International Journal of Research
Available at https://edupediapublications.org/journals

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 02 Issue 10
October 2015

A Hybrid Cloud Approach for Secure Authorized Deduplication

Samala Pravalika
M.Tech student
Email id:pravali.2001@gmail.com
Department of computer Science and Engineering
Aurora's Technological & Research Institute

Mrs.N.Nirmala Jyothi
Senior Associate Professor
Email id:nirmala.narisetty@gmail.com
Department of computer Science and Engineering and Information Technology
Aurora's Technological & Research Institute

Abstract—**Data deduplication is one of important techniques for eliminating duplicate copies of existing data, and is widely used in cloud storage to minimize the amount of storage space and save bandwidth. To protect the confidentiality of available data while supporting deduplication, the convergent encryption technique is proposed to encrypt the data before outsourcing. To protect data security, this paper does make the first attempt to formally address the problem of authorized data deduplication. Different from traditional deduplication systems, the differential privileges of users are further considered in duplicate check besides the data itself. We also present different new deduplication constructions supporting authorized duplicate check in a hybrid cloud architecture. Security analysis demonstrates that our system is secure in terms of the definitions specified in the proposed security model. We implement a prototype of our proposed authorized duplicate check scheme and conduct testbed experiments using our prototype, As a proof of concept. We show that our proposed authorized duplicate check scheme incurs minimal overhead compared to normal operations.**

1 INTRODUCTION

For data management scalable in cloud computing, deduplication is a well-known technique. Data deduplication is a specialized data compression technique for removing duplicate copies of data in storage. The technique is used to improve storage utilization and we can also apply to network data transfers to reduce the number of bytes that must be sent. Instead of keeping multiple copies of data with the same content, deduplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy. Deduplication can take either the file level or the block level.

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 confiden-tiality, is incompatible with data deduplication. Specif-ically, traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different ci-phertexts, making deduplication impossible. Convergent encryption [8] has been proposed to enforce data con-fidentiality while making deduplication 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 ciphertext to the cloud. Since the encryption operation is deterministic and is derived from the data content, iden-tical data copies will generate the same convergent key and hence the same ciphertext. 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 deduplication on the ciphertexts and the proof of ownership prevents the unauthorized user to access the file

However, previous deduplication systems cannot sup-port *differential authorization duplicate check*, which is im-portant in many applications. In such an authorized deduplication system, each user is issued a set of priv-ileges during system initialization (in Section 3, we elaborate the definition of a privilege with examples). Each file uploaded to the cloud is also bounded by a set of privileges to specify which kind of users is allowed to perform the duplicate check and access the files. Before submitting his duplicate check request for some file, the user needs to take this file and his own privileges as inputs. The user is able to find a duplicate for this file if and only if there is a copy of this file and a matched privilege stored in cloud. For example, in a company, many different privileges will be assigned to employees. In order to save cost and efficiently management, the data will be moved to the storage server provider (S-CSP) in the public cloud with specified privileges and the deduplication technique will be applied to store only one copy of the same file. Because of privacy consid-eration, some files will be encrypted and allowed the duplicate check by employees with specified privileges to realize the access control. Traditional deduplication systems based on convergent encryption, although pro-viding confidentiality to some extent, do not support the duplicate check with differential privileges. In other words, no differential privileges have been considered in the deduplication based on convergent encryption technique. It seems to be contradicted if we want to realize both deduplication and differential authorization duplicate check at the same time.

1.1 Contributions

In deduplication with differential privileges in cloud com-puting, we consider a hybrid cloud architecture consist-ing of a public cloud and a private cloud. Unlike existing data deduplication systems, the private cloud is involved as a proxy to allow data owner/users to securely per-form duplicate check with differential privileges. Such an 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. A new dedu-plication 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.

Furthermore, 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 correspond-ing privileges cannot perform the duplicate check. Fur-thermore, such unauthorized users cannot decrypt the ciphertext even collude with the S-CSP. Security analysis demonstrates that our system is secure in terms of the definitions specified in the proposed security model.

