International Journal of Innovative Research and Advanced Studies (IJIRAS)
Volume 4 Issue 6, June 2017

ISSN: 2394-4404

A Authorized And Secured Structure For Replicated Data Through Hybrid Cloud

Mr.Waghamare Amol Arjun

Prof. D. C. Mhetre

Department of Computer Engineering (KJCOEMR)
KJ College of Engineering Management and Research,
Pune University, India

Abstract: ***Efficient storage management is one of the challenges for Cloud computing. To achieve greater efficiency the redundant data must be removed or not to be allowed to store in cloud. Data deduplication is one method that can achieve this redundancy free data storage on cloud. This technique eliminates duplicate copies of repeating data which saves the space and bandwidth. Further to protect functionality the convergent encryption technique applied before deduplication. After confidentiality protection, security is major concern. To tackle security issues, authorized deduplication is proposed. This is different than the traditional approach because privilege of user are checked for duplicate check besides the data itself. To support this authorized check the new construction of hybrid cloud architecture is proposed. Security analysis states that the proposed scheme is secure in terms of definitions proposed in security model. The proposed authorized duplicate check scheme intends to minimize overhead compared to normal operations.***

Keywords: Confidentiality, authorized duplicate check, hybrid cloud.

I. INTRODUCTION

Data deduplication is one of the important data compression techniques for eliminating duplicate copies of repeating data and has been widely used in cloud storage to reduce the amount of storage space and save bandwidth.

A Hybrid Cloud is a combined form of private clouds and public clouds in which some critical data resides in the enterprise's private cloud while other data is stored in and accessible from a public cloud. Hybrid clouds seek to deliver the advantages of scalability, reliability, rapid deployment and potential cost savings of public clouds with the security and increased control and management of private clouds. Deduplication[4] can take place at either the file level or the block level. For filelevel deduplication, it eliminates duplicate copies of the same file. Deduplication can also take place at the block level, which eliminates duplicate blocks of data that occur in non-identical files. Although data deduplication brings a lot of benefits, security and privacy concerns arise as users sensitive data are susceptible to both insider and outsider attacks. Traditional encryption, while providing data confidentiality, is incompatible with data de-duplication. Specifically, traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different cipher texts, making de-duplication impossible. Convergent encryption [8] has been proposed to enforce data confidentiality while making de-duplication feasible. It en-crypts/decrypts a data copy with a convergent key, which is obtained by computing the cryptographic hash value of the content of the data copy. After key generation and data encryption, users retain the keys and send the cipher text [2],[4] to the cloud. Since the encryption operation is deterministic and is derived from the data content, identical data copies will generate the same convergent key and hence the same cipher text. To prevent unauthorized access, a secure proof of ownership protocol [11] is also needed to provide the proof that the user indeed owns the same file when a duplicate is found. After the proof, subsequent users with the same file will be provided a pointer from the server without needing to upload the same file. A

user can download the encrypted file with the pointer from the server, which can only be decrypted by the corresponding data owners with their convergent keys. Thus, convergent encryption allows the cloud to perform de-duplication on the cipher texts and the proof of ownership prevents the unauthorized user to access the file. Data de-duplication system, the private cloud [15] is involved as a proxy to allow data owner/user to security perform duplicate check with differential privileges. Such architecture is practical and has attracted much attention from researchers. The data owners only outsource their data storage by utilizing public cloud while the data operation is managed in private cloud.

