Authorized Client-side Deduplication Using Access Policy-based Convergent Encryption

Taek-Young Youn[1], Ku-Young Chang[1], Kyung Hyune Rhee[2], Sang Uk Shin[2]

[1] Electronics and Telecommunications Research Institute (ETRI), Korea
[2] Dept. of IT Convergence and Application Engineering, Pukyong National University, Korea
{taekyoung, jang1090}@etri.re.kr, {khrhee, shinsu}@pknu.ac.kr

Abstract

This paper proposes the method to provide efficient use of cloud storage while supporting secure data sharing in the cloud. In order to provide authorized deduplication, we use the convergent encryption scheme and apply an access privilege to generate a convergent key. Because of this, the user without proper privileges will not be able to generate the convergent key and thus cannot access the shared data. To verify the ownership of the file in the client-side deduplication procedure, we also propose a new proofs of ownership protocol based on an existing Merkle Tree-based protocol. Our scheme provides an adequate trade-off between security and storage space efficiency. By executing the deduplication for users with the same privilege, the effect of deduplication can be reduced. However, in view of the data sharing, our approach has as advantage in the sense that only authorized users can access the files encrypted based on privileges allowed to the users. The proposed scheme is very suitable for the hybrid cloud model considering both the data security and the storage efficiency.

Keywords: Client-side deduplication, Convergent encryption, Proofs of ownership, Access privilege

1 Introduction

With the rapid development of cloud computing, many companies and individual users are outsourcing their confidential data to cloud storage service providers such as Dropbox, Google drive. These service providers reduce the burden of data storage and maintenance at the client side while providing high-quality data storage and computing services in a ubiquitous manner. As a result, the amount of data stored in the storage of the cloud storage provider (CSP) is increasing rapidly, and in particular, has more rapidly increased in the era of big data. According to the analysis of the IDC, global data volume is expected to reach 40 trillion GB in 2020 [1]. Therefore, one of the key issues of CSP is that how efficiently manage the data that increases continuously.

Deduplication is one important approach to deal with this issue. Deduplication is a special data compression technique to remove redundant copy of the repeated data [2]. This technique is used to improve the utilization ratio of the storage and can be applied to the network data transfer in order to reduce the amount of data that should be transmitted. CSP performs a save operation on the first uploaded data files. For the subsequent upload request for the same file, it ensures that each data file is stored one copy only at the server by assigning a link to the first uploaded data copy to the requester. Deduplication can be used to effectively reduce the data storage space and the communication overhead.

Deduplication technology can be classified on the client side deduplication and server-side deduplication technology. Because the client-side deduplication technology has the advantage from the viewpoint of efficient use of bandwidth, many studies on the client-side deduplication is being carried out. Despite the many advantages of the deduplication technology, deduplication technology on critical data has caused some new security problems. In particular, the confidentiality protection of outsourcing data is a very important issue. To deal with this issue, it is possible to consider the encryption prior to outsourcing data to the cloud. If a conventional encryption is used, different ciphertexts will be generated for the same plaintext users use different key for encryption. In this case, it becomes impossible to perform the deduplication. To solve this problem, Douceur et al. [3] proposed the convergent encryption technique using the hash value of the plaintext as the encryption key (where, encryption scheme $E()$ is deterministic algorithm and a convergent key K depends only on the input data file F). However, convergent encryption techniques are vulnerable to offline brute-force attack since the message can be predicted in many cases as indicated in [4]. By an attacker to execute encrypt all possible plaintext in offline phase, it is possible to determine the corresponding plaintext information. To solve this

*Corresponding Author: Sang Uk Shin; E-mail: shinsu@pknu.ac.kr
DOI: 10.3966/160792642018081904025

problem, several studies have been conducted [5-7].

Meanwhile, cloud computing system may be used to provide a data sharing service, which provides numerous advantages for the user. Currently, many information technology (IT) organizations are possible to obtain high productivity benefits through data sharing, and it can result in savings of time and money. Typically, healthcare service provider in the healthcare environment saves and shares the medical records through the cloud, which makes it possible to eliminate geographic dependencies between health care providers and patients. However, the sharing of data via the cloud has threats on the privacy and security, and thus solving this problem is a top priority issue for the dissemination of the cloud. For secure data sharing in the cloud, the owner of the data must be able to specify a user or a group of users which is allowed access to data. Entities other than the owner and the user with the right should not be permitted to access to the data [8]. The simplest solution for this is to encrypt their data before the owner uploads the data to the cloud. Thereafter, if the owner of the data may want to share, it transmits the key to the members of the group. However, this approach is very inefficient and many complex problems occur in the delivery of the key. Also, in terms of cloud storage, there is a problem that deduplication for efficient use of storage space does not support.

