ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (34-41), Month: April - June 2015, Available at: www.researchpublish.com

Secure Authorised Deduplication by Using Hybrid Cloud Approach

[1]Boga Venkatesh, [2]Anamika Sharma, [3]Gaurav Desai, [4]Dadaram Jadhav

[1, 2, 3]B.E Students, [4]Assistant Professor, Dept. of CSE, Trinity College of Engineering, Pune, India

Abstract: **Data deduplication is one of important data compression techniques for eliminating duplicate copies of repeating data and has been widely used in cloud storage in order to minimize the amount of storage space and save bandwidth. For protection of data security, this paper makes an attempt to primarily address the problem of authorized data deduplication. To protect the confidentiality of important data while supporting deduplication, the convergent encryption technique has been proposed to encrypt the data before outsourcing. Along with the data the privilege level of the user is also checked in order to assure whether he is an authorised user or not. Security analysis demonstrates that our scheme is secure in terms of the definitions specified in the proposed security model. We show that our proposed authorized duplicate check scheme has minimal overhead compared to normal operations. As a proof of concept, we implement a prototype of our proposed authorized duplicate check scheme and conduct tested experiments using our prototype. This paper tries to minimize the data duplication that occurs in hybrid cloud storage by using various techniques.**

Keywords: **Deduplication, authorized duplicate check, confidentiality, hybrid cloud.**

I. INTRODUCTION

Cloud computing provides many "virtualized" resources to users as services across the entire Internet, while hiding platform and implementation details. Nowadays cloud service providers offer both highly available storage and massively parallel computing resources at relatively low costs. GMAIL is one of the best example of cloud storage which is used by most of us regularly. One of the major issues of cloud storage services is the management of the ever-increasing volume of data. To make data management scalable in cloud computing, deduplication [5] has been a well-known technique which is being used by most of the users. Data Deduplication is one of the specialised data compression technique which is used to eliminate duplicate copies of data. Deduplication can take place at file level or either block level. For file level 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 there are several advantages of data deduplication security and privacy concerns arise as users' sensitive data are susceptible to both insider and outsider attacks. Encryption techniques which were used traditionally were not compatible with data deduplication while providing data confidentiality. Traditional encryption requires different users to encrypt their data with their own keys by which identical data copies of different users will lead to different cipher texts, making deduplication impossible. Convergent encryption [4] has been proposed to enforce data confidentiality while making deduplication feasible. It encrypts/decrypts a data copy with a *convergent key*, which is obtained by computing the cryptographic hash value of the content of the data copy. Whenever the key is generated users retain the keys and send the cipher text to the cloud. In order to prevent unauthorized access, a secure proof of ownership protocol [2] is also needed to provide the proof that the user indeed owns the same file when a duplicate is found. Hence convergent encryption allows the cloud to perform deduplication on the ciphertexts and the proof of ownership prevents the unauthorized user to access the file. Traditional deduplication systems based on convergent encryption, although providing confidentiality to some extent; do not support the duplicate check with differential privileges. Contradiction occurs when we try to realize both deduplication and differential authorization duplicate check at the same time.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (34-41), Month: April - June 2015, Available at: www.researchpublish.com

1.1 CONTRIBUTION:

For solving the problems of deduplication we consider a hybrid cloud architecture consisting of a public cloud and a private cloud. Different privilege levels have been allocated to securely perform duplicate check in private cloud. A new deduplication system supporting differential duplicate check is proposed under this hybrid cloud architecture where the S-CSP resides in the public cloud. The user is only allowed to perform the duplicate check for files marked with the corresponding privileges. Finally, we implement a prototype of the proposed authorized duplicate check and conduct tested experiments to evaluate the overhead of the prototype and show that the overhead is minimal compared to the normal convergent encryption.

Acronym	Description
S-CSP	Storage-cloud service provider
PoW	Proof of Ownership
(pk_U, sk_U)	User's public and secret key pair
k_F	Convergent encryption key for file F
P_U	Privilege set of a user U
P_F	Specified privilege set of a file F
$\phi'_{F,p}$	Token of file F with privilege p

TABLE 1
Notations Used in This Paper

II. PRELIMINARIES

In this section, we first define the notations used in this paper, review some secure primitives used in our secure deduplication.

