A Hybrid Cloud Approach for Secure Authorized Deduplication

Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P. C. Lee, Wenjing Lou

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 to reduce the amount of storage space and save bandwidth. 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. Different from traditional deduplication systems, the differential privileges of users are further considered in duplicate check besides the data itself. We also present several new deduplication constructions supporting authorized duplicate check in a hybrid cloud architecture. Security analysis demonstrates that our scheme is secure in terms of the definitions specified in the proposed security model. As a proof of concept, we implement a prototype of our proposed authorized duplicate check scheme and conduct testbed experiments using our prototype. We show that our proposed authorized duplicate check scheme incurs minimal overhead compared to normal operations.

Index Terms—Deduplication, authorized duplicate check, confidentiality, hybrid cloud

1 Introduction

Cloud computing provides seemingly unlimited "virtualized" resources to users as services across the whole Internet, while hiding platform and implementation details. Today's cloud service providers offer both highly available storage and massively parallel computing resources at relatively low costs. As cloud computing becomes prevalent, an increasing amount of data is being stored in the cloud and shared by users with specified *privileges*, which define the access rights of the stored data. One critical challenge of cloud storage services is the management of the ever-increasing volume of data.

To make data management scalable in cloud computing, deduplication [17] has been a well-known technique and has attracted more and more attention recently. Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data in storage. The 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. Instead of keeping multiple data copies 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 place at either the file level or the 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 data deduplication brings a lot of benefits, security and privacy concerns arise as users' sensitive data are susceptible to both insider and outsider attacks. Traditional encryption, while providing data confidentiality, is incompatible with data deduplication. Specifically, traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different ciphertexts, making deduplication impossible. Convergent encryption [8] 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. 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, identical 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

- *J. Li is with the School of Computer Science, Guangzhou University, P.R. China and Department of Computer Science, Virginia Polytechnic Institute and State University, USA (Email: lijin@gzhu.edu.cn)*
- *Xiaofeng Chen is with the State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, P.R. China and Department of Computer Science, Virginia Polytechnic Institute and State University, USA (Email: xfchen@xidian.edu.cn)*
- *Y. Li and P. Lee are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong (emails: {liyk,pclee}@cse.cuhk.edu.hk).*
- *W. Lou is with the Department of Computer Science, Virginia Polytechnic Institute and State University, USA (Email: wjlou@vt.edu)*

the file.

However, previous deduplication systems cannot support *differential authorization duplicate check*, which is important in many applications. In such an authorized deduplication system, each user is issued a set of privileges 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 consideration, 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 providing 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 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. 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 corresponding privileges cannot perform the duplicate check. Furthermore, 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 experiments to evaluate the overhead of the prototype. We show that the overhead is minimal compared to the normal convergent 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 deduplication system with differential privileges in cloud computing. The security and efficiency analysis for the proposed system are respectively presented in Section 5. In Section 6, we present the implementation of our prototype, and in Section 7, we present testbed 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 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 information. A symmetric encryption scheme consists of three primitive functions:

- $\mathsf{KeyGen}_{\mathsf{SE}}(1^\lambda) \rightarrow \kappa$ is the key generation algorithm that generates κ using security parameter 1^λ;
- $\mathsf{Enc}_{\mathsf{SE}}(\kappa, M) \rightarrow C$ is the symmetric encryption algorithm that takes the secret κ and message M and then outputs the ciphertext C; and
- $\mathsf{Dec}_{\mathsf{SE}}(\kappa, C) \rightarrow M$ 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

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. 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:

- $\mathsf{KeyGen}_{\mathsf{CE}}(M) \rightarrow K$ is the key generation algorithm that maps a data copy M to a convergent key K;
- $\mathsf{Enc}_{\mathsf{CE}}(K, M) \rightarrow C$ 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;
- $\mathsf{Dec}_{\mathsf{CE}}(K, C) \rightarrow M$ 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
- $\mathsf{TagGen}(M) \rightarrow T(M)$ is the tag generation algorithm that maps the original data copy M and outputs a tag $T(M)$.

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 Π 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 sk_U 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 pk_U related to sk_U. 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].