Therefore, in this paper, we propose the method to provide efficient use of cloud storage while supporting secure data sharing in the cloud. To support deduplication, the proposed scheme uses the convergent encryption techniques, and applies an access privilege to generate a convergent key. Due to this, the user without proper privileges will not be able to generate a convergent key and thus cannot access the shared data. We analyze the security and performance of the proposed scheme, and compare the features with the existing authorized deduplication scheme. One of notable merits of the proposed method is that it provides an adequate trade-off between security and storage space efficiency, and is very suitable for the hybrid cloud model considering both the data security and the storage efficiency.

Note that the preliminary version of this paper was presented at MobiSec2016 [9]. The main improvement of this paper is that we propose an improved scheme which can support stronger security than the scheme in [9]. Especially, the scheme [9] is secure only if users do not collude to break the security of the scheme. In this paper, we analyze the security of the authorized convergent key distribution protocol in [9] with respect to the collusion attacks where some users collude to extend their access privilege without the help of the authorization server. Then, we propose an improved protocol based on blind BLS signature, which can support stronger security against the collusion attacks. We also provide a detailed analysis of security and complexity.

The rest of the paper organized as follows. In Section 2, we briefly describe related works on the convergent encryption and the proofs of ownership (PoW). In Section 3, we propose an authorized deduplication using RSA blind signature and an improved Merkle Tree-based PoW. An improved authorized deduplication based on blind BLS signature is proposed in Section 4, and also the security and efficiency analysis are presented. Finally, we draw conclusion in Section 5.

2 Related Works

2.1 Convergent Encryption

Convergent encryption scheme proposed by Douceur et al. [3] encrypts a file F using a symmetric key encryption algorithm $E()$ with the key which is generated by hashing the file F. $C=E(H(F), F)$. Therefore, the same plaintext becomes the same encryption, allowing duplication process. However, convergent encryption scheme is very vulnerable to a dictionary attack which is a brute force attack [4]. DupLESS scheme used a key server (KS) to prevent offline brute force dictionary attack [5]. It uses RSA blind signature based oblivious PRF (pseudo random function) protocol between KS and a client. And DupLESS applied rate-limiting scheme in order to prevent a brute force attack which one client sends multiple request messages to KS.

Duan [6] proposed a scheme that eliminates KS. To do this, they used distributed oblivious key generation scheme utilizing Shoup's RSA based threshold signature. This scheme is modeling a key server to a group of key servers, and obtains a convergent key by interacting with t key servers to perform distributed threshold blind signature protocol. DupLESS needs a key server and is not secure against a collusion attack attempted by KS and CSP, while Duan's scheme generates a convergent key using distributed oblivious key generation scheme with the aid of other users (trusted dealers). In some ways, trusted dealer does a similar function as a key server. Because of this, Miao et al. [7] proposed multi-server-aided data deduplication scheme using threshold blind signature.

Meanwhile, Shah et al. [10] proposed Lamassu scheme. DupLESS needs a key server and an interaction between a client and a key server. In order to remove it, Lamassu scheme adds secret key to the convergent key generation process. Client is allowed to directly access this secret key, and is possible to generate one's own convergent key locally. In this approach, the clients using other secret key produce different ciphertexts for the same plaintext, and thus it is not possible to deduplicate the file for the users. If more than two clients share a secret key, those clients are capable of accessing the shared data and deduplication is possible. A set of clients that share a secret key forms an isolation zone by configuring security zone and deduplication group.

Deduplication scheme applying the privilege information was proposed first in [2]. In the proposed paper, the privilege information is applied when calculating the file authentication tag for the deduplication. The authentication tag is generated by the private cloud functioning as the authorized server, and the private cloud possesses the privilege private key corresponding to the user's privileges. This privilege private key does not be distributed to the user. For users with the same privilege, therefore, authentication tag identifying deduplication is generated and deduplication is possible.

2.2 Proofs of Ownership (PoW)

In order to prevent leakage of information about the files stored in the cloud server from the attacker without the access right to files stored in the cloud server, Halevi et al. [13] proposed the proofs of ownership protocol for the file. In this scheme, the client owning the actual file generates the authentication information from the file and by using this information, the client proves the ownership of the file by sending a response to the challenge received from the cloud server. The purpose of PoW is contrary to that of remote data checking such as PDP (Provable Data Possession)/PoR (Proof of Retreivability). In PoW, the client proves the possession of a file, but PDP/PoR is the opposite. Although these two concepts are somewhat related, PoR/PDP schemes cannot be used to build PoW because clients would need to share a common state. For PDP/PoR, see [11-12].

