International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Performance Evaluation of Fine-Grained Deduplication using Hybrid Cloud

Shital S. Dadge* **and Ashish Kumar**

G.H.Raisoni College of Engineering and Management, Ahmednagar, India

Accepted 10 July 2016, Available online 15 July 2016, **Vol.6, No.4 (Aug 2016)**

Abstract

Cloud computing provides unlimited virtualized resources to the users as services at very low price. It has attracted large number of users from all fields. As the number of users will increase more data will be uploaded on cloud. So to store and upload large volume of data is big problem today. The efficient data reduction technology will be required to handle this problem. Deduplication is such a technology which does not allows us to store and transfer duplicate data in storing devices. So it plays an important role for bandwidth-sensitive data uploading and cloud applications. This system combines both file level and block level deduplication. For both levels of deduplication it uses a variant of convergent encryption along with privileges, which resists brute-force attacks. Here keys are server-generated, which provides security to cross-client deduplication. This system also uses various levels of keys to efficiently manage them, and to reduce the computation time. This system provides security to both data and key by using hybrid cloud architecture with several new deduplication constructions. It support authorized duplicate check, in which the duplicate-check tokens for files and blocks are generated by the private cloud server with file and block level keys. This system provides better protection to data security, and attempt to address the problem of authorized fine-grained deduplication.

Keywords: *Cloud computing; hybrid cloud; convergent encryption; authorised deduplication*

Introduction

Cloud computing is helping us to use new business models and various cost effective resource usage. In these days no business companies want to maintain large data centers. They just gives importance to their own business and they can purchase the resources as needed. So the problem is to provide protection for the sensitive data along with deduplication support. The customers and companies generally, backup their data to cloud storage which performs deduplication to save storage and uploading bandwidth. So performing deduplication with security becomes a challenge for storage providers.

The existing deduplication methods are not suitable for fine grained data deduplication, because it may be vulnerable to brute-force attacks which recovers files from a previously known set, or produces large time computation overheads. At present the current approaches of convergent key management produces large key space overheads because of the large number of blocks shared among users. Deduplication can be performed at both the block and file level. In File-level deduplication, redundant copies of the same file are skipped from uploading. And in block-level, redundant blocks of data from non-identical files are omitted from uploading. There are differences in deduplicaton granularity. In file level deduplication, the deduplication ratio is not obvious when compared with block-level deduplication or sub-file level. Therefore, block chunking algorithms are often used to realize block level deduplicaton, including fixed-sized and variable-sized algorithms. Many applications (e.g., office, virtual machine environments, etc.) works well with fixed-sized deduplication.

Although the deduplication provides a lot of benefits, but it also has some security and privacy issues. Because user's data is sensitive and may suffer from various attacks. Just encryption, provides data confidentiality, but not the deduplication. As in the simple encryption, individual users will encrypt their data with their own private keys. So, same data copies of various users will generate dissimilar ciphertexts, which does not allow deduplication. Hence, a variant of convergent encryption is used to allow deduplication while providing security and confidentiality to user's data.

Deduplication scheme for cloud storage aims to improve storage efficiency and maintaining redundancy for fault tolerance with security becomes a challenge for storage providers.

*Corresponding author: **Shital S. Dadge**

Motivation

Many people are using cloud as a storage service. The increasing number of users requires enormous space. So for cloud to manage such a huge amount of data is main problem. For this purpose many techniques have been introduced, deduplication is one of them. Although, deduplication has many advantages but it has some security and design issues. So the main problem is to protect the confidentiality of sensitive data while supporting deduplication.

The file level deduplication method has large storage overhead and less efficiency than block level. To overcome this problem, this system will divide the unique files into blocks and then it will again check each block against the storage to avoid the duplicate copies of data. It will reduce space used for storage as well as data overhead. The system will encrypts the data before sending to storage provider and attempts to address the problem of confidentiality of sensitive data while providing deduplication. It tries to solve the problem of handling huge volume of data by performing block level deduplication

Related work

There are various implementations of convergent encryption variants for secure deduplication (A. El-Shimi *et al*, 2012; S. Quinlan *et al*, 2011; *S*. Bugiel *et al*, 2002; Y. Tan *et al*, 2010). But all the above schemes do not consider data privacy and how to minimize the key management overhead. To protect the data confidentiality DupLESS (M. Bellare *et al*, 2013) transforms the predictable message into unpredictable message, it introduces another third party key server which generates the file tag for checking duplicate. The key server may lead to single point of failure. And it is not suitable for block level deduplication as it induces large time and space overhead.