Symmetric encryption. Symmetric encryption uses a common secret key *to* encrypt and decrypt information.
A symmetric encryption scheme consists of three Primitive functions:

- KeyGenSE(1_) *!* κ is the key generation algorithm that generates κ using security parameter 1;
- DecSE(κ,*C*) *!* *M* is the symmetric decryption algorithm that takes the secret κ and ciphertext *C* and then outputs the original message *M*.
- EncSE(κ,*M*) *!* *C* is the symmetric encryption algorithm that takes the secret κ and message *M* and then outputs the ciphertext *C*.

Convergent encryption. Convergent encryption [3], [4] provides data confidentiality in deduplication. A data owner derives a convergent key from each original data copy and encrypts the data copy with the convergent key.

- KeyGenCE(*M*) *!* *K* is the key generation algorithm that maps a data copy *M* to a convergent key *K*
- EncCE(*K*,*M*) *!* *C* is the symmetric encryption algorithm that takes both the convergent key *K* and the data copy *M* as inputs and then outputs a ciphertext *C*.
- DecCE(*K*,*C*) *!* *M* is the decryption algorithm that takes both the ciphertext *C* and the convergent key *K* as inputs and then outputs the original data copy *M*; and
- TagGen(*M*) *!* *T*(*M*) is the tag generation algorithm that maps the original data copy *M* and outputs a tag *T*(*M*).
- DecSE(κ,*C*) *!* *M* is the symmetric decryption algorithm that takes the secret κ and ciphertext *C* and then outputs the original message *M*.

Proof of ownership. The notion of proof of ownership (PoW) [2] enables users to prove their ownership of data copies to the storage server. However PoW is implemented as an interactive algorithm (denoted by PoW) run by a prover (i.e., user) and a verifier (i.e.storage server).

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (34-41), Month: April - June 2015, Available at: www.researchpublish.com

Identification Protocol. An identification protocol can be described with two phases: Proof and Verify. In the stage of Proof, a prover/user *U* can demonstrate his identity to a verifier by performing some identification proof related to his identity. The input of the prover/user is his private key *skU* that is sensitive information such as private key of a public key in his certificate or credit card number etc. that he would not like to share with the other users.

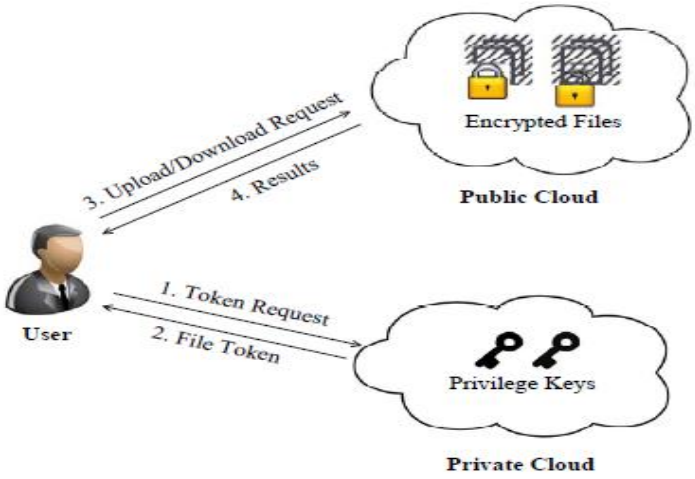

Fig. 1. Architecture for Authorized Deduplication

III. SYSTEM MODEL

3.1 Hybrid Architecture for Secure Deduplication:

At a high level, our setting of interest is an enterprise network, consisting of a group of affiliated clients (for example, employees of a company) who will use the S-CSP and store data with deduplication technique. The S-CSP performs deduplication by checking if the contents of two files are the same and stores only one of them. Each privilege is represented in the form of a short message called *token*. Each file is associated with some *file tokens*, which denote the tag with specified privileges. Role of the private cloud server will be explained in the paper. block-level deduplication can be easily deduced from file-level deduplication, which is similar to [5]. Specifically, to upload a file, a user first performs the file-level duplicate check. If the file is a duplicate, then all its blocks must be duplicates as well.