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

Finally, we implement a prototype of the proposed authorized duplicate check and conduct testbed exper-iments to evaluate the overhead of the prototype. We show that the overhead is minimal compared to the nor-mal convergent encryption and file upload operations.

2 PRELIMINARIES

In this section, we first define the notations used in this paper, review some secure primitives used in our secure deduplication. The notations used in this paper are listed in TABLE 1.

Symmetric encryption. Symmetric encryption uses a common secret key κ to encrypt and decrypt informa-tion. A symmetric encryption scheme consists of three primitive functions:

- $\mathrm{KeyGen}_{SE}(1\ )\ !\ \varkappa$ is the key generation algorithm that generates $\varkappa$ using security parameter 1 ;
- $\mathrm{Enc}_{SE}(\varkappa, M)\ !\ C$ is the symmetric encryption algo-rithm that takes the secret $\varkappa$ and message M and then outputs the ciphertext C; and
- $\mathrm{Dec}_{SE}(\varkappa, C)\ !\ M$ is the symmetric decryption algo-rithm that takes the secret $\varkappa$ and ciphertext C and then outputs the original message M.
-

Convergent encryption. Convergent encryption [4], [8] provides data confidentiality in deduplication. A user (or data owner) derives a convergent key from each original data copy and encrypts the data copy with the convergent key. In addition, the user also derives a *tag* for the data copy, such that the tag will be used to detect duplicates. Here, we assume that the tag correctness property [4] holds, i.e., if two data copies are the same, then their tags are the same. To detect duplicates, the user first sends the tag to the server side to check if the identical copy has been already stored. Note that both the convergent key and the tag are independently derived, and the tag cannot be used to deduce the convergent key and compromise data confidentiality.

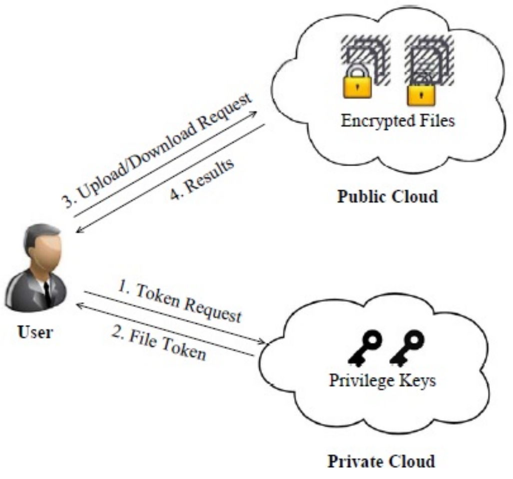

Fig. 1. Architecture for Authorized Deduplication

3 SYSTEM MODEL

3.1 Hybrid Architecture for Secure Deduplication

At a high level, our setting of interest is an enterprise network, consists of a group of affiliated clients who will use the S-CSP and store data with deduplication technique. In this setting, deduplication can be frequently used in these settings for data backup and disaster recovery applications while greatly reducing storage space. That type of systems are widespread and are more suitable to backup and synchronization applications than richer storage abstractions. There are three entities defined in our system, that is, *users*, *private cloud* and S-CSP in *public cloud* as shown in Fig. 1. The S-CSP performs deduplication by checking if the contents of two files are the same and stores only one of them.

The access right to a file is defined based on a set of *privileges*. The exact definition of a privilege varies across applications. For example, we may define a *role-based* privilege [9], [19] according to job positions (e.g., Director, Project Lead, and Engineer), or we may define a *time-based* privilege that specifies a valid time period (e.g., 2014-01-01 to 2014-01-31) within which a file can be accessed. A user, say Alice, may be assigned two privileges “Director” and “access right valid on 2014-01-01”, so that she can access any file whose access role is “Director” and accessible time period covers 2014-01-01. 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 (see the definition of a tag in Section 2). A user computes and sends *duplicate-check tokens* to the public cloud for authorized duplicate check.

