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

Elimination of extra copies using a Secure Authorized Deduplication framework in the Cloud Environment to Reduce Amount of Storage Space

Amboji Raju
M.Tech Student,
Department of CSE,
Sarada Institute of Technology and Science
Khammam, Telangana, India.

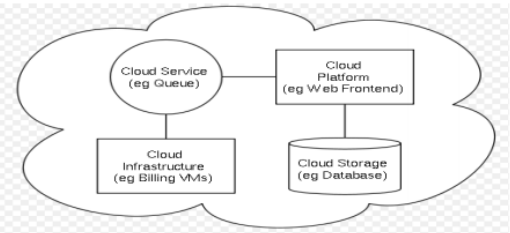
M.Prasanna
Associate Professor
Department of CSE,
Sarada Institute of Technology and Science
Khammam, Telangana, India.

Abstract:
Data deduplication is a technique for reducing the amount of storage space an organization needs to save its data. In most organizations, the storage systems contain duplicate copies of many pieces of data. For example, the same file may be saved in several different places by different users, or two or more files that aren't identical may still include much of the same data. Deduplication eliminates these extra copies by saving just one copy of the data and replacing the other copies with pointers that lead back to the original copy. Companies frequently use deduplication in backup and disaster recovery applications, but it can be used to free up space in primary storage as well. To avoid this duplication of data and to maintain the confidentiality in the cloud we using the concept of Hybrid cloud. To protect the confidentiality of sensitive data while supporting deduplication, the convergent encryption technique has been proposed to encrypt the data before outsourcing. To better protect data security, this paper makes the first attempt to formally address the problem of authorized data deduplication

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

1 Introduction

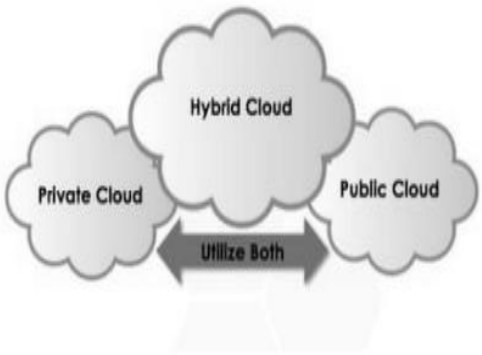
In computing, data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data. Related and somewhat synonymous terms are intelligent (data) compression and single-instance (data) storage. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. In the deduplication process, unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk. 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 A Hybrid Cloud Approach For Secure Authorized Deduplication 13 G. Kakariya and S. Rangdale control and management of private clouds. As cloud computing becomes famous, an increasing amount of data is being stored in the cloud and used by users with specified privileges, which define the access rights of the stored data.



The critical challenge of cloud storage or cloud computing is the management of the continuously increasing volume of data. Data deduplication or Single Instancing essentially refers to the elimination of redundant data. In the deduplication process, duplicate data is deleted, leaving only one copy (single instance) of the data to be stored. However, indexing

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

of all data is still retained should that data ever be required. In general the data deduplication eliminates the duplicate copies of repeating data. The data is encrypted before outsourcing it on the cloud or network. This encryption requires more time and space requirements to encode data. In case of large data storage the encryption becomes even more complex and critical. By using the data deduplication inside a hybrid cloud, the encryption will become simpler. violated. It creates the burden on the operation of cloud.



To avoid this duplication of data and to maintain the confidentiality in the cloud we using the concept of Hybrid cloud. It is a combination of public and private cloud. Hybrid cloud storage combines the advantages of scalability, reliability, rapid deployment and potential cost savings of public cloud storage with the security and full control of private cloud storage.

1.1 Contributions

In this paper, aiming at efficiently solving the problem of deduplication with differential privileges in cloud computing, we consider a hybrid cloud architecture consisting 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 perform 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 deduplication system supporting differential duplicate check is proposed under this hybrid cloud architecture where the S-CSP resides in the public cloud.

TABLE 1:Notations Used in This Paper

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

demonstrates that our system is secure in terms of the definitions specified in the proposed security model. Finally, we implement a prototype of the proposed authorized duplicate check and conduct testbed experimentsto evaluate the overhead of the prototype. Weshow that the overhead is minimal compared to the normalconvergent encryption and file upload operations.

1.2 Organization

The rest of this paper proceeds as follows. In Section 2,we briefly revisit some preliminaries of this paper. In Section 3, we propose the system model for our deduplication system. In Section 4, we propose a practical de duplication system with differential privileges in cloud computing. The security and efficiency analysis for theproposed system are respectively presented in Section 5.In Section 6, we present the implementation of our prototype, and in Section 7, we present test bed evaluation results. Finally we draw conclusion in Section 8.