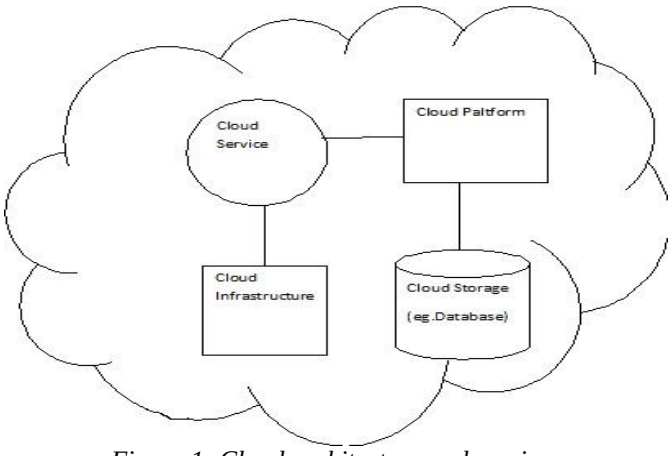

Figure 1: Cloud architecture and services

II. LITRAURE SURVEY

Cloud computing provides seemingly unlimited "virtualized" Resources to users as services across the whole Internet, while hiding platform and implementation details. Todays cloud service providers offer both highly available storage and massively parallel computing resources at relatively low costs. As cloud computing becomes prevalent, an increasing amount of data is being stored in the cloud and shared by users with specified privileges, which define the access rights of the stored data[3]. One critical challenge of cloud storage services is the management of the ever-increasing volume of data. To make data management scalable in cloud computing, deduplication has been a well-known technique and has attracted more and more attention recently. Cloud computing promises a more cost effective enabling technology to outsource storage and computations. Existing approaches for secure outsourcing of data and arbitrary computations are either based on a single tamper-proof hardware, or based on recently proposed fully homomorphic encryption. The hardware based solutions are not scaleable, and fully homomorphic encryption is currently only of theoretical interest and very inefficient.

DISADVANTAGES OF EXISTING SYSTEM

Traditional encryption, while providing data confidentiality, is incompatible with data de-duplication.

Identical data copies of different users will lead to different cipher texts, making de-duplication impossible.

III. IMPLEMENTATION

For deduplication, we enhance our system in security. Specifically, we present an advanced scheme to support stronger security by encrypting the file with differential privilege keys. In this way, the users without corresponding privileges cannot perform the duplicate check. Furthermore, such unauthorized users cannot decrypt the cipher text even cloud with the S-CSP. Security analysis demonstrates that our system is secure in terms of the definitions specified in the proposed security model.

ADVANTAGES OF PROPOSED SYSTEM

The user is only allowed to perform the duplicate check for files marked with the corresponding privileges.

We present an advanced scheme to support stronger security by encrypting the file with differential privilege keys.

Reduce the storage size of the tags for integrity check. To enhance the security of de-duplication and protect the data confidentiality.

A. HASH FUNCTION

A hash function H projects a value from a set with many (or even an infinite number of) members to a value from a set with a fixed number of (fewer) members. Hash functions are not reversible. A hash function H might, for instance, be defined as $y = H(x) = \lfloor 10\,x \pmod 1 \rfloor$, where $x \in \mathbb{R}$, $y \in [0, 9]$, and $\lfloor x \rfloor$ is the floor function.

Hash functions can be used to determine if two objects are equal (possibly with a fixed average number of mistakes). Other common uses of hash functions are checksums over a large amount of data (e.g., the cyclic redundancy check [CRC]) and finding an entry in a database by a key value.

B. ARCHITECTURE

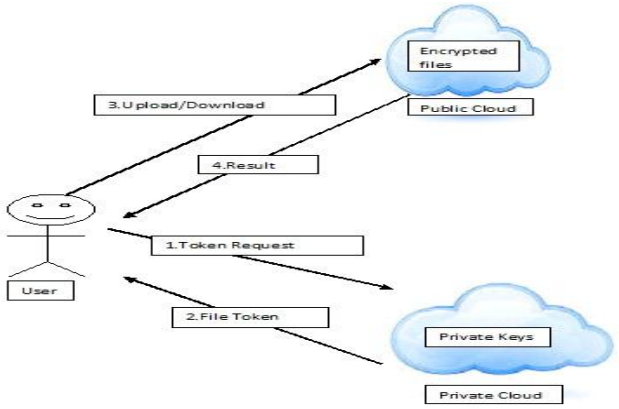

Figure 2: Architecture to avoid Replicated data

This is a Authorized De-duplication architecture for data de-duplication in cloud computing, which consists of a twin clouds (i.e., the public cloud and the private cloud).