- *S-CSP.* This is an entity that provides a data storage service in public cloud. The S-CSP provides the data outsourcing service and stores data on behalf of the users.
- *Data Users.* A user is an entity that wants to outsource data storage to the S-CSP and access the data later. In a storage system supporting deduplication, the user only uploads unique data but does not upload any duplicate data to save the upload bandwidth, which may be owned by the same use or different users. Every single file is protected with the convergent encryption key and privilege keys to realize the authorized deduplication with differential privileges.
- *Private Cloud.* Compared with the traditional deduplication architecture in cloud computing, this is a new entity introduced for facilitating user's secure usage of cloud service. Private Keys are managed by private cloud in order to give them privileges as per their designation.

3.2 Design Goals:

We have proposed a new deduplication system for the following:

- *Differential Authorization.* Each authorized user is able to get his/her individual token of his file to perform duplicate check based on his privileges. Under this assumption, any user cannot generate a token for duplicate check out of his privileges or without the aid from the private cloud server.
- *Authorized Duplicate Check.* Authorized user is able to use his/her individual private keys to generate query for certain file and the privileges he/she owned with the help of private cloud, while the public cloud performs duplicate check directly and tells the user if there is any duplicate.

- *Unforgeability of file token/duplicate-check token.* Unauthorized users without appropriate privileges or file should be prevented from getting or generating the file tokens for duplicate check of any file stored at the S-CSP. The duplicate check token of users should be issued from the private cloud server in our scheme

IV. SECURE DEDUPLICATION SYSTEMS

MAIN IDEA. o support authorized deduplication, the tag of a file *F* will be determined by the file *F* and the privilege. To show the difference with traditional notation of tag, we call it file token instead. To support authorized access, a secret key *kp* will be bounded with a privilege *p* to generate a file token. Let $\phi' F;p$ = TagGen(*F*, *kp*) denote the token of *F* that is only allowed to access by user with privilege *p*. In another word, the token $\phi' F;p$ could only be computed by the users with privilege *p*. As a result, if a file has been uploaded by a user with a duplicate token $\phi' F;p$, then a duplicate check sent from another user will be successful if and only if he also has the file *F* and privilege *p*. Such a token generation function could be easily implemented as *H*(*F*, *kp*), where *H*(_) denotes a cryptographic hash function.

4.1 A First Attempt:

Before introducing our construction of differential deduplication, we present a straightforward attempt with the technique of token generation TagGen(*F*, *kp*) above to design such a deduplication system. The main idea of this basic construction is to issue corresponding privilege keys to each user, who will compute the file tokens and perform the duplicate check based on the privilege keys and files. In more details, suppose that there are *N* users in the system and the privileges in the universe is defined as *P* = *fp*1, . . . , *psg*. For each privilege *p* in *P*, a private key *kp* will be selected. For a user *U* with a set of privileges *PU*, he will be assigned the set of keys *fkpi gpi∈PU* .

File Uploading. Suppose that a data owner *U* with privilege set *PU* wants to upload and share a file *F* with users who have the privilege set *PF* = *fpjg*. The user computes and sends S-CSP the file token $\phi' F;p$ = TagGen(*F*, *kp*) for all *p 2 PF*.

• If a duplicate is found by the S-CSP, the user proceeds proof of ownership of this file with the S-CSP. If the proof is passed, the user will be assigned a pointer, which allows him to access the file.

• Otherwise, if no duplicate is found, the user computes the encrypted file *CF* = EncCE(*kF* , *F*) with the convergent key *kF* = KeyGenCE(*F*) and uploads (*CF* , $f\phi' F;p$ *g*) to the cloud server. The convergent key *kF* is stored by the user locally.