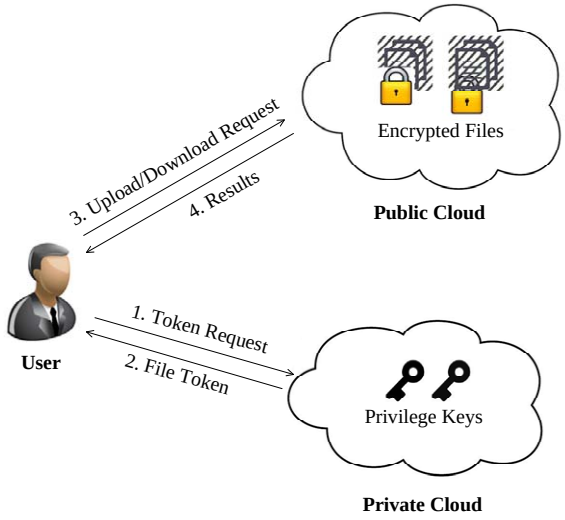


Fig. 1. Architecture for Authorized Deduplication

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. 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 deduplicable 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 deduplication 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 data storage service in public 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 deduplication 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 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 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 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. 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.

Notice that this is a novel architecture for data deduplication in cloud computing, which consists of a twin clouds (i.e., the public cloud and the private cloud). Actually, this hybrid cloud setting has attracted more and more attention recently. For example, an enterprise might use a public cloud service, such as Amazon S3, for archived data, but continue to maintain in-house storage for operational customer data. Alternatively, the trusted private cloud could be a cluster of virtualized cryptographic co-processors, which are offered as a service by a third party and provide the necessary hardware-based security features to implement a remote execution environment trusted by the users.

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 sensitive 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 information 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 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 indistinguishability 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

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.
- *Indistinguishability of file token/duplicate-check token.* It 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.
- *Data Confidentiality.* Unauthorized users without appropriate 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 definition of data confidentiality based on convergent encryption, a higher level confidentiality is defined and achieved.

4 Secure Deduplication Systems

Main Idea. To 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 k_p will be bounded with a privilege p to generate a file token. Let $\phi'_{F,p} = \mathsf{TagGen}(F, k_p)$ denote the token of F that is only allowed to access by user with privilege p. In another word, the token $\phi'_{F,p}$ 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 $H(F, k_p)$, where $H(\cdot)$ 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 $\mathsf{TagGen}(F, k_p)$ 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 $\mathcal{P} = \{p_1, \ldots, p_s\}$. For each privilege p in $\mathcal{P}$, 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 $\{k_{p_i}\}_{p_i \in P_U}$.

File Uploading. Suppose that a data owner U with privilege set P_U wants to upload and share a file $\mathcal{F}$ with users who have the privilege set $P_F = \{p_j\}$. The user computes and sends S-CSP the file token $\phi'_{F,p} = \mathsf{TagGen}(F, k_p)$ for all $p \in P_F$.

- 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 $C_F = \mathsf{Enc}_{\mathsf{CE}}(k_F, F)$ with the convergent key $k_F = \mathsf{KeyGen}_{\mathsf{CE}}(F)$ and uploads $(C_F, \{\phi'_{F,p}\})$ 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.

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

- First, each user will be issued private keys $\{k_{p_i}\}_{p_i \in P_U}$ for their corresponding privileges, denoted by P_U in our above construction. These private keys $\{k_{p_i}\}_{p_i \in P_U}$ 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 P_F. To compute these file tokens, the user needs to know the private keys for P_F, which means P_F could only be chosen from P_U. 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 P_{U1} may collude with another user with privilege set P_{U2} to get a privilege set P^*=$P_{U1} \cup P_{U2}$.
- 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 Our Proposed System Description

To solve the problems of the construction in Section 4.1, we propose another advanced deduplication system supporting authorized duplicate check. In this new deduplication 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. 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 $\mathbb{R} = \{((p, p')\}$ as follows. Given two privileges p and p', we say that p matches p' if and only if $\mathbb{R}(p, p') = 1$. 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 privilege. 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 $\mathcal{P}$ is defined as in Section 4.1. A symmetric key k_{p_i} for each $p_i \in \mathcal{P}$ will be selected and the set of keys $\{k_{p_i}\}_{p_i \in \mathcal{P}}$ will be sent to the private cloud. An identification protocol $\Pi = (\mathsf{Proof}, \mathsf{Verify})$ 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 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 $\perp$.