Actually, this hybrid cloud setting has attracted more and more attention recently. For example, an enterprise might use a public cloud service, such as Amazon S3, for archived data, but continue to maintain in-house storage for operational customer data. Alternatively, the trusted private cloud could

be a cluster of virtualized cryptographic co-processors, which are offered as a service by a third party and provide the necessary hardware based security features to implement a remote execution environment trusted by the users.

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

This paper contains two main module:

a. USER

This perform following functions:

- ✓ Select a file: In this we first select a file and generate file Tag by using SHA-1 algorithm .In this module we ask the user to make file private or shared and then make request to the private server. i.e admin to generate token. But before sending request to the admin first we check file is duplicate or not.
- ✓ View file:In this we see all the file that selected by user.
- ✓ Upload file to Cloud: In this module we upload the file that we are selected.
- ✓ Download file In this it is possible download file that is uploaded by our self and shared by other by other user .

b. ADMIN

In this module admin act as private server which handle request coming from user. This perform two functions

Token-Gen(Tag, User-ID) - It loads the associated privilege keys of the user and generate the token with HMAC-SHA-1 algorithm

Share Token-Gen(Tag, {Priv.}) - It generates the share token with the corresponding privilege keys of the sharing privilege set with HMAC-SHA-1 algorithm.

Our implementation of the Storage Server provides de-duplication and data storage with following handlers and maintains a map between existing files and associated token with Hash Map.

Dup-Check(Token) - It searches the File to Token Map for Duplicate; and

File Store (File-ID, File, Token) - It stores the File on Disk and updates the Mapping.

C. ENCRYPTION OF FILE

To encrypt the user data we are using secrete key resides at the private cloud. This key is used to convert plain text to cipher text and again for the decryption of the user data. To encrypt and decrypt we have used three basic functions as follow:

KeyGenSE: In this k is the key generation algorithm which can generate the secrete file by using security parameter.

EncSE (k, M): in this M is the text message and key is the secrete key by using this both we have generated a cipher text C.

DecSE (k, C): Here C is the cipher text and k is the encryption key by using cipher text and secrete key we have to generate plain text.

D. CONFIDWNTIAL ENCRYPTION OF DATA

This ensures a data confidentiality in the duplication. User derives a convergent key from each original data and encrypt the data copy with the generated convergent key. User also add the tag for the data so that the tag will helps to detect the duplicate data. By using convergent key generation algorithm key is get generated this key is used to encrypt the user data. This will ensures the security, ownership and authority of the data.

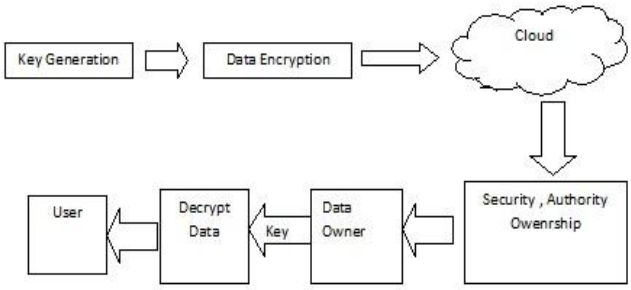

Figure 3: Confidential data encryption

E. SHA-1 ALGORITHM

a. INITIALIZE VARIABLES

h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0
ml = message length in bits (always a multiple of the number of bits in a character).

b. PRE-PROCESSING

Append the bit '1' to the message e.g. by adding 0x80 if message length is a multiple of 8 bits.