2 PRELIMINARIES

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

Symmetric encryption. Symmetric encryption uses a common secret key κ to encrypt and decrypt

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

information. A symmetric encryption scheme consists of three primitive functions:KeyGenSE(1_) ! κ is the key generation algorithm that generates κ using security parameter EncSE(κ,M) ! C is the symmetric encryption algorithm that takes the secret κ and message M and then outputs the ciphertext C; and DecSE(κ,C) ! M is the symmetric decryption algorithm that takes the secret κ 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 This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TPDS.2014.2318320, IEEE Transactions on Parallel and Distributed Systems 3property [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. Both the encrypted data copy and its corresponding tag will be stored on the server side. Formally, a convergent encryption scheme can be defined with four primitive functions:• 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 atag T(M).

Proof of ownership. The notion of proof of ownership (PoW) [11] enables users to prove their ownership of data copies to the storage server. Specifically, PoW is implemented as an interactive algorithm (denoted by PoW) run by a prover (i.e., user) and a verifier (i.e., storage server). The verifier derives a short value $\phi(M)$ from a data copy M. To prove the ownership of the data copy M, the prover needs to send ϕ' to the verifier such that $\phi' = \phi(M)$. The formal security definition for PoW roughly follows the threat model in a content distribution network, where an attacker does not know the entire file, but has accomplices who have the file. The accomplices follow the "bounded retrieval model", such that they can help the attacker obtain the file, subject to the constraint that they must send fewer bits than the initial min-entropy of the file to the attacker [11].

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. The verifier performs the verification with input of public information *pkU* related to *skU*. At the conclusion of the protocol, the verifier outputs either accept or reject to denote whether the proof is passed or not. There are many efficient identification protocols in literature, including certificate-based, identity-based identification etc. [5], [6].

User

1. Public Cloud
2. Private Cloud
3. Upload/Download Request
4. Results
5. Encrypted Files

Privilege Keys
Token Request
File Token

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

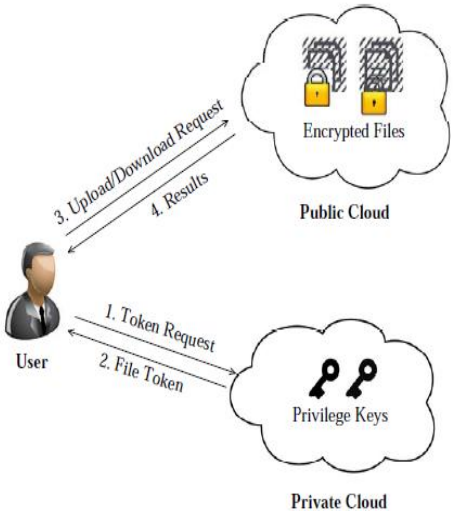


Fig. 1. Architecture for Authorized Deduplication

3 System Model

3.1 Hybrid Architecture for Secure Deduplication

At a high level, our setting of interest is an enterprise network, 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. In this setting, deduplication can be frequently used in these settings for data backup and disaster recovery applications while greatly reducing storage space. Such systems are widespread and are often more suitable to user file 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.

3.2 Design Goals In this paper, we address the problem of privacypreserving deduplication in cloud computing and propose 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. The security requirements considered in this paper lie in two folds, including the security of file token and security of data files. For the security of file token, two aspects are defined as unforgeability and in distinguish ability of file token.

The details are given below. *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 users are not allowed to collude with the public cloud server to break the unforgeability This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2318320, IEEE Transactions on Parallel and Distributed Systems requires that any user without querying the private cloud server for some file token, he cannot get any useful information from the token, which includes the file information or the privilege information.

4 Secure Deduplication Systems Main Idea.

To support authorized de duplication, 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.

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

4.1 A First Attempt Before introducing our construction of differentialdeduplication, 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

5 Security Analysis Our system is 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. Some basic tools have been used to construct the secure deduplication, which are assumed to be secure. 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 security analysis.

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 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) in distinguishability of duplicatecheck 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

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

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. Next, we will give detailed security analysis for scheme in Section 4.2 based on the above definitions. This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2318320, IEEE Transactions on Parallel and Distributed Systems 8

Unforgeability of duplicate-check token Assume a user with privilege p could forge a new duplicate-check token $\phi' F;p'$ for any p' that does not match p. If it is a valid token, then it should be calculated as $\phi' F;p' = H1(H(F), kp')$. Recall that kp' is a secret key kept by the private cloud server and $H1(H(F), kp')$ is a valid message authentication code. Thus, without kp', the adversary cannot forge and output a new valid one for any file F. • For any user with privilege p, to output a new duplicate-check token $\phi' F;p$, it also requires the knowledge of kp. Otherwise, the adversary could break the security of message authentication code.

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 unknown 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 skp.