File Uploading. Suppose that a data owner wants to upload and share a file $\mathcal{F}$ with users whose privilege belongs to the set $P_F = \{p_j\}$. The data owner needs interact with the private cloud before performing duplicate check with the S-CSP. More precisely, the data owner performs an identification to prove its identity with private key sk_U. If it is passed, the private cloud server will find the corresponding privileges P_U of the user from its stored table list. The user computes and sends the file tag $\phi_F = \mathsf{TagGen}(F)$ to the private cloud server, who will return $\{\phi'_{F,p_\tau} = \mathsf{TagGen}(\phi_F, k_{p_\tau})\}$ back to the user for all p_τ satisfying $\mathbb{R}(p, p_\tau) = 1$ and $p \in P_U$. Then, the user will interact and send the file token $\{\phi'_{F,p_\tau}\}$ 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 $\{\phi'_{F,p_\tau}\}$, pk_U and a time stamp. The user sends the privilege set $P_F = \{p_j\}$ 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 $\{\phi'_{F,p_\tau} = \mathsf{TagGen}(\phi_F, k_{p_\tau})\}$ for all p_τ satisfying $\mathbb{R}(p, p_\tau) = 1$ for each $p \in P_F$-P_U, 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 P_F 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 $\{\phi'_{F,p_\tau}\}$, pk_U and a time stamp. The user sends the privilege set $P_F = \{p_j\}$ 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 $\{\phi'_{F,p_\tau} = \mathsf{TagGen}(\phi_F, k_{p_\tau})\}$ for all p_τ satisfying $\mathbb{R}(p, p_\tau) = 1$ and $p \in P_F$. Finally, the user computes the encrypted file $C_F = \mathsf{Enc}_{\mathsf{CE}}(k_F, F)$ with the convergent key $k_F = \mathsf{KeyGen}_{\mathsf{CE}}(F)$ and uploads $\{C_F, \{\phi'_{F,p_\tau}\}\}$ with privilege P_F.

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 k_F after receiving the encrypted data from the S-CSP.

4.3 Further Enhancement

Though the above solution supports the differential privilege duplicate, it is inherently subject to brute-force attacks launched by the public cloud server, which can recover files falling into a known set. More specifically, knowing that the target file space underlying a given ciphertext C is drawn from a message space $S = \{F_1, \cdots, F_n\}$ of size n, the public cloud server can recover F after at most n off-line encryptions. That is, for each $i = 1, \cdots, n$, it simply encrypts F_i to get a ciphertext denoted by C_i. If $C = C_i$, it means that the underlying file is F_i. Security is thus only possible when such a message is unpredictable. This traditional

convergent encryption will be insecure for predictable file.

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 $k_F = H(F)$. To avoid the deterministic key generation, the encryption key k_F for file F in our system will be generated with the aid of the private key cloud server with privilege key k_p. The encryption key can be viewed as the form of $k_{F,p} = H_0(H(F), k_p) \bigoplus H_2(F)$, 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 $k_{F,p}$. 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. For S-CSP, if the file is unpredicatable, then it is semantically secure too. The details of the scheme, which has been instantiated with hash functions for simplicity, are described below.

System Setup. The privilege universe $\mathcal{P}$ and the symmetric key k_{p_i} for each $p_i \in \mathcal{P}$ will be selected for the private cloud as above. An identification protocol $\Pi = (\mathsf{Proof}, \mathsf{Verify})$ is also defined. The proof of ownership POW is instantiated by hash functions H, H_0, H_1 and H_2, which will be shown as follows. The private cloud server maintains a table which stores each user's identity and its corresponding privilege.