PoW scheme of [13] is a Merkle-Tree based protocols, and to prove ownership of the file the client has to present the exact sibling path for random block indexes that are requested from the server. This method is that client proves the ownership of the file to the server. To do this, the client configures the Merkle Tree and then sends the number of leaf nodes (which is the lowest node of the tree) and the value of the root (which is the top-level node) to the server. The server sends random indexes of leaf nodes to the client, and the client responds corresponding leaf nodes and accurate sibling path. The server computes a root value by using the values received from the client, and compares it with the received root value. If both values match, the server admits the ownership of the file. In [13], however, since the server stores unencrypted files, there is the privacy issue that the information of the file is revealed, as well as a drawback that requires a lot of operations in order to prove the ownership of the data. In addition, there is a problem that it does not guarantee the freshness of the proof in all the challenge which the cloud server requests to the client.

3 The proposed Authorized Client-side Deduplication Scheme

If the confidentiality of the data is requested by the client-side deduplication scheme, the user has to encrypt the data before uploading it to the cloud storage. To support deduplication, the convergent encryption scheme is used and it requires the generation of convergent encryption keys for this purpose. As described in Section 2, a simple technique is to use the hash value of the plaintext data as a convergent key. However, this method is vulnerable to offline brute force attack. So, DupLESS scheme introduced the key server and generated a convergent key by performing RSA blind signature based oblivious PRF protocol between the key server and the data owner. In our scheme, a privilege information is applied in this process in order to allow only the authorized user to access to data. By adding a privilege information to the generation process of a convergent encryption key, it provides the appropriate trade-off between the efficiency of deduplication and security of data sharing. By executing the deduplication for users with the same privilege, the effect of deduplication can be reduced, but in view of the data sharing, there is an advantage that only authorized users can access by uploading the encrypted file with the privilege information. When applying the hybrid cloud model for the data security and the storage efficiency, our scheme is very suitable. As with DupLESS, the proposed method uses RSA blind signature-based oblivious PRF protocol, however we add the privilege information to this process.

3.1 System Model

The proposed model consists of a user, an authorization server (or a key server), and a cloud storage provider (CSP). Here, an authorization server (AS) can be regarded as a private cloud and a CSP can be considered as a public cloud. An authorization server is an entity that helps a user to securely use a CSP. Also, AS generates and manages to the private key corresponding to the privilege, and computes a convergent encryption key for the file by applying the privilege through the interaction with a user. Thus, users are possible to perform the duplication check based on their privilege, and only the authorized user can generate a convergent encryption key. AS and CSP both are assumed to be “honest-but-curious”.

In the initialization process (or the registration process), a set of privileges is given to each user by AS. A set of privileges describes which users are allowed to perform the duplication check and the file access, and is bound to the file uploaded to the cloud. It is defined on the basis of a set of privileges. The exact definition of a privilege is slightly different for each application. For example, the role-based privilege may be defined by the title within the organization. Or time-based privilege may specify the valid access period [2].

The authorization server issues a privilege secret key corresponding to the privilege to the user, and this secret key is applied to the generation of the convergent encryption key. Also, this key does not directly being distributed to the user and the authorization server manages it securely. The reason to do so is to prevent a collusion attack between users (if this privilege secret key is distributed to the user and the user manages it, a collusion attack between users is possible easily).

It assumes that a universe of privileges is defined as $\wp = \{p_1, p_2, \ldots, p_n\}$. The secret key k_{pi} for each privilege p_i belonging to the set $\wp$ is randomly generated. A set of privileges, P_U is assigned to a user U, and the set of corresponding keys, $\{ k_{pi} \}_{pi \in P_u}$ is also assigned to the user U. As mentioned above, this set of keys is not issued to the user directly, and is securely kept and managed by the authorization server. It assumes the authorization server stores a user's identity and the set of privileges corresponding to the user identity.

Table 1 shows terms and symbols used in this paper.