Append $0 \leq k < 512$ bits '0', such that the resulting message length in bits is congruent to $-64 \equiv 448 \pmod{512}$

Append ml, in a 64-bit big-endian integer. Thus, the total length is a multiple of 512 bits.

c. PROCESS THE MESSAGE IN SUCCESSIVE 512-BIT CHUNKS

Break message into 512-bit chunks for each chunk break chunk into sixteen 32-bit big-endian words w[i], $0 \leq i \leq 15$.

d. Extend the sixteen 32-bit words into eighty 32-bit words

for i from 16 to 79 w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) left rotate 1

e. INITIALIZE HASH VALUE FOR THIS CHUNK

a = h0
b = h1
c = h2
d = h3
e = h4

f. MAIN LOOP AS

for i from 0 to 79
if $0 \leq i \leq 19$ then
f = (b and c) or ((not b) and d)
k = 0x5A827999
else if $20 \leq i \leq 39$
f = b xor c xor d
k = 0x6ED9EBA1
else if $40 \leq i \leq 59$
f = (b and c) or (b and d) or (c and d)
k = 0x8F1BBCDC
else if $60 \leq i \leq 79$
f = b xor c xor d
k = 0xCA62C1D6
temp = (a left rotate 5) + f + e + k + w[i]
e = d
d = c
c = b left rotate 30
b = a
a = temp

g. ADD THIS CHUNK'S HASH TO RESULT SO AS

h0 = h0 + a
h1 = h1 + b
h2 = h2 + c
h3 = h3 + d
h4 = h4 + e

h. PRODUCE THE FINAL HASH VALUE (BIG-ENDIAN) AS A 160 BIT NUMBER

hh = (h0 left shift 128) or (h1 left shift 96) or (h2 left shift 64) or (h3 left shift 32) or h4

i. END

The number hh is the message digest, which can be written in hexadecimal (base 16), but is often written using Base64 binary to ASCII text encoding. The constant values used are chosen to be nothing up my sleeve numbers: the four round constants k are 2^{30} times the square roots of 2, 3, 5 and 10. The first four starting values for h0 through h3 are the same with the MD5 algorithm, and the fifth (for h4) is similar.

IV. GRAPHICAL RESULT DISSCUSION

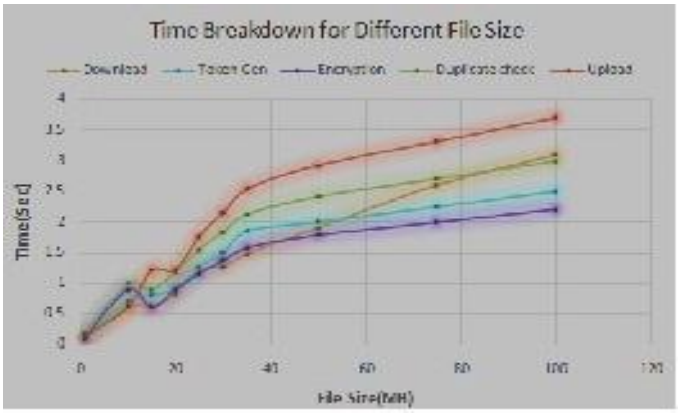

Figure 4: Time Breakdown for different file size

File Size: To evaluate the effect of file size to the time spent on different steps, we upload 100 unique files of particular file size and record the time break down. Using the unique files enables us to evaluate the worst-case scenario where we have to upload all file data. The average time of the steps from test sets of different file size are plotted in Figure 4. The time spent on downloading, encryption, upload increases linearly with the file size, since these operations involve the actual file data and incur file I/O with the whole file. In contrast, other steps such as token generation and duplicate check only use the file metadata for computation and therefore the time spent remains constant.With the file size increasing from 5MB to 11MB.