File Uploading. Suppose that a data owner with privilege p wants to upload and share a file $\mathcal{F}$ with users whose privilege belongs to the set $P = \{p_j\}$. The data owner performs the identification and sends $H(F)$ to the private cloud server. Two file tag sets $\{\phi_{F,p_\tau} = H_0(H(F), k_{p_\tau})\}$ and $\{\phi'_{F,p_\tau} = H_1(H(F), k_{p_\tau})\}$ for all p_τ satisfying $\mathbb{R}(p, p_\tau) = 1$ and $p \in P_U$ will be sent back to the user if the identification passes. After receiving the tag $\{\phi_{F,p_\tau}\}$, and $\{\phi'_{F,p_\tau}\}$, the user will interact and send these two tag sets 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 also passed, the user will be provided a pointer for the file. Otherwise, if no duplicate is found, a proof from the S-CSP will be returned, which could be a signature. The user sends the privilege set $P = \{p_j\}$ as well as the proof to the private cloud server for file upload request. Upon receiving the request, the private cloud server verifies the signature first. If it is passed, the private cloud server will compute $\phi_{F,p_j} = H_0(H(F), k_{p_j})$ and $\phi'_{F,p_j} = H_1(H(F), k_{p_j})$ for each p_j satisfying $\mathbb{R}(p, p_j) = 1$ and $p \in P_F$, which will be returned to the user. Finally, the user computes the encryption $C_F = \mathsf{Enc}_{\mathsf{SE}}(k, F)$, where k is random key, which will be encrypted into ciphertext C_{k,p_j} with each key in $\{k_{F,p_j} = \phi_{F,p_j} \bigoplus H_2(F)\}$ using a symmetric encryption algorithm. Finally, the user uploads $\{\phi'_{F,p_j}, C_F, C_{k,p_j}\}$.

File Retrieving. The procedure of file retrieving is similar to the construction in Section 4.2. Suppose a user wants to download a file F. The user first uses his key k_{F,p_j} to decrypt C_{k,p_j} and obtain k. Then the user uses k to recover the original file F.

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) *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.

Next, we will give detailed security analysis for scheme in Section 4.2 based on the above definitions.

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 $\phi'_{F,p'} = H_1(H(F), k_{p'})$. Recall that $k_{p'}$ is a secret key kept by the private cloud server and $H_1(H(F), k_{p'})$ is a valid message authentication code. Thus, without $k_{p'}$, 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 k_p. 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 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 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 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.

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 $k_{F,p}$. 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 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.

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

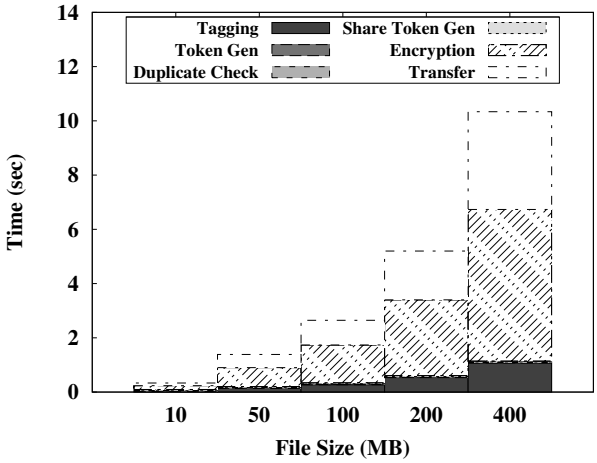


Fig. 2. Time Breakdown for Different File Size

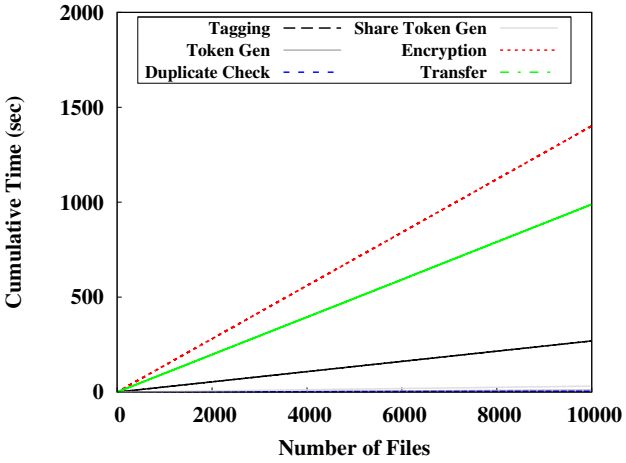


Fig. 3. Time Breakdown for Different Number of Stored Files

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

Our implementation of the **Storage Server** provides 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 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. 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