Users have access to the private cloud server, a semi-trusted third party which will aid in performing dedu-plicable encryption by generating file tokens for the requesting users. We will explain further the role of the private cloud server below. Users are also provisioned with per-user encryption keys and credentials (e.g., user certificates). In this paper, we will only consider the file-level deduplication for simplicity. In another word, we refer a data copy to be a whole file and file-level dedu-plication which eliminates the storage of any redundant files. Actually, block-level deduplication can be easily deduced from file-level deduplication, which is similar to [12]. 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; otherwise, the user further performs the block-level duplicate check and identifies the unique blocks to be uploaded. Each data copy (i.e., a file or a block) is associated with a token for the duplicate check.

- *S-CSP*. This is an entity that provides a information storage service in cloud. The S-CSP provides the data outsourcing service and stores data on behalf of the users. To reduce the storage cost, the S-CSP eliminates the storage of redundant data via dedu-plication and keeps only unique data. In this paper, we assume that S-CSP is always online and has abundant storage capacity and computation power.
- *Data Users*. A user is an entity that wants to out-source data storage to the S-CSP and access the data later. In a storage system supporting dedupli-cation, 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 user or different users. In the authorized deduplication system, each user is issued a set of privileges in the setup of the system. Each file is protected with the convergent encryption key and privilege keys to re-alize the authorized deduplication with differential

privileges.

- *Private Cloud.* Compared with the traditional dedu-
- plication architecture in cloud computing, this is a new entity introduced for facilitating user's se-cure usage of cloud service. Specifically, since the computing resources at data user/owner side are restricted and the public cloud is not fully trusted in practice, private cloud is able to provide data user/owner with an execution environment and infrastructure working as an interface between user and the public cloud. The private keys for the privileges are managed by the private cloud, who answers the file token requests from the users. The interface offered by the private cloud allows user to submit files and queries to be securely stored and computed respectively.

This is a novel architecture, that consists of a twin clouds and this hybrid cloud setting has attracted so many 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 crypto-graphic 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.

3.2 Adversary Model

Typically, we assume that the public cloud and private cloud are both "honest-but-curious". Specifically they will follow our proposed protocol, but try to find out as much secret information as possible based on their possessions. Users would try to access data either within or out of the scopes of their privileges.

In this paper, we suppose that all the files are sen-sitive and needed to be fully protected against both public cloud and private cloud. Under the assumption, two kinds of adversaries are considered, that is, 1) external adversaries which aim to extract secret infor-mation as much as possible from both public cloud and private cloud; 2) internal adversaries who aim to obtain more information on the file from the public cloud and duplicate-check token information from the private cloud outside of their scopes. Such adversaries may include S-CSP, private cloud server and authorized users. The detailed security definitions against these adversaries are discussed below and in Section 5, where attacks launched by external adversaries are viewed as special attacks from internal adversaries.

3.3 Design Goals

In this paper, we address the problem of privacy-preserving deduplication in cloud computing and pro-pose a new deduplication system supporting for

- *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.* Unau-thorized 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 users are not allowed to collude with the public cloud server to break the unforgeability of file tokens. In our system, the S-CSP is honest but curious and will honestly perform the duplicate check upon receiving the duplicate request from users. The duplicate check token of users should be issued from the private cloud server in our scheme
- *Data Confidentiality.* Unauthorized users without ap-propriate privileges or files, including the S-CSP and the private cloud server, should be prevented from access to the underlying plaintext stored at S-CSP. In another word, the goal of the adversary is to retrieve and recover the files that do not belong to them. In our system, compared to the previous def-inition of data confidentiality based on convergent encryption, a higher level confidentiality is defined and achieved.

4 SECURE DEDUPLICATION SYSTEMS

The main Idea for support authorized deduplication, the tag of a file F is determined by the file F and the priv-ilege. To show the difference with traditional notation of tag, we call it file token instead. For supporting authorized access, a secret key k_p is bounded with a privilege p to generate a file token. Let $\phi'_{F;p}$ =

TagGen(F, k_p) denote the token of F that is only allowed to access by user with privilege p. In another word, the token $\phi'_{F;p}$ is 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 $H(F, k_p)$, 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, k_p) 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 = $fp_1, \ldots, p_s g$. For each privilege p in P, a private key k_p will be selected. For a user U with a set of privileges P_U , he will be assigned the set of keys $fk_{pi}\ g_{pi□PU}$.