ClouDedup preserves the combined advantages of deduplication and convergent encryption (P. Puzio *et al,* 2013). But the single key server leads to single point of failure risk. Dekay addresses the problem of providing reliable and efficient key management in secure deduplication. It distributes convergent key shares among various key servers, to preserve the security of key. It supports deduplication at both file and block level (Jin Li *et al,* 2014). But it has large key space overhead and the master key may leads to single point of failure risk.

Proposed System Overview

System Architecture

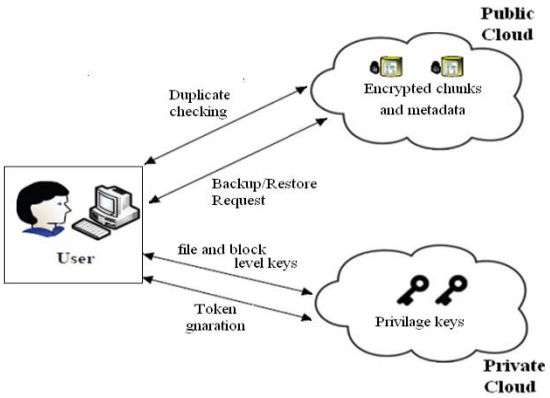
This system has three entities users, Admin, and two server process. When a user wants to access the private cloud server and the public cloud, his/her password and privilege should be verified first. The architecture of this system is shown in following fig1.This system will not back up any repeated data to save storing space and upload bandwidth.

Admin is the person who has the authority to register users (for example employees of a company) of the system. First admin has to log in into the system, then can be able to add users and give credentials to the users for further accessing the system.

After getting credentials from admin user can log in to the system. The successful log in leaves the user into the user screen module. Here the user can either upload or download files from system. For uploading a file user has to give access permission for that file, so that this file can be shared by those users who has this access permission.

This system uses two server application programs as its server processes. One works as a private server program which performs all the data related operations such as token generation, key generation, performing deduplicable encryption. Another server process works as public server who performs the duplicate checking and stores the files on the public cloud. The public cloud first divides the files into chunks and then performs duplicate checking for each chunk. Then it uploads only the unique chunks which are not already present on cloud. And for duplicate chunks it stores the pointer which references the original chunk.

When user send request for uploading a file the private server generates token for that file. Then it sends this token to the public cloud for duplicate checking, if this token is already present means that the data copy either file or block is duplicate and is already present on cloud. So public cloud does not upload this data copy. As the system does not store the duplicate data copies, hence it saves the storage space.



System Architecture for authorised deduplication

When user uploads a unique file means, its token is not present on storage cloud. Now system performs block level deduplication to find out that is there any redundant data present inside the file. Hence the whole file is divided into small chunks or blocks which are of fixed size, for example each chunk will be of size 2KB.

The private cloud server generates tokens for each chunk by using SHA-1 hash function and sends them to public cloud for duplicate checking. If public cloud founds that some duplicate blocks are present inside

the file, then it does not uploads the duplicate chunks, only unique chunks are stored on cloud. The block level deduplication saves more storage space than file level.

Mathematical model of system

- Consider S as a system that finds out duplicate copies of the files and blocks.

S = {U, F, B, C, T, K, M, O}

Where,
$U = \{U_1, U_2, U_3, \ldots U_n\}$ are users
$F = \{F_1, F_2, F_3, \ldots F_n\}$ are files
$F_i = \{B_1, B_2, B_3, \ldots B_n\}$ are blocks
$B_i = \{C_{Bi}, T_{Bi}, K_{Bi}\}$,
$T = \{T_F, T_{Bi}\}$ are tokens
$K = \{K_F, K_{Bi}, K_P\}$ are keys
M = Metadata of file,
O = Output consist reduced database size.

Algorithms

1. **System Algorithm**

Input: Data file to upload
Output: file uploaded to cloud

Steps:

1. Take F_i = input file.
2. Generate token T(F)←GenToken (F)
3. Compare token
4. **If** token found **then** inform user, skip uploading
5. **Else** apply Chunking to the data file
6. **For** each divided chunk $B_1, B_2, B_3 \ldots B_N$.
7. Generate token T(Bi)←GenToken (Bi)
8. Check for duplicate
9. Upload unique encrypted chunks
10. **End for**

2. **Encryp(K,M)** → C is the symmetric encryption algorithm that takes key K and the data copy M as inputs and then produces a ciphertext C. In this system AES algorithm is used which is a symmetric key algorithm. It uses fixed block size but variable key size for encrypting the files.
3. **Decryp(K,C)** → M is the decryption algorithm which takes the ciphertext C and the key K as inputs and then gives the original data copy M.

