read, and delete operations on them. To achieve security enhancement of Swift, we extend its functionalities with algorithms and protocols designed in our scheme.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

We have designed our own architecture, performing an installation of swift. Indeed, our architecture consists in dividing the machine drive into four physical volumes. Then, each volume is divided into four logical volumes. In total, we obtain sixteen partitions; each one represents one logical storage zone.The simulation consists of two components: the client side and the cloud side. We implement several cryptographic algorithms based on cryptographic functions from the Open SSL library the GMP library and the Pairing Based Cryptography (PBC) library, with independent native processes such, we conducted some tests by choosing different files of different sizes. At each time, we computed the average time for uploading and downloading the encrypted file. Next, we present the results of average time computation to upload and download data file in the cloud.

	Average Time in "s"		Standard Deviation σ	
Size in Bytes	Upload	Download	Upload	Download
10	0.338	0.193	0.231	0.067
10^2	0.329	0.192	0.210	0.060
10^3	0.339	0.189	0.233	0.027
10^4	0.326	0.194	0.191	0.080

TABLE I: Average time to upload and download file of size from 10 to 104 bytes, encrypted by AES-256-CBC

By analyzing the results, we can conclude that:

• The time to upload a given data in the cloud is greater than the time to download it from remote servers.
• For data size less than 5 × 104 bits, the time needed to upload the file in the cloud (respectively download it

	Average Time in "s"		Standard Deviation σ	
Size in Bytes	Upload	Download	Upload	Download
1×10^3	0.340	0.190	0.230	0.021
2×10^3	0.328	0.190	0.202	0.035
3×10^3	0.340	0.189	0.260	0.018
4×10^3	0.333	0.191	0.235	0.059
5×10^3	0.333	0.193	0.260	0.127
6×10^3	0.324	0.188	0.200	0.020
7×10^3	0.337	0.188	0.241	0.026
8×10^3	0.325	0.191	0.176	0.026
9×10^3	0.328	0.192	0.194	0.025

TABLE II: Average time to upload and download data file 6 of size from 103 to 9 × 103 bytes, encrypted by AES-256-cbc

4.2 Swift-Client Access Control Integration:

In order to include the security procedures at the client side, we first choose an asymmetric algorithm based on the use of Elliptic Curve Cryptography (ECC).

In order to use the Elliptic Curve Integrated Encryption Scheme (ECIES) asymmetric encryption scheme [9], we append encryption functions to upload swift command, in order to support encrypted data key in user metadata. As such, the cloud client has to apply the following command: swift -A http://ip:port/auth/v1.0 -U account:login -K password upload container object -I ECIES Encrypt.

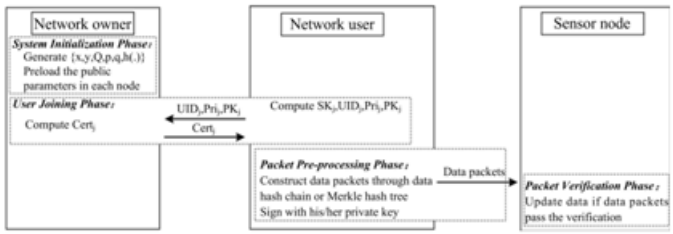

Fig3: Information processing flow in DiDrip.

Before disseminating the n data items, user U j signs the root node with his/her private key SKj and then transmits the advertisement packet P0 comprising user certificate Cert j Subsequently, user Uj disseminates each data item along with the appropriate internal nodes for verification purpose. Note that as described above, user certificate Certj contains user identity information UID j and dissemination privilege Prij. Before the network deployment, the network owner assigns a predefined key to identify this advertisement packet.However, a limitation of the data hash tree method is that it just works well in networks with in-sequence packet delivery. Such a limitation does not exist in the Merkle hash tree method since it allows each packet to be immediately authenticated upon its arrival at a node. Therefore, the choice of each method depends on this characteristic of the WSNs.The network has 24 TelosB nodes arranged in a 4 6 grid. The distance between each node is about 35 cm,

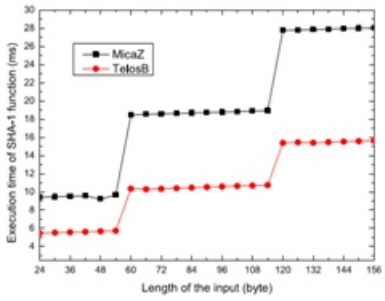

Fig4: The execution times of SHA-1 hash function on MicaZ and TelosB motes

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

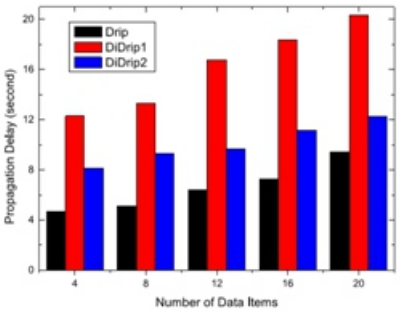

Fig5: Propagation delay comparison of three protocols when the data hash chain method is employed.