Table 1. Terms and symbols

Notation	Meaning
U	A user
AS	The authorization server
CSP	The cloud storage provider
ID_U	The identity of a user U
sk	A user's secret key
K	A convergent encryption key
$H()$	A cryptographically secure hash function
p_f	A access privilege
k_{pf}	A privilege secret key corresponding to the privilege p_f. This key is managed securely by AS.
$G()$	A collision-resistant hash function
$E(k, m)$	Symmetric key encryption scheme with a secret key k and a plaintext m
C_F	An encrypted file using $E()$ with a key K
C_U	An encrypted convergent key using $E()$ with a key sk
T	A file tag which is used to identify duplicated files
$MT(F)$	A Merkle Tree configured for a file F
leaf_nodes	Leaf nodes of a Merkle Tree corresponding to specific indexes
$SP(idxs)$	Sibling path of leaf nodes corresponding to specific indexes *idxs* in a Merkle Tree

3.2 Access Privilege Based Convergent Encryption

It assumes that a user wants to upload a file F to a cloud storage provider. The proposed scheme ensures the user the confidentiality by uploading the encrypted file, and at the same time, provides CSP for the efficient use of the storage by performing the deduplication. For the file encryption supporting the deduplication, it needs the generation of a convergent encryption key. So, we generate a convergent encryption key by interacting with the authorization server to prevent an offline brute force attack. In this process, we apply the access privilege for secure sharing of uploaded files.

The generation process of a convergent encryption key uses the protocol based on RSA blind signature which is used in DupLESS. The AS generates a public and private key pair based on RSA. That is, generate two large prime numbers, p and q, and compute $N = p \cdot q$. Then find $e \cdot d \equiv 1 \pmod{\phi(N)}$, where $\phi()$ denotes Euler's totient function or Euler's phi function. A public key is (N, e) and a private key is d.

A user U selects $r \in \mathbb{Z}_n$ randomly and then computes $r^e \pmod N$ using AS's public key (N, e). And the user computes a hash value $h = H(F)$ of a uploading file F using a cryptographically secure hash function $H()$, and then computes $x = (r^e \cdot h) \bmod N$. The user sends (x, ID_U) to AS. After AS identifies the user U from ID_U, AS reads a secret key k_{pf} corresponding to the user's privilege p_f. And then computes $y = (x^d \cdot k_{pf}) \bmod N$ using the private key d and sends it to the user. The user computes $z = y \cdot r^{-1} \bmod N$, and then generates a convergent encryption key $K = G(z)$. Figure 1 shows the distribution process of a convergent encryption key between the user and AS.

3.3 File Upload

Before uploading a file F, the user has to encrypt F using the key K computed in the distribution process of a convergent encryption key. $C_F = E(K, F)$. Also, it encrypts the convergent key K using the user's secret key sk. $C_U = E(sk, K)$. Next, it computes the file tag T for the encrypted file C_F. $T = \text{TagGen}(C_F)$, where TagGen() function is a collision-resistant hash function. The user sends the tag T and the file size *FileSize* to the CSP. Then the CSP checks whether the received tag exists or not. If the same tag does not exist, it runs the first upload process. If the same tag already exists, it will perform the deduplication process. In this process, it has to perform the proofs of ownership protocol.