File Retrieving. Suppose a user wants to download a file *F*. It first sends a request and the file name to the S-CSP. Upon receiving the request and file name, the S-CSP will check whether the user is eligible to download *F*. If failed, the S-CSP sends back an abort signal to the user to indicate the download failure. Otherwise, the S-CSP returns the corresponding ciphertext *CF* . Upon receiving the encrypted data from the S-CSP, the user uses the key *kF* stored locally to recover the original €file *F*.

Problems. Such a construction of authorized deduplication has several serious security problems, which are listed below.

• First, each user will be issued private keys *Fkpi gpi∈PU* for their corresponding privileges, denoted by *PU* in our above construction. These private keys *fkpi gpi∈PU* can be applied by the user to generate file token for duplicate check. However, during file uploading, the user needs to compute file tokens for sharing with other users with privileges *PF* . To compute these file tokens, the user needs to know the private keys for *PF* , which means *PF* could only be chosen from *PU*. Such a restriction makes the authorized deduplication system unable to be widely used and limited.

• Second, the above deduplication system cannot prevent the privilege private key sharing among users. The users will be issued the same private key for the same privilege in the construction. As a result, the users may collude and generate privilege private keys for a new privilege set $P*$ that does not belong to any of the colluded user. For example, a user with privilege set *PU*1 may collude with another user with privilege set *PU*2 to get a privilege set $P* = PU1 \lceil PU2$.

• The construction is inherently subject to brute-force attacks that can recover files falling into a known set. That is, the deduplication system cannot protect the security of predictable files. One of critical reasons is that the traditional convergent encryption system can only protect the semantic security of unpredictable files.

4.2 Proposed System Description:

We have introduced a hybrid cloud architecture in our proposed deduplication system. The private keys for privileges will not be issued to users directly, which will be kept and managed by the private cloud server instead. In this way, the

users cannot share these private keys of privileges in this proposed construction, which means that it can prevent the privilege key sharing among users in the above straightforward construction. To get a file token, the user needs to send a request to the private cloud server. The intuition of this construction can be described as follows. To perform the duplicate check for some file, the user needs to get the file token from the private cloud server. The private cloud server will also check the user's identity before issuing the corresponding file token to the user. The authorized duplicate check for this file can be performed by the user with the public cloud before uploading this file. Based on the results of duplicate check, the user either uploads this file or runs PoW.

Before giving our construction of the deduplication system, we define a binary relation R = *f*((*p, p′*)*g* as follows. Given two privileges *p* and *p′*, we say that *p* matches *p′* if and only if R(*p, p′*) = 1.

System Setup. An identification protocol _ = (Proof, Verify) is also defined, where Proof and Verify are the proof and verification algorithm respectively. Furthermore, each user *U* is assumed to have a secret key *skU* to perform the identification with servers. Assume that user *U* has the privilege set *PU*. It also initializes a PoW protocol POW for the file ownership proof. The private cloud server will maintain a table which stores each user's public information *pkU* and its corresponding privilege set *PU*. The file storage system for the storage server is set to be _|_

File Uploading. Suppose that a data owner wants to upload and share a file *F* with users whose privilege belongs to the set *PF* = *fpjg*. The data owner needs interact with the private cloud before performing duplicate check with the S-CSP. Data owner performs an identification to prove its identity with private key *skU*. If it is passed, the private cloud server will find the corresponding privileges *PU* of the user from its stored table list. The user computes and sends the file tag ϕF = TagGen(*F*) to the private cloud server, who will return $f\phi' F;p\_$ = TagGen(ϕF , *kp_*)*g* back to the user for all *p_* satisfying R(*p, p_*) = 1 and *p 2 PU*. Then, the user will interact and send the file token $f\phi' F;p\_ g$ to the S-CSP.