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 respectively. In the scheme in Section 4.2, the data is encrypted with the traditional encryption scheme. The data encrypted with such encryption method cannot achieve semantic security as it is inherently subject to bruteforce 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. We discuss the confidentiality of data in our further enhanced construction in Section 4.3. The security analysis for external adversaries and internal adversaries is almost identical, except the internal adversaries are provided with some convergent encryption keys additionally. However, these convergent encryption keys have no security impact on the data confidentiality because these convergent encryption keys are computed with different privileges. Recall that the data are encrypted with the symmetric key encryption technique, instead of the convergent encryption method. Though the symmetric key k is randomly chosen, it is encrypted by another convergent encryption key $kF;p$.

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 deterministic 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 semantically 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.

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

6 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 deduplicates files. We implement cryptographic operations of hashing and encryption with the OpenSSL library [1].We also implement the communication between the entities based on HTTP, using GNU Libmicrohttpd [10] and libcurl [13]. Thus, users can issue HTTP Post requests to the servers. 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; 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. 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; and This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2318320, IEEE Transactions on Parallel and Distributed Systems

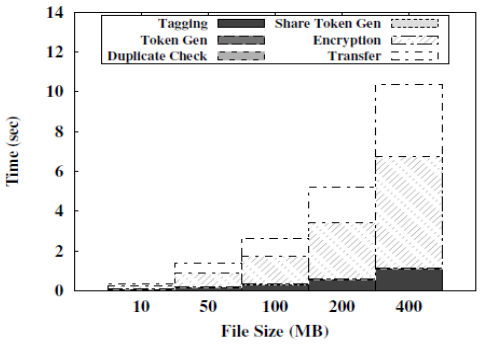


Fig. 2. Time Breakdown for Different File Size

ShareTokenGen (Tag, {Priv.}) - It generates the share token with the corresponding privilege keys of the sharing privilege set with HMAC-SHA-1algorithm. Our implementation of the **Storage Server** provides deduplication and data storage with following handlers and maintains a map between existing files and associated token with Hash Map. DupCheck(Token) - It searches the File to Token Map for Duplicate; and FileStore(FileID, File, Token) - It stores the File on Disk and updates the Mapping.

7 Evaluation

We conduct testbed evaluation on our prototype. 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 conduct the experiments with three machines equipped with an Intel Core-2-Quad 2.66GHz Quad Core CPU, 4GB RAM and installed with Ubuntu 12.04 32- Bit Operation System. The machines are connected with 1Gbps Ethernet network. 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

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

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 of particular file size and record the time break down.

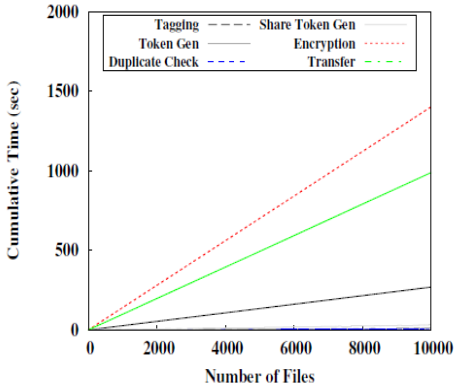


Fig. 3. Time Breakdown for Different Number of Stored Files

Using the unique files enables us to evaluate the worst-case scenario whre 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. 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 10MB to 400MB, the overhead of the proposed authorization steps decreases from 14.9% to 0.483%.

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. 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

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. As uploading and encryption would be skipped in case of duplicate files, the time spent on both of them decreases with increasing

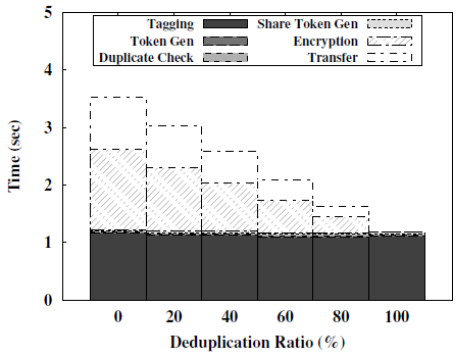


Fig. 4. Time Breakdown for Different Deduplication Ratio

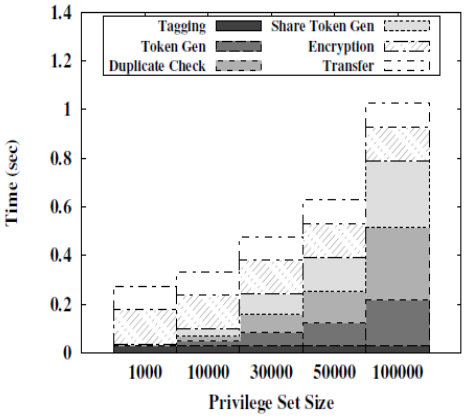


Fig. 5. Time Breakdown for Different Privilege Set Size

Fig. 5. Time Breakdown for Different Privilege Set Size deduplication ratio. The time spent on duplicate check also decreases as the searching would be ended when duplicate is found. Total time spent on uploading the file with deduplication ratio at 100% is only 33.5% with unique files.