File Uploading. Suppose that a data owner U with privilege set P_U wants to upload and share a file F with users who have the privilege set $P_F = fp_j\ g$.

The user computes and sends S-CSP the file token $\phi'_{F;p}$ = TagGen(F, k_p) for all $p\ 2\ P_F$.

- 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 C_F = Enc$_{CE}$(k_F , F) with the convergent key k_F = KeyGen$_{CE}$(F) and uploads (C_F , $f\phi'_{F;p}g$) to the cloud server. The convergent key k_F 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 C_F . Upon receiving the encrypted data from the S-CSP, the user uses the key k_F stored locally to recover the original file F .

4.2 Our Proposed System Description

In this new dedupli-cation system, a hybrid cloud architecture is introduced to solve the problem. 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.. 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. This kind of a generic binary relation definition could be instantiated based on the background of applications, such as the common hierarchical relation. More precisely, in a hierarchical relation, p matches p' if p is a higher-level privi-lege. For example, in an enterprise management system, three hierarchical privilege levels are defined as Director, Project lead, and Engineer, where Director is at the top level and Engineer is at the bottom level. Obviously, in this simple example, the privilege of Director matches the privileges of Project lead and Engineer. We provide the proposed deduplication system as follows.

System Setup. The privilege universe P is defined as in Section 4.1. A symmetric key k_{pi} for each $p_i\ 2\ P$ will be selected and the set of keys $fk_{pi}\ g_{pi□P}$ will be sent to the private cloud. An identification protocol = (Proof, Verify) is also defined, where Proof and Verify are the proof and verification algorithm respec-tively. Furthermore, each user U is assumed to have a secret key sk_U to perform the identification with servers. Assume that user U has the privilege set P_U . 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 pk_U and its corresponding privilege set P_U . The file storage system for the storage server is set to be ?.

5 SECURITY ANALYSIS

Our system is designed to solve the differential privilege problem in secure deduplication. The security is analyzed in terms of two aspects, i.e., the authoriza-tion of duplicate check and the confidentiality of data. Some basic tools are used to construct the secure deduplication. These basic tools include the convergent encryption scheme, symmetric encryption scheme, and the PoW scheme. Based on this assumption, we show that systems are secure with respect to the following analysis.

5.1 Security of Duplicate-Check Token

We consider different 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 forge 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'. Furthermore, it also require that if the adversary does not make a request of token with its own privilege from private cloud server, it cannot distinguish a valid duplicate token with p on any other F that the adversary has not queried. In the security definition of indistinguishablity, we require that the adversary is not allowed to collude with the public cloud servers. Actually, such an assumption could be removed if the private cloud server maintains the tag list for all the files uploaded. Similar to the analysis of unforgeability, the security against external adversaries is implied in the security against the internal adversaries.

Indistinguishiability of duplicate-check token

The security of indistinguishability of token can be also proved based on the assumption of the underlying message authentication code is secure. The security of message authentication code requires that the adversary cannot distinguish if a code is generated from an un-known key. In our deduplication system, all the privilege keys are kept secret by the private cloud server. Thus, even if a user has privilege p, given a token ϕ', the adversary cannot distinguish which privilege or file in the token because he does not have the knowledge of privilege key sk_p.

5.2 Confidentiality of Data

The data will be encrypted in our deduplication system before outsourcing to the S-CSP. Furthermore, two kinds of different encryption methods have been applied in our two constructions. Thus, we will analyze them re-spectively. In the scheme in Section 4.2, the data is en-crypted with the traditional encryption scheme. The data encrypted with such encryption method cannot achieve semantic security as it is inherently subject to brute-force attacks that can recover files falling into a known set. Thus, several new security notations of privacy against chosen-distribution attacks have been defined for unpredictable message. In another word, the adapted security definition guarantees that the encryptions of two unpredictable messages should be indistinguishable. Thus, the security of data in our first construction could be guaranteed under this security notion.