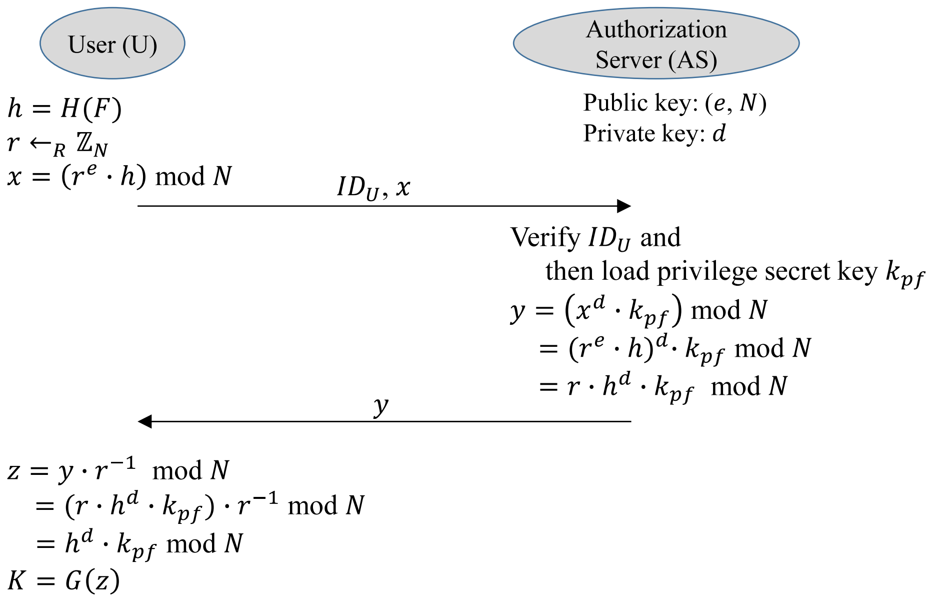

User (U)
Authorization Server (AS)
$h = H(F)$
$r \leftarrow_R \mathbb{Z}_N$
$x = (r^e \cdot h) \bmod N$
Public key: (e, N)
Private key: d
ID_U, x
Verify ID_U and then load privilege secret key k_{pf}
$y = (x^d \cdot k_{pf}) \bmod N$
$= (r^e \cdot h)^d \cdot k_{pf} \bmod N$
$= r \cdot h^d \cdot k_{pf} \bmod N$
y
$z = y \cdot r^{-1} \bmod N$
$= (r \cdot h^d \cdot k_{pf}) \cdot r^{-1} \bmod N$
$= h^d \cdot k_{pf} \bmod N$
$K = G(z)$

Figure 1. The distribution process of a convergent encryption key

First upload process. The CSP selects a random number r and a block size b that divides the file, and sends them to the user $U1$. The file will be split into nb blocks. $nb = \lceil FileSize / b \rceil$. The user configures the Merkle Tree MT by dividing the file C_F into nb blocks and calculates a root value $AuthTag = MT(C_F)$. And then it sends (C_F, C_{U1}, $rTag$) to the CSP, where $rTag = H(r||AuthTag)$. The CSP configures the Merkle Tree using the received C_F and computes the root value $AuthTag'$. If $H(r||AuthTag')$ matches the received $rTag$, the CSP stores $\{T, FileSize, b, C_F, AuthTag', ID_{U1}, C_{U1}\}$ to a storage. If not match, the CSP returns '⊥' and stops the procedure. When the upload is completed successfully, the user stores $AuthTag$ and removes the file F. Figure 2 shows the first upload process.

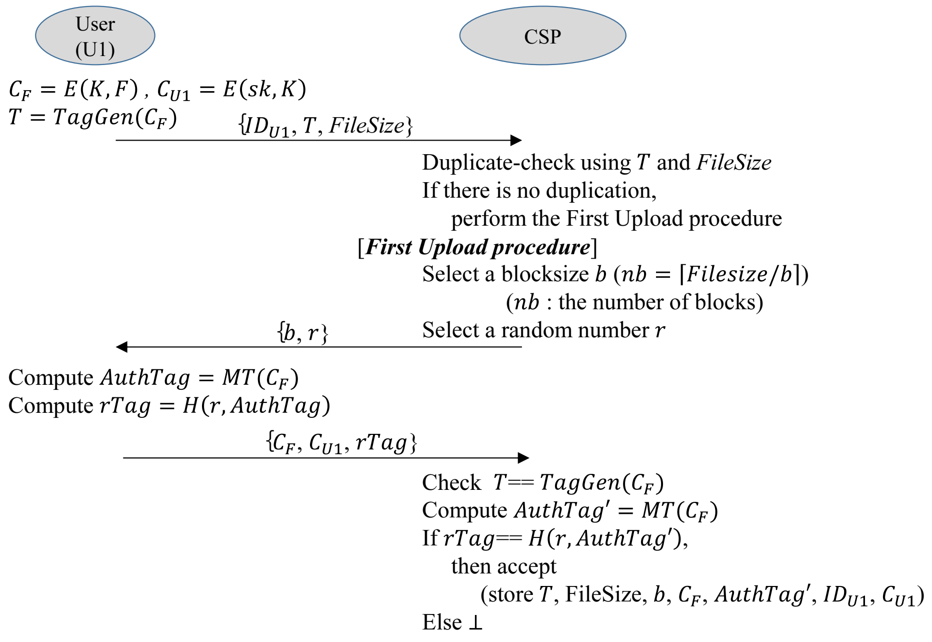

User (U1)
CSP
$C_F = E(K, F)$, $C_{U1} = E(sk, K)$
$T = TagGen(C_F)$
$\{ID_{U1}, T, FileSize\}$
Duplicate-check using T and $FileSize$
If there is no duplication,
perform the First Upload procedure
[***First Upload procedure***]
Select a blocksize b ($nb = \lceil Filesize/b \rceil$)
(nb : the number of blocks)
Select a random number r
$\{b, r\}$
Compute $AuthTag = MT(C_F)$
Compute $rTag = H(r, AuthTag)$
$\{C_F, C_{U1}, rTag\}$
Check $T == TagGen(C_F)$
Compute $AuthTag' = MT(C_F)$
If $rTag == H(r, AuthTag')$,
then accept
(store T, FileSize, b, C_F, $AuthTag'$, ID_{U1}, C_{U1})
Else ⊥