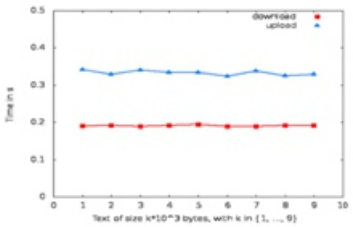

Fig6: Average time to upload and download file of different sizes

At the client side, our scheme brings acceptable computation costs. The overhead of the implemented security mechanisms does not affect the client resources. At the cloud side, our reduplication proposal preserves the efficiency of content upload/download operations with a smaller overhead of cryptographic operations.

5. LITERATURE SURVEY

5.1 Introduction:

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system. Literature survey is the documentation of a comprehensive review of the published and unpublished work from secondary sources data in the areas of specific interest to the researcher. The library is a rich storage base for secondary data and researchers used to spend several weeks and sometimes months going through books, journals, newspapers, magazines, conference proceedings, doctoral dissertations, master's theses, government publications and financial reports to find information on their research topic. With computerized databases now readily available and accessible the literature search is much speedier and easier and can be done without entering the portals of a library building.

5.2 Result Analysis :

This paper presents a small sample study that examines whether economic, structural, and cultural characteristics of a community explain the incidence of Craigslist-based scams. We present an empirical investigation with automobiles scams, an example of advance fee fraud.How to avoid the scam depend upon the craigslist, deal locally and buy and sell locally with security system. In this paper how to avoid scam means, deal locally and find the product location and then precede the transaction, with cash payment Expected value: is nothing but expected behavior of application. Actual value: is nothing but actual behavior of Application Bug Tracing: Collect all the failed cases, prepare documents. Reporting:Prepare document (status of the application)

5.3 Acknowledgments:

This research is supported by a strategic research grant from City University of Hong Kong Project No. 7004054], the Fundamental Research Funds for the Central Universities, and the Specialized Research Fund for the Doctoral Program of Higher Education. D. He is the corresponding author of this article.

6.Conclusion:

The growing need for secure cloud storage services and the attractive properties of the convergent cryptography lead us to combine them, thus, defining an innovative solution to the data outsourcing security and efficiency issues. Our solution is based on a cryptographic usage of symmetric encryption used for enciphering the data file and asymmetric encryption for meta data files, due to the highest sensibility of these information towards several intrusions. In addition, thanks to the Merkle tree properties, this proposal is shown to support data deduplication, as it employs an pre-verfication of data existence, in cloud servers, which is useful for saving bandwidth.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Besides, our solution is also shown to be resistant to unauthorized access to data and to any data disclosure during sharing process, providing two levels of access control verification. Finally, we believe that cloud data storage security is still full of challenges and of paramount importance, and many research problems remain to be identified.

7.Reference:

•https://github.com/openstack/swift.

•L. Ben. On the implementation of pairing-based cryptosystems, 2007.

•R. Di Pietro and A. Sorniotti. Boosting efficiency and security in proof of ownership for deduplication. In Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ASIACCS '12, pages 81–82, New York, NY, USA, 2012. ACM.

•J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In In Proceedings of 22nd International Conference on Distributed Computing Systems (ICDCS, 2002.

•M. Dutch. Understanding data deduplication ratios. SNIA White Paper, June 2008.

•T. G. et al. GNU multiple precision arithmetic library 4.1.2, December 2002.

•O. Goldreich. Foundations of Cryptography: Basic Tools. Cambridge University Press, New York, NY, USA, 2000.

•S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg. Proofs of ownership in remote storage systems. In Proceedings of the 18th ACM conference on Computer and communications security, CCS '11, pages 491–500, New York, NY, USA, 2011. ACM.

•D. Hankerson, A. J. Menezes, and S. Vanstone. Guide to Elliptic Curve Cryptography. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

•D. Harnik, B. Pinkas, and A. Shulman-Peleg. Side channels in cloud services: Deduplication in cloud storage. IEEE Security And Privacy, 8(6):40–47, 2010.

•R. C. Merkle. A digital signature based on a conventional encryption function. In A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, CRYPTO '87, pages 369–378, London, UK, UK, 1988. Springer-Verlag.