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

7.4 Privilege 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 Real-World VM Images

To evaluate the overhead introduced under read-world workload dataset, we consider a dataset of weekly VM

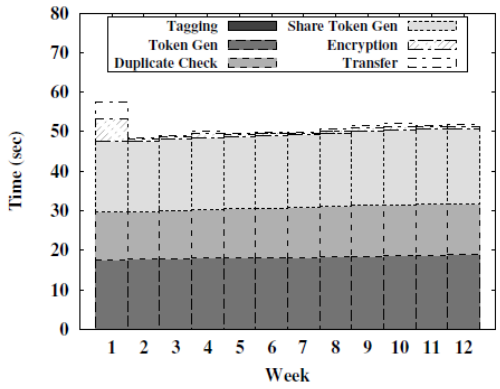


Fig. 6. Time Breakdown for the VM dataset.

image snapshots collected over a 12-week span in a university programming course, while the same dataset is also used in the prior work [14]. We perform blocklevel deduplication with a fixed block size of 4KB. The initial data size of an image is 3.2GB (excluding all zero blocks).

8 Related Work

Secure Deduplication. With the advent of cloudcomputing, secure data deduplication has attracted much attention recently from research community. Yuan et al. [24] proposed a deduplication system in the cloud storage to reduce the storage size of the tags for integrity check. To enhance the security of deduplication and protect the data confidentiality, Bellare et al. [3] showed how to protect the data confidentiality by transforming the predicatable message This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2318320, IEEE Transactions on Parallel and Distributed Systems 11 into unpredicatable message. In their system, another third party called key server is introduced to generate the file tag for duplicate check. Stanek et al. [20] presented a novel encryption scheme that provides differential security for popular data and unpopular data. For popular data that are not particularly sensitive, the traditional conventional encryption is performed. Another two-layered encryption scheme with stronger security while supporting deduplication is proposed for unpopular data. In this way, they achieved better tradeoff between the efficiency and security of the outsourced data. Li et al. [12] addressed the keymanagement issue in block-level deduplication by distributing these keys across multiple servers after encrypting the files.

Convergent Encryption. Convergent encryption [8] ensures data privacy in deduplication. Bellare et al. [4] formalized this primitive as message-locked encryption, and explored its application in space-efficient secure outsourced storage. Xu et al. [23] also addressed the problem and showed a secure convergent encryption for efficient encryption, without considering issues of the key-management and block-level deduplication. There are also several implementations of convergent implementations of different convergent encryption variants for secure deduplication (e.g., [2], [18], [21], [22]). It is known that some commercial cloud storage providers, such as Bitcasa, also deploy convergent encryption.

Proof of ownership. Halevi et al. [11] proposed the notion of "proofs of ownership" (PoW) for deduplication systems, such that a client can efficiently prove to the cloud storage server that he/she owns a file without uploading the file itself. Several PoW constructions based on the Merkle-Hash Tree are

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

proposed [11] to enable client-side deduplication, which include the bounded leakage setting. Pietro and Sorniotti [16] proposed another efficient PoW scheme by choosing the projection of a file onto some randomly selected bit-positions as the file proof. Note that all the above schemes do not consider data privacy. Recently, Ng et al. [15] extended PoW for encrypted files, but they do not address how to minimize the key management overhead.

Twin Clouds Architecture. Recently, Bugiel et al. [7] provided an architecture consisting of twin clouds for secure outsourcing of data and arbitrary computations to an untrusted commodity cloud. Zhang et al. [25] also presented the hybrid cloud techniques to support privacy-aware data-intensive computing. In our work, we consider to address the authorized deduplication problem over data in public cloud. The security model of our systems is similar to those related work, where the private cloud is assume to be honest but curious.

9 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.

10.References

[1] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Serveraided encryption for deduplicated storage. In USENIX Security Symposium, 2013. [2] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In Proc. of USENIX LISA, 2010.

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

[4] 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.

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

[6] C. Ng and P. Lee. Revdedup: A reverse de-duplication storage system optimized for reads to latest backups. In Proc. of APSYS, Apr 2013.

[7] C.-K Huang, L.-F Chien, and Y.-J Oyang, "Relevant Term Suggestion in Interactive Web Search Based on Contextual Information in Query Session Logs," J. Am. Soc. for Information science and Technology, vol. 54, no. 7, pp. 638-649, 2003.

[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 1217889). multiplier follows steps such as delete non require bits, reduce the level, truncation, round up result