Figure 2. First upload process

Deduplication process. In this case, it assumes the user $U2$ wants to upload a duplicated file. Figure 3 shows the deduplication process. When the CSP has found a duplicated file upload, the CSP loads the block size b and $AuthTag$ from the storage. The file is divided into nb blocks. $nb = \lceil FileSize / b \rceil$. The CSP randomly selects a set of block indexes, $idxs$ which is challenged to the user. $idxs \subset \{0, 1, \ldots, nb\}$. The CSP also selects a random number r. And then the CSP requests the verification of the file ownership by sending (r, $idxs$, $cTag$) to the user, where $cTag = H(ID_{U2}, T, AuthTag)$.

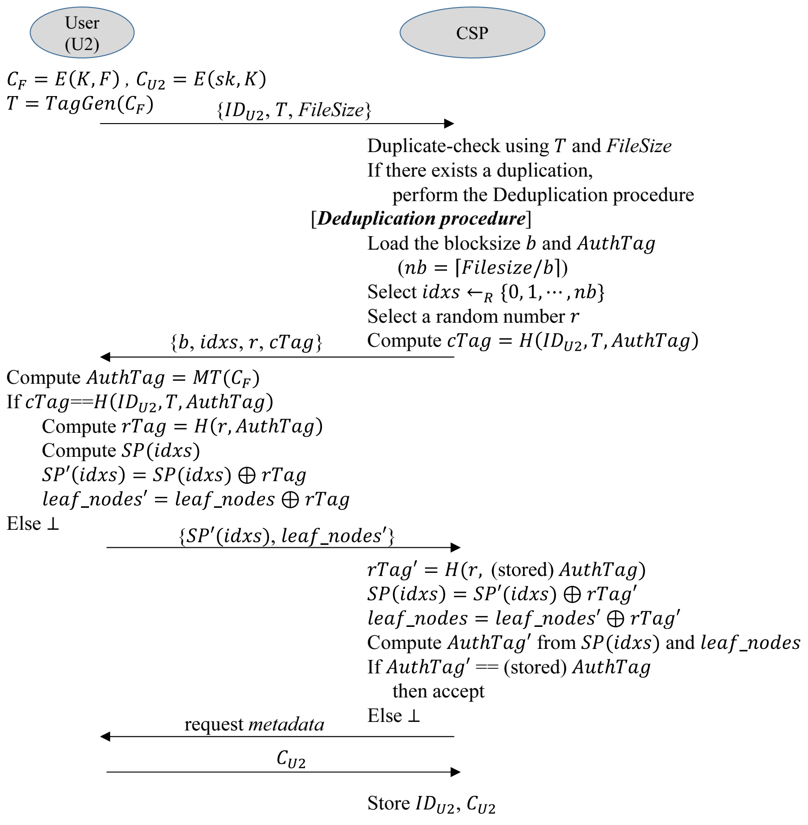

User (U2)
CSP
$C_F = E(K,F)$, $C_{U2} = E(sk,K)$
$T = TagGen(C_F)$
$\{ID_{U2}, T, FileSize\}$
Duplicate-check using T and $FileSize$
If there exists a duplication,
perform the Deduplication procedure
[***Deduplication procedure***]
Load the blocksize b and $AuthTag$
$(nb = \lceil Filesize/b \rceil)$
Select $idxs \leftarrow_R \{0, 1, \cdots, nb\}$
Select a random number r
Compute $cTag = H(ID_{U2}, T, AuthTag)$
$\{b, idxs, r, cTag\}$
Compute $AuthTag = MT(C_F)$
If $cTag{=}{=}H(ID_{U2}, T, AuthTag)$
Compute $rTag = H(r, AuthTag)$
Compute $SP(idxs)$
$SP'(idxs) = SP(idxs) \oplus rTag$
$leaf\_nodes' = leaf\_nodes \oplus rTag$
Else ⊥
$\{SP'(idxs), leaf\_nodes'\}$
$rTag' = H(r,$ (stored) $AuthTag)$
$SP(idxs) = SP'(idxs) \oplus rTag'$
$leaf\_nodes = leaf\_nodes' \oplus rTag'$
Compute $AuthTag'$ from $SP(idxs)$ and $leaf\_nodes$
If $AuthTag'$ == (stored) $AuthTag$
then accept
Else ⊥
request *metadata*
C_{U2}
Store ID_{U2}, C_{U2}