• If a file duplicate is found, the user needs to run the PoW protocol POW with the S-CSP to prove the file ownership. If the proof is passed, the user will be provided a pointer for the file. Furthermore, a proof from the S-CSP will be returned, which could be a signature on $f\phi' F;p\_ g$, *pkU* and a time stamp. The user sends the privilege set *PF* = *fpjg* for the file *F* as well as the proof to the private cloud server. Upon receiving the request, the private cloud server first verifies the proof from the S-CSP. If it is passed, the private cloud server computes $f\phi' F;p\_$ = TagGen(ϕF , *kp_*)*g* for all *p_* satisfying R(*p, p_*) = 1 for each *p 2 PF -PU*, which will be returned to the user. The user also uploads these tokens of the file *F* to the private cloud server. Then, the privilege set of the file is set to be the union of *PF* and the privilege sets defined by the other data owners.

• Otherwise, if no duplicate is found, a proof from the S-CSP will be returned, which is also a signature on $f\phi' F;p\_ g$, *pkU* and a time stamp. The user sends the privilege set *PF* = *fpjg* for the file *F* as well as the proof to the private cloud server. Upon receiving the request, the private cloud server first verifies the proof from the S-CSP. If it is passed, the private cloud server computes $f\phi' F;p\_$ = TagGen(ϕF , *kp_*)*g* for all *p_* satisfying R(*p, p_*) = 1 and *p 2 PF* . Finally, the user computes the encrypted file *CF* = EncCE(*kF* , *F*) with the convergent key *kF* = KeyGenCE(*F*) and uploads *fCF* , $f\phi' F;p\_$ *Gg* with privilege *PF* .

File Retrieving. The user downloads his files in the same way as the deduplication system in Section 4.1. That is, the user can recover the original file with the convergent key *kF* after receiving the encrypted data from the S-CSP.

4.3 Further Enhancement:

We design and implement a new system which could protect the security for predicatable message. The *main idea* of our technique is that the novel encryption key generation algorithm. For simplicity, we will use the hash functions to define the tag generation functions and convergent keys in this section. In traditional convergent encryption, to support duplicate check, the key is derived from the file *F* by using some cryptographic hash function *kF* = *H*(*F*). To avoid the deterministic key generation, the encryption key *kF* for file *F* in our system will be generated with the aid of the private key cloud server with privilege key *kp*. The encryption key can be viewed as the form of *kF;p* = *H*0(*H*(*F*), *kp*) ⊕*H*2(*F*), where *H*0,*H* and *H*2 are all cryptographic hash functions. The file *F* is encrypted with another key *k*, while *k* will be encrypted with *kF;p*. In this way, both the private cloud server and S-CSP cannot decrypt the ciphertext. Furthermore, it is semantically secure to the S-CSP based on the security of symmetric encryption. //////////with the traditional deduplication architecture in cloud computing, this is a new entity introduced for

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (34-41), Month: April - June 2015, Available at: www.researchpublish.com

V. SECURITY ANALYSIS

Proposed system has been designed to solve the differential privilege problem in secure deduplication. The security will be analyzed in terms of two aspects, that is, the authorization of duplicate check and the confidentiality of data.

5.1 Security of Duplicate-Check Token:

We consider several types of privacy we need protect, that is, i) unforgeability of duplicate-check token: There are two types of adversaries, that is, external adversary and internal adversary. As shown below, the external adversary can be viewed as an internal adversary without any privilege. If a user has privilege p, it requires that the adversary cannot forge and output a valid duplicate token with any other privilege p' on any file F, where p does not match p'. Furthermore, it also requires that if the adversary does not make a request of token with its own privilege from private cloud server, it cannot forget and output a valid duplicate token with p on any F that has been queried. The internal adversaries have more attack power than the external adversaries and thus we only need to consider the security against the internal attacker, ii) indistinguishability of duplicate check token : this property is also defined in terms of two aspects as the definition of unforgeability. First, if a user has privilege p, given a token ϕ', it requires that the adversary cannot distinguish which privilege or file in the token if p does not match p'.

VI. IMPLEMENTATION

We implement a prototype of the proposed authorized deduplication system, in which we model three entities as separate C++ programs. A *Client* program is used to model the data users to carry out the file upload process. A *Private Server* program is used to model the private cloud which manages the private keys and handles the file token computation. A *Storage Server* program is used to model the S-CSP which stores and deduplicate files.