. The security anal-ysis for external adversaries and internal adversaries is almost identical, except the internal adversaries are pro-vided with some convergent encryption keys addition-ally. The data are encrypted with the symmetric key encryption technique, instead of the convergent encryption method. Though the sym-metric key k is randomly chosen, it is encrypted by another convergent encryption key $k_{F;p}$. Thus, we still need analyze the confidentiality of data by considering the convergent encryption. Different from the previous one, the convergent key in our construction is not de-terministic in terms of the file, which still depends on the privilege secret key stored by the private cloud server and unknown to the adversary. Therefore, if the adversary does not collude with the private cloud server, the confidentiality of our second construction is seman-tically secure for both predictable and unpredictable file. Otherwise, if they collude, then the confidentiality of file will be reduced to convergent encryption because the encryption key is deterministic

6 IMPLEMENTATION

We develop cryptographic operations of hashing and encryption with the OpenSSL library .We also implement the communication between the entities

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 02 Issue 10
October 2015

based on HTTP, using GNU Libmicrohttpd [10] and libcurl [13]. Thus, users can issue HTTP Post requests to the servers.

Our development of the **Client** givess 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; and
- FileUploadReq(FileID, File, Token) - It uploads the File Data to the Storage Server if the file is Unique and updates the File Token stored.

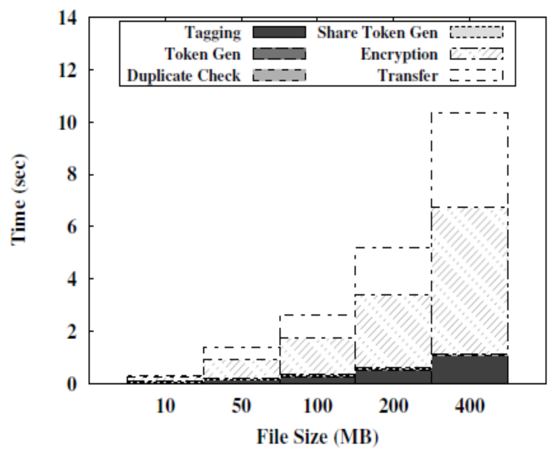

Fig. 2. Time Breakdown for Different File Size

7 EVALUATION

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 also evaluate the prototype with a real-world workload based on VM images.
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.

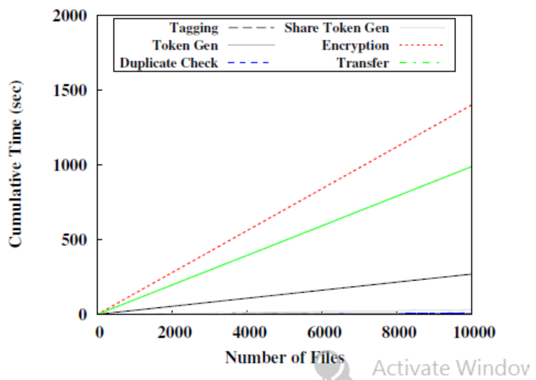

Fig. 3. Time Breakdown for Different Number of Stored Files

7.1 File Size

To evaluate the effect of file size to the time pent on different steps, we upload multiple unique files of particular file size and record the time break down. The average time of the steps from test sets of different file size are plotted in Figure 2.

7.2 Number of Stored Files

To calculate the effect of number of stored files in the system, we upload multiple 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. Despite of the possibility of a linear search, the time taken in duplicate check remains stable due to the low collision probability

7.3 Deduplication Ratio

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 02 Issue 10
October 2015

For this ratio, we prepare two unique data sets, each of which consists of 50 100MB files. First we upload the first set as an initial upload. And then for second, 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.