Implementation Details

The proposed system includes three different entities, which are implemented as three separate java programs. First the user screen program it stores the users information and allows users to perform the uploading and downloading procedure. The users are registered by the admin authority. The second program is private server program which acts as a private cloud and it generate the private keys for users according to the privileges mentioned by him and also computes the file and block token using SHA-1 hash function, which are used for duplicate checking. It also performs the functions file splitting algorithm, key and tag generation algorithms and symmetric decryption algorithm.

Third is public server program it works as a public cloud and stores the files and blocks in encrypted form and it also performs the deduplication against hash table of previously stored files and blocks. For storing the data copies the real time cloud amazon S3 is used.
The fig 2 shows the user registration process which is performed by admin of system. After entering the user details in correct format the registration process gets completed. Here the admin gives the credentials like user id and password to the user. After this by using id and password now user can login to the system and goes to user screen module.

User registration screen

Figure 3 shows user screen module. After verifying the users identity and privileges user goes to this screen user can perform uploading or downloading. For that user has to first choose the file to upload then must give access permission for that file. This access permission is for sharing this file among users. Then at time of uploading first the file level deduplication is performed.

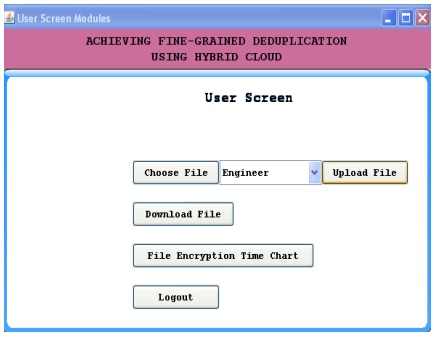


Fig.3 User screen

If duplicate file exists then user is informed about it and uploading is skipped. Else further the block level deduplication is performed by splitting file into number of fixed sized blocks. And only unique blocks are uploaded on cloud. After complete uploading user is informed that file successfully uploaded on cloud. At the time of downloading it is checked that the intended user must have the access permission for that file, otherwise the user is not allowed to download it.

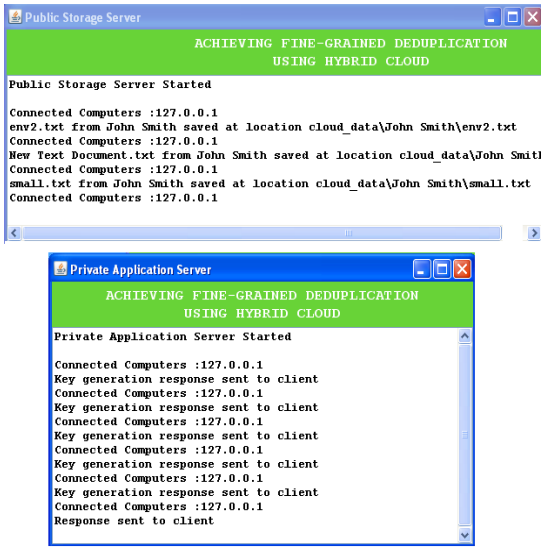


Fig. 4 Public and private server applications

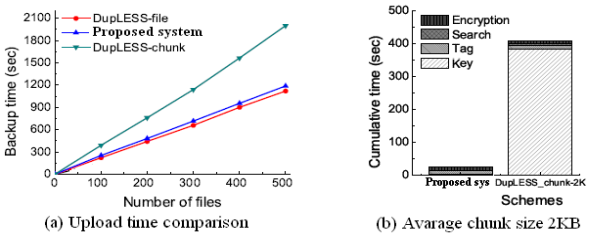
Fig 4 shows the screenshots of both public and private clouds. First it indicates the IP address of user. When user uploads a file the request is sent to private application server, which generates the key and token of that file and sends it to user. Then user sends this token to public application server for duplicate checking. Public server shows that which files are successfully uploaded on cloud.

Performance Evaluation

To compare the performance of this system with existing ones following factors are considered as the performance metrics, Number of files, storage time (which includes time for key generation, tag generations, search, encrypting chunks and key encryption time). Deduplication factor is taken as ratio of the data sizes before or after the deduplication.

This system can easily avoid the external attacks by making use of authentication. To resist internal or brute force attacks the system encrypts all chunks with chunk level keys, and chunk level keys are encrypted with file level keys. Each file level key is encrypted along with privilege key by using convergent encryption. So system ensures security of both the data and keys.