Our implementation of the **Client** provides the following function calls to support token generation and deduplication along the file upload process.

• FileTag(File) - It computes SHA-1 hash of the File as File Tag;

• TokenReq(Tag, UserID) - It requests the Private Server for File Token generation with the File Tag and User ID;

• DupCheckReq(Token) - It requests the Storage Server for Duplicate Check of the File by sending the file token received from private server;

• ShareTokenReq(Tag, {Priv.}) - It requests the Private Server to generate the Share File Token with the File Tag and Target Sharing Privilege Set;

• FileEncrypt(File) - It encrypts the File with Convergent Encryption using 256-bit AES algorithm in cipher block chaining (CBC) mode, where the convergent key is from SHA-256 Hashing of the file;

• FileUploadReq(FileID, File, Token) – It uploads the File Data to the Storage Server if the file is Unique and updates the File Token stored. Our implementation of the **Private Server** includes corresponding request handlers for the token generation and maintains a key storage with Hash Map.

• TokenGen(Tag, UserID) - It loads the associated privilege keys of the user and generate the token with HMAC-SHA-1 algorithm

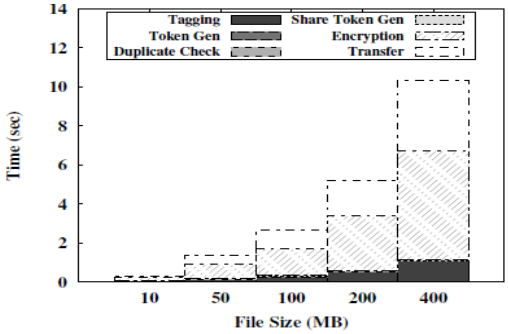

Fig. 2. Time Breakdown for Different File Size

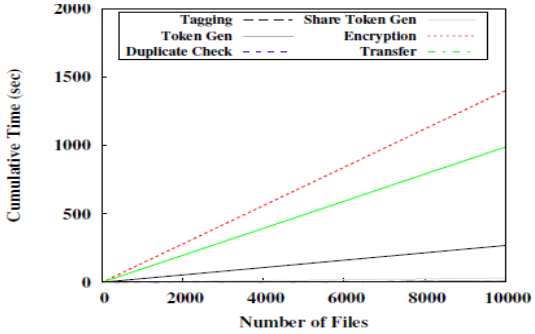

Fig. 3. Time Breakdown for Different Number of Stored Files

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (34-41), Month: April - June 2015, Available at: www.researchpublish.com

VII. ASSESSMENT

Our evaluation focuses on comparing the overhead induced by authorization steps, including file token generation and share token generation, against the convergent encryption and file upload steps. We evaluate the overhead by varying different factors, including 1) File Size 2) Number of Stored Files 3) Deduplication Ratio 4) Privilege Set Size.

We break down the upload process into 6 steps, 1) Tagging 2) Token Generation 3) Duplicate Check 4) Share Token Generation 5) Encryption 6) Transfer . For each step, we record the start and end time of it and therefore obtain the breakdown of the total time spent. We present the average time taken in each data set in the figures

7.1 File Size:

To evaluate the effect of file size to the time spent on different steps, we upload 100 unique files (i.e., without any deduplication opportunity) 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 2. The time spent on tagging, 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.

7.2 Number of Stored Files:

To evaluate the effect of number of stored files in the system, we upload 10000 10MB unique files to the system and record the breakdown for every file upload. From Figure 3, every step remains constant along the time. Token checking is done with a hash table and a linear search would be carried out in case of collision.

7.3 Deduplication Ratio:

To evaluate the effect of the deduplication ratio, we prepare two unique data sets, each of which consists of 50

100MB files. We first upload the first set as an initial upload. For the second upload, we pick a portion of 50 files, according to the given deduplication ratio, from the initial set as duplicate files and remaining files from the second set as unique files. The average time of uploading the second set is presented in Figure 4.