Figure 3. Deduplication process

The user *U2* computes *AuthTag'* = *MT*(C_F) and then verifies that the received *cTag* is valid. With this step, the user can check whether its own file matches a duplicated file stored on the CSP. If this verification succeeds, computes *rTag* = *H*(*r*, *AuthTag'*) and then selects *leaf_nodes* and *SP*(*idxs*) of the Merkle Tree corresponding to the received block indexes *idxs*. The user scrambles *SP*(*idxs*) and *leaf_nodes* by using *rTag*. *SP'*(*idxs*) = *SP*(*idxs*) ⊕ *rTag* and *leaf_nodes'* = *leaf_nodes* ⊕ *rTag*. The user responds *SP'*(*idxs*) and *leaf_nodes'* to the CSP.

The CSP computes *rTag'* using the stored *AuthTag*, and then descrambles *SP*(*idxs*) and *leaf_nodes* xoring it with the received *SP'*(*idxs*) and *leaf_nodes'*. After computing *AuthTag'* using *SP*(*idxs*) and *leaf_nodes*, the CSP checks if it matches the stored *AuthTag*. If valid, then the CSP admits the file ownership of the user *U2* and requests its metadata C_{U2} to *U2*. When receiving C_{U2} from the user, the CSP allows the file access by assigning the file pointer to the user and stores $\{ID_{U2}, C_{U2}\}$. If this process is completed successfully, the user stores *AuthTag* and removes the file *F*.

3.4 File Download

If the user wants to download a file *F*, he first sends the file request message to the CSP. The CSP checks the right for download of the requesting user for the file *F*, and then sends a random number *r* as a challenge to the user. The user responds *rTag* = *H*(*r*, *AuthTag*) to the CSP. If *rTag* is valid, the CSP sends to the user, C_F and C_U corresponding to the user. The user obtains a convergent key *K* by decrypting the received C_U with its secret key *sk*, and then recovers the file *F* by decrypting C_F.

4 An Improved Authorized Client-side Deduplication with Stronger Security

In this section, we review the scheme in Section 3 in terms of the security against the collusion attacks. Then we give an improved authorized client-side deduplication scheme which is secure against the collusion attacks. We also analyze the proposed scheme in terms of the security and the performance.

4.1 Insecurity of the RSA Based Authorized Convergent Key Distribution against Collusion Attacks

In the case of the above RSA-based authorized convergence encryption method, there is a disadvantage that the user cannot verify if the value *y* received from the AS is correct. There is also the possibility of collusion attacks among users. Suppose that there are two users, *A* and *B* who are authorized by k_{pf} and $k_{pf'}$, respectively. These two privilege secret keys are managed by AS. For the same file *h* = *H*(*F*), both users obtain the convergent keys, $h^d \cdot k_{pf}$ and $h^d \cdot k_{pf'}$ from AS, respectively. This allows both users to perform a collusion attack. When two users perform a division operation using each other's information, they can compute $k_{pf}^{-1} \cdot k_{pf'}$ or $k_{pf'}^{-1} \cdot k_{pf}$, respectively. A colluding attacker who has a legitimate privilege attribute can use the information to transform a convergent key to a key for another privilege attribute. For example, a user *A* with a privilege *pf* obtains $h'^{d} \cdot k_{pf}$ from AS for $h' = H(F')$. Using $k_{pf}^{-1} \cdot k_{pf'}$ obtained by colluding another user *B* with a privilege pf', he can obtain $h'^{d} \cdot k_{pf'}$ by multiplying two values. This $h'^{d} \cdot k_{pf'}$ value is an authorized convergent key material by a privilege key $k_{pf'}$ which is not access right of the user *A*. This malicious collusion attack may be assumed by various methods such as exposing existing key information by a revoked user.