Figure 5 shows time comparison of proposed system with existing dupLESS-chunk. Fig (a) shows the upload time comparison with DupLESS file and chunk approaches. System takes little more time than DupLESS file as it combines both file and chunk level deduplication, but it is more superior than DupLESS chunk. In Fig (b) X axis shows schemes and Y axis shows cumulative time taken for each step such as encryption, tag and key generation. Proposed system takes very less time than dupLESS-chunk, because it avoids the complex RSA-OPRF protocol. DupLESS uses this protocol to generate keys for each block; it is server generated and follows some blind signatures. In proposed system these steps are avoided so it saves time.



(a) Upload time comparison (b) Avarage chunk size 2KB

Fig. 5 Comparison with existing system

Conclusion

System provides authorized fine grained data deduplication for protecting the security of data. It provides different privileges to users for checking duplicate data copies .This system is developed to avoid duplicate chunks uploaded on public cloud by using authorized deduplication. It will render the system more efficiently than file level deduplication. This system uses file level keys to encrypt the chunk level keys so it will reduce the extra key space overhead required for chunks or blocks.

Also solve the problems of key security and data confidentiality for fine-grained deduplication in cloud backup systems. The state-of-the-art secure deduplication methods may be affected due to brute-force attack, or can induce large computation timing, and key management also induces large key space overheads. But this system exploits duplicate data distribution on both file-level and block-level to introduce different security features, so as to make a trade-off between the data security and deduplication performance.

References

J. Li, Y. K. Li, X. Chen, P. P.C. Lee, and W. Lou (2015), A hybrid cloud approach for secure authorized deduplication. Parallel and Distributed Systems, IEEE Transactions on, 26(5), 1206-1216.

P. Puzio, R. Molva, M. Onen, and S. Loureiro (2013,) Cloudedup: secure deduplication with encrypted data for cloud storage, in Proceedings of the 5th International Conference on Cloud Computing Technology and Science. Bristol, pp. 363–370.

M. Bellare, S. Keelveedhi, and T. Ristenpart (2013), Dupless: Server aided encryption for deduplicated storage, in Proc. 22nd USENIX Conf. Sec. Symp, pp. 179–194.

M. Bellare, S. Keelveedhi, and T. Ristenpart (2013), Message-locked encryption and secure deduplication, in Proc. 32nd

Annu. Int. Conf. Theory Appl. Cryptographic Tech, pp. 296–312.

D. Meyer and W. Bolosky (2011), A study of practical deduplication, Proceedings of the USENIX Conference on File and Storage Technologies. San Jose, CA, USA: USENIX Association , pp. 229-241.

A. El-Shimi, R. Kalach, A. Kumar (2012), Primary data deduplication- large scale study and system design, in Proceedings of the 2012 conference on USENIX Annual Technical Conference. Boston, MA, USA: USENIX Association, pp. 1-12.

S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider(2011), Twin clouds: An architecture for secure cloud computing, in Proc. Workshop Cryptography Security Clouds, pp. 32–44.

S. Quinlan and S. Dorward (2002), Venti: A new approach to archival storage, In Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST02). erkeley, CA, USA: USENIX Association, , pp. 89-101..

S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg (2011), Proofs of ownership in remote storage systems, in Proc. ACM Conf. Comput. Commun. Security, pp. 491-500.

J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou (2014), Secure deduplication with efficient and reliable convergent key management, in Proc. IEEE Trans. Parallel Distrib. Syst., 25(6), pp. 1615–1625.

C. Ng and P. Lee(2013), Revdedup: A reverse deduplication storage system optimized for reads to latest backups, in Proc. 4th Asia- Pacific Workshop Syst. http://doi.acm.org/10.1145/2500727-2500731,

W. K. Ng, Y. Wen, and H. Zhu (2012), Private data deduplication protocols in cloud storage, in Proc. 27th Annu. ACM Symp. Appl. Comput., pp. 441-446.

R. D. Pietro and A. Sorniotti (2012), Boosting efficiency and security in proof of ownership for deduplication, in Proc. ACM Symp. Inf., Comput. Commun. Security, pp. 81-82.

A. Rahumed, H.C.H.Chen, Y.Tang, P.P.C. Lee, and J. C. S. Lui (2011), A secure cloud backup system with assured deletion and version control, in Proc. 3rd Int. Workshop Secutiry Cloud Comput., pp. 160-167.

J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl(2013), A secure data deduplication scheme for cloud storage, Tech. Rep. IBM Research, Zurich, ZUR 1308-022. Y. Tan, H. Jiang, D. Feng (2010), SAM: A Semantic- Aware Multi Tiered Source De-duplication Framework for Cloud Backup. In Proceedings of the 39th International Conference on Parallel Processing (ICPP10). San Diego, CA, USA: IEEE Computer Society Press, pp. 614 - 623.