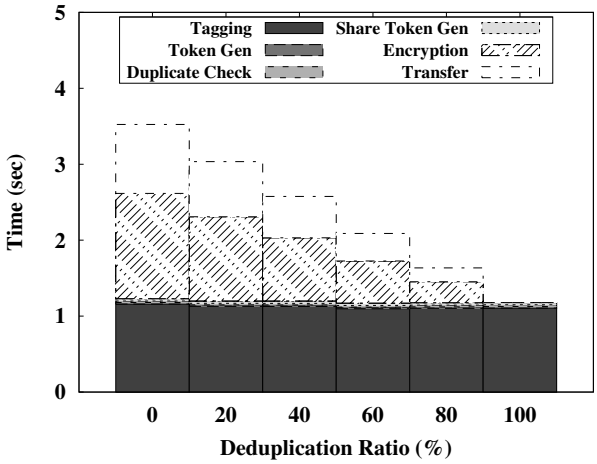


Fig. 4. Time Breakdown for Different Deduplication Ratio

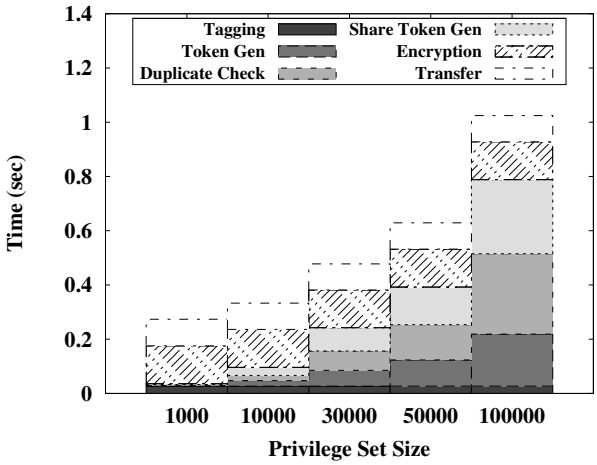


Fig. 5. Time Breakdown for Different Privilege Set Size

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.

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

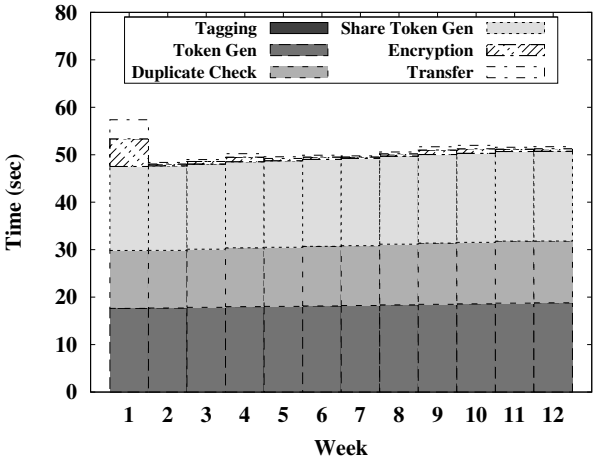


Fig. 6. Time Breakdown for the VM dataset.

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 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%. For privacy, we only collected cryptographic hashes on 4KB fixed-size blocks; in other words, the tagging phase is done beforehand. Here, we randomly pick 10 VM image series to form the dataset. Figure 6 shows that the time taken in token generation and duplicate checking increases linearly as the VM image grows in data size. The time taken in encryption and data transfer is low because of the high deduplication ratio. Time taken for the first week is the highest as the initial upload contains more unique data. Overall, the results are consistent with the prior experiments that use synthetic workloads.

7.6 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.

8 Related Work

Secure Deduplication. With the advent of cloud computing, 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

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 key-management 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 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.

Acknowledgements

This work was supported by National Natural Science Foundation of China (NO.61100224 and NO.61272455), GRF CUHK 413813 from the Research Grant Council of Hong Kong, Distinguished Young Scholars Fund of Department of Education(No. Yq2013126), Guangdong Province, China. Besides, Lou's work is supported by US National Science Foundation under grant (CNS-1217889).