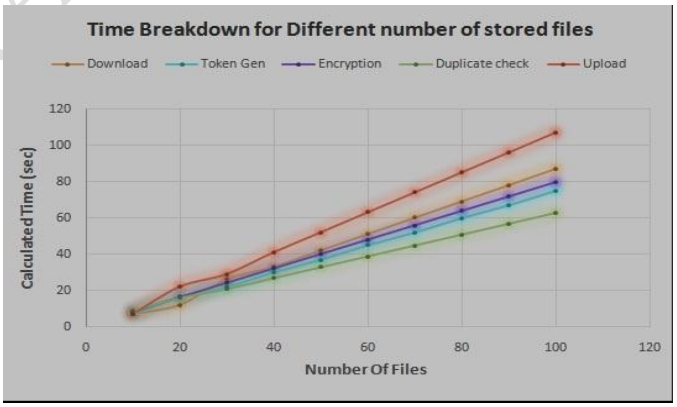

Figure 5: Time Breakdown for different number of stored file

Number of Stored Files: To evaluate the effect of number of stored files in the system, we upload different number of unique size files and record the breakdown for every file upload. From Figure 5, every step remains constant along the time. Token checking is done with ahash table and a linear search would be carried out in case of collision.

V. CONCLUSION

This paper shows that, the notion of authorized data de-duplication was proposed to protect the data security by including differential privileges of users in the duplicate check. We also presented several new de-duplication constructions supporting authorized duplicate check in hybrid cloud architecture, in which the duplicate-check tokens of files are generated by the private cloud server with private keys.

Security analysis demonstrates that our schemes are secure in terms of insider and outsider attacks specified in the proposed security model. As a proof of concept, we implemented a prototype of our proposed authorized duplicate check scheme and con-duct tested experiments on our prototype. We showed that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer.

REFERENCES

[1] A. Waghamare A secure hybrid cloud approach to avoid deduplication.In IJCSMC,Vol 4 Issue 4,April2015 pg.743-746.
[2] OpenSSL Project. http://www.openssl.org/.
[3] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In Proc. of USENIX LISA, 2010.
[4] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Server-aided encryption for deduplicated storage. In USENIX Security Symposium, 2013.
[5] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In EUROCRYPT, pages 296– 312, 2013.
[6] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. J. Cryptology, 22(1):1–61, 2009.
[7] M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concur-rent attacks. In CRYPTO, pages 162–177, 2002.
[8] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider. Twin clouds: An architecture for secure cloud computing. In Workshop on Cryptography and Security in Clouds (WCSC 2011), 2011.
[9] 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 ICDCS, pages 617–624, 2002.
[10] D. Ferraiolo and R. Kuhn. Role-based access controls. In 15th NIST-NCSC National Computer Security Conf., 1992.
[11] GNU Libmicrohttpd. http://www.gnu.org/software/libmicrohttpd/.
[12] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg. Proofs of ownership in remote storage systems. In Y. Chen, G. Danezis, and V. Shmatikov, editors, ACM Conference on Computer and Communications Security, pages 491–500. ACM, 2011.
[13] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou. Secure deduplication with efficient and reliable convergent key management. In IEEE Transactions on Parallel and Distributed Systems, 2013.
[14] libcurl. http://curl.haxx.se/libcurl/.
[15] C. Ng and P. Lee. Revdedup: A reverse deduplication storage system optimized for reads to latest backups. In Proc. of APSYS, Apr 2013.
[16] W. K. Ng, Y. Wen, and H. Zhu. Private data deduplication protocols in cloud storage. In S. Ossowski and P. Lecca, editors,
[17] Proceedings of the 27th Annual ACM Symposium on Applied Computing, pages 441–446. ACM, 2012.
[18] R. D. Pietro and A. Sorniotti. Boosting efficiency and security in proof of ownership for deduplication. In H. Y. Youm and Y. Won, editors, ACM Symposium on Information, Computer and Communications Security, pages 81–82. ACM, 2012.
[19] S. Quinlan and S. Dorward. Venti: a new approach to archival storage. In Proc. USENIX FAST, Jan 2002.