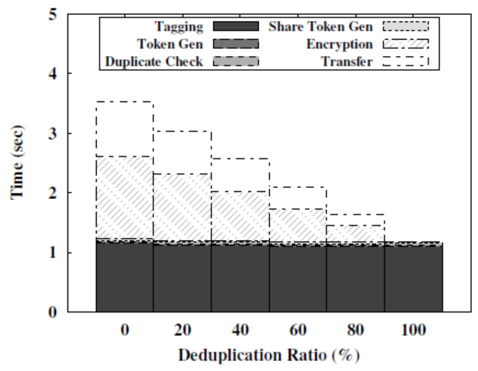

Fig. 4. Time Breakdown for Different Deduplication Ratio

7.4 Privilege Set Size

To calculate 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.
Figure shows the time taken in token generation increases linearly as more number 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 big files.

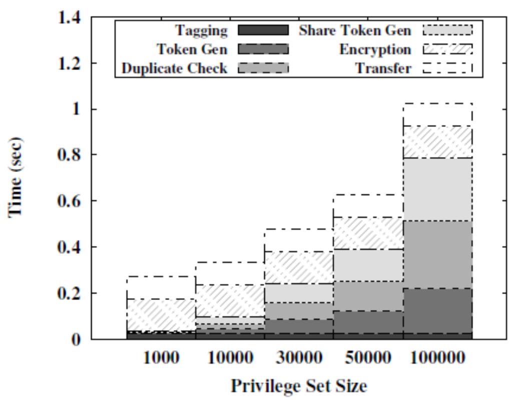

Fig. 5. Time Breakdown for Different Privilege Set Size

7.5 Real-World VM Images

Inorder to calculate the overhead introduced under read-world workload dataset, we take a dataset of weekly VM image snapshots collected over a 12-week span in a university programming course, while the same dataset is also used in the prior work We perform block-level deduplication with a fixed block size of 4KB. The initial data size of an image is 3.2GB (excluding all zero blocks). After 12 weeks, the average data size of an image increases to 4GB and the average deduplication ratio is 97.9%. Figure 6 shows that the time taken in token generation and duplicate checking increases linearly as the VM image grows in data size

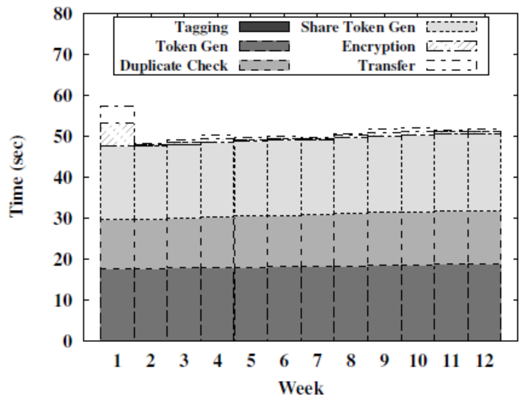

Fig. 6. Time Breakdown for the VM dataset.

7.6 Summary

To conclude , the token generation introduces only minimal overhead in the upload process and is almost 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.

8 CONCLUSION

In this paper, we proposed the notion of authorized data deduplication for protecting the information security by including differential privileges of users in the duplicate check. And also presented different new deduplication constructions supporting authorized duplicate check in hybrid cloud architecture. Security analysis describes that our systems are secure in terms of insider and attacks specified in our security model. We developed a prototype of our proposed authorized duplicate check scheme and con-duct testbed experiments on our prototype (as a proof). We showed that our authorized duplicate check scheme incurs min-imal overhead compared to convergent encryption and network transfer.

REFERENCES

[1] OpenSSL Project. http://www.openssl.org/.
[2] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In *Proc. of USENIX LISA*, 2010.
[3] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Server-aided encryption for deduplicated storage. In *USENIX Security Symposium*, 2013.
[4] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In *EUROCRYPT*, pages 296– 312, 2013.
[5] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. *J. Cryptology*, 22(1):1–61, 2009.
[6] 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.
[7] 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.
[8] 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.
[9] D. Ferraiolo and R. Kuhn. Role-based access controls. In *15th NIST-NCSC National Computer Security Conf.*, 1992.
[10] GNU Libmicrohttpd. http://www.gnu.org/software/libmicrohttpd/.
[11] 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.
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.