References

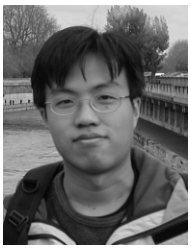
[1] OpenSSL Project. http://www.openssl.org/.
[2] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In *Proc. of USENIX LISA*, 2010.
[3] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Server-aided encryption for deduplicated storage. In *USENIX Security Symposium*, 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 concurrent 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.
[12] 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.
[13] libcurl. http://curl.haxx.se/libcurl/.
[14] C. Ng and P. Lee. Revdedup: A reverse deduplication storage system optimized for reads to latest backups. In *Proc. of APSYS*, Apr 2013.

[15] W. K. Ng, Y. Wen, and H. Zhu. Private data deduplication protocols in cloud storage. In S. Ossowski and P. Lecca, editors, *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 441–446. ACM, 2012.
[16] R. D. Pietro and A. Sorniotti. Boosting efficiency and security in proof of ownership for deduplication. In H. Y. Youm and Y. Won, editors, *ACM Symposium on Information, Computer and Communications Security*, pages 81–82. ACM, 2012.
[17] S. Quinlan and S. Dorward. Venti: a new approach to archival storage. In *Proc. USENIX FAST*, Jan 2002.
[18] A. Rahumed, H. C. H. Chen, Y. Tang, P. P. C. Lee, and J. C. S. Lui. A secure cloud backup system with assured deletion and version control. In *3rd International Workshop on Security in Cloud Computing*, 2011.
[19] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29:38–47, Feb 1996.
[20] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl. A secure data deduplication scheme for cloud storage. In *Technical Report*, 2013.
[21] M. W. Storer, K. Greenan, D. D. E. Long, and E. L. Miller. Secure data deduplication. In *Proc. of StorageSS*, 2008.
[22] Z. Wilcox-O'Hearn and B. Warner. Tahoe: the least-authority filesystem. In *Proc. of ACM StorageSS*, 2008.
[23] J. Xu, E.-C. Chang, and J. Zhou. Weak leakage-resilient client-side deduplication of encrypted data in cloud storage. In *ASIACCS*, pages 195–206, 2013.
[24] J. Yuan and S. Yu. Secure and constant cost public cloud storage auditing with deduplication. *IACR Cryptology ePrint Archive*, 2013:149, 2013.
[25] K. Zhang, X. Zhou, Y. Chen, X. Wang, and Y. Ruan. Sedic: privacy-aware data intensive computing on hybrid clouds. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS'11, pages 515–526, New York, NY, USA, 2011. ACM.

Jin Li received his B.S. (2002) in Mathematics from Southwest University. He got his Ph.D degree in information security from Sun Yat-sen University at 2007. Currently, he works at Guangzhou University as a professor. He has been selected as one of science and technology new star in Guangdong province. His research interests include Applied Cryptography and Security in Cloud Computing. He has published over 70 research papers in refereed international conferences and journals and has served as the program chair or program committee member in many international conferences.

Yan Kit Li received the BEng. Degree in Computer Engineering from the Chinese University of Hong Kong in 2012. Currently he is an M.Phil. student of the Department of Computer Science and Engineering at the same school. His research interests including deduplication and distributed storage.

Xiaofeng Chen received his B.S. and M.S. on Mathematics in Northwest University, China. He got his Ph.D degree in Cryptography from Xidian University at 2003. Currently, he works at Xidian University as a professor. His research interests include applied cryptography and cloud computing security. He has published over 100 research papers in refereed international conferences and journals. He has served as the program/general chair or program committee member in over 20 international conferences.



Patrick P. C. Lee received the B.Eng. degree (first-class honors) in Information Engineering from the Chinese University of Hong Kong in 2001, the M.Phil. degree in Computer Science and Engineering from the Chinese University of Hong Kong in 2003, and the Ph.D. degree in Computer Science from Columbia University in 2008. He is now an assistant professor of the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests are in various applied/systems topics including cloud computing and storage, distributed systems and networks, operating systems, and security/resilience.

Wenjing Lou received a B.S. and an M.S. in Computer Science and Engineering at Xi'an Jiaotong University in China, an M.A.Sc. in Computer Communications at the Nanyang Technological University in Singapore, and a Ph.D. in Electrical and Computer Engineering at the University of Florida. She is now an associate professor in the Computer Science department at Virginia Polytechnic Institute and State University.