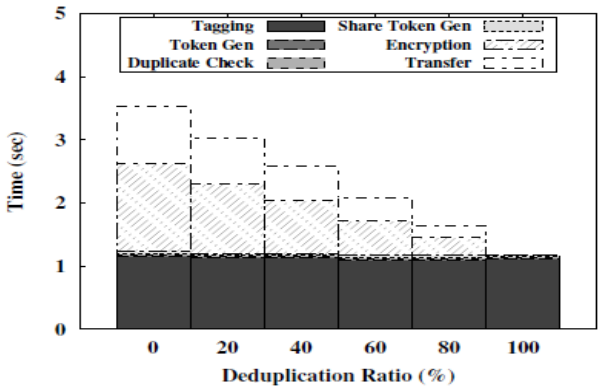

Fig. 4. Time Breakdown for Different Deduplication Ratio

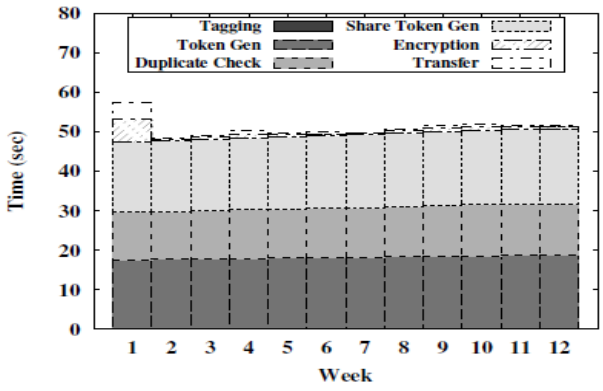

Fig. 6. Time Breakdown for the VM dataset.

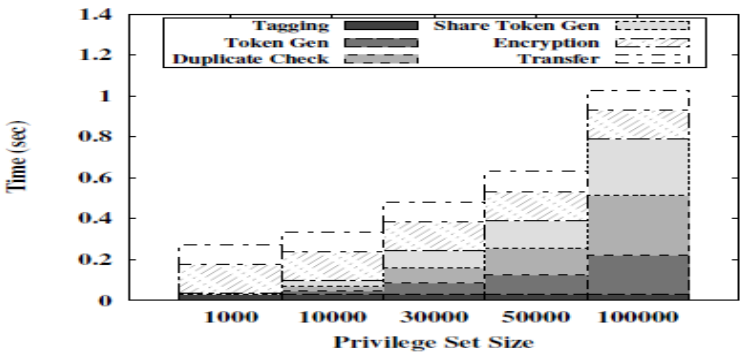

Fig. 5. Time Breakdown for Different Privilege Set Size

7.4 Privelege Set Size:

To evaluate the effect of privilege set size, we upload 100 10MB unique files with different size of the data owner and target share privilege set size. In Figure 5, it shows the time taken in token generation increases linearly as more keys are associated with the file and also the duplicate check time. While the number of keys increases 100 times from 1000 to 100000, the total time spent only increases to 3.81 times and it is noted that the file size of the experiment is set at a small level (10MB), the effect would become less significant in case of larger files.

7.5 Summary:

To conclude the findings, the token generation introduces only minimal overhead in the entire upload process and is negligible for moderate file sizes, for example, less than 2% with 100MB files. This suggests that the scheme is suitable to construct an authorized deduplication system for backup storage.

VIII. CONCLUSION

In this paper, the notion of authorized data deduplication was proposed to protect the data security by including differential privileges of users in the duplicate check. We also presented several new deduplication 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 conduct testbed experiments on our prototype. We showed that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer.

REFERENCES

[1] S. Quinlan and S. Dorward. Venti: a new approach to archival storage. In *Proc. USENIX FAST*, Jan 2002.

[2] 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.

[3] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In *EUROCRYPT*, pages 296– 312, 2013.

[4] 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.

[5] J. Li, X. Chen, M. Li, J. Li, P. Lee, andW. Lou. Secure deduplication with efficient and reliable convergent key management. In *IEEE Transactions on Parallel and Distributed Systems*, 2013.