4.2 BLS Signature Based Authorized Convergent Key Distribution with Stronger Security

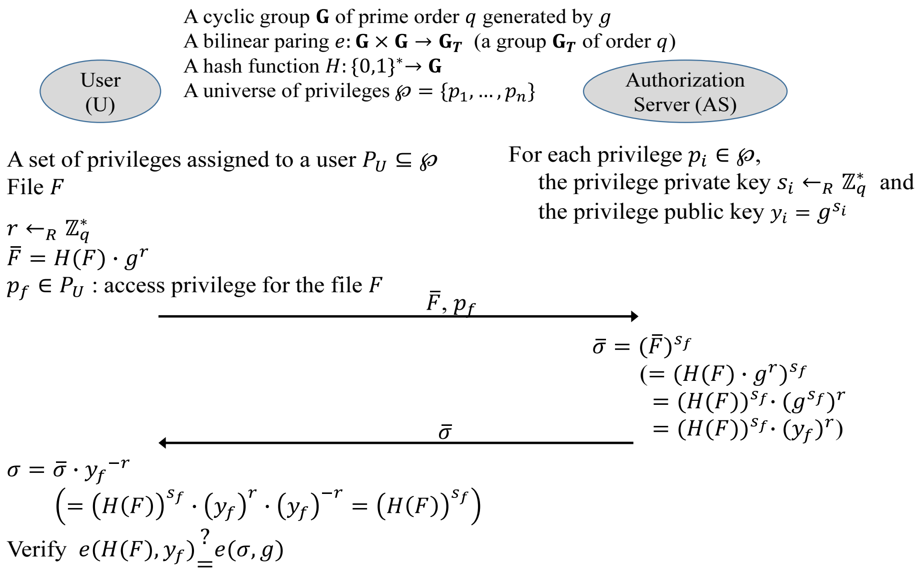
To solve these problems, we propose an authorized convergence key distribution scheme based on BLS signature. BLS signature is proposed by Boneh et al., and uses bilinear pairing based on the difficulty of the discrete logarithm problem on an elliptic curve [14]. Assuming two cyclic multiplicative group *G* and G_T with the same prime order *q*, a bilinear map $e{:}G \times G \to G_T$ satisfies three conditions; bilinearity, non-degeneracy, computability. BLS signature consists of key generation, signing and verification.

Key generation. Pick random *x* from $\mathbb{Z}_q$ and compute $v = g^x$, where g is a generator of *G*. The public key is *v* and the private key is *x*.

Signing. Given a private key *x* and a message *m*, compute *h* = *H*(*m*). The signature is $\sigma = h^x$. Here, $H{:}\{0,1\}^* \to G$ is a BLS hash function.

Verification. Given a public key *v*, a message *m*, and a signature *σ*, compute *h* = *H*(*m*) and verify that $e(\sigma, g)$ is equal to $e(H(m), g^x)$. If so, output valid; if not, output invalid.

BLS blind signature based convergence encryption is performed as Figure 4. It assumes that the user has a set of privileges, $P_U \subset \wp$ and AS has the privilege private/public key pairs (s_i, y_i) for each privilege $p_i \in \wp$. When the user uploads a file *F* to the cloud storage, it selects *r* from $\mathbb{Z}_q^*$ randomly and computes $\bar{F} = H(F) \cdot g^r$. Then the user sends $\bar{F}$ and the access privilege $p_f \in P_U$ to AS. AS computes $\bar{\sigma} = (\bar{F})^{s_f}$ using the privilege private/public key pair (s_f, y_f) for the access privilege p_f, and sends $\bar{\sigma}$ to the user. The user computes $\sigma = \bar{\sigma} \cdot (y_f)^{-r}$ and then verifies that $e(H(F), y_f)$ is equal to $e(\sigma, g)$ as following.



A cyclic group $\mathbf{G}$ of prime order q generated by g
A bilinear paring $e{:}\,\mathbf{G} \times \mathbf{G} \to \mathbf{G_T}$ (a group $\mathbf{G_T}$ of order q)
A hash function $H{:}\,\{0,1\}^* \to \mathbf{G}$
A universe of privileges $\wp = \{p_1, \dots, p_n\}$
User (U)
Authorization Server (AS)
A set of privileges assigned to a user $P_U \subseteq \wp$
File F
For each privilege $p_i \in \wp$,
the privilege private key $s_i \leftarrow_R \mathbb{Z}_q^*$ and
the privilege public key $y_i = g^{s_i}$
$r \leftarrow_R \mathbb{Z}_q^*$
$\bar{F} = H(F) \cdot g^r$
$p_f \in P_U$: access privilege for the file F
$\bar{F}, p_f$
$\bar{\sigma} = (\bar{F})^{s_f}$
$(= (H(F) \cdot g^r)^{s_f}$
$= (H(F))^{s_f} \cdot (g^{s_f})^r$
$= (H(F))^{s_f} \cdot (y_f)^r)$
$\bar{\sigma}$
$\sigma = \bar{\sigma} \cdot y_f^{-r}$
$\left(= (H(F))^{s_f} \cdot (y_f)^r \cdot (y_f)^{-r} = (H(F))^{s_f}\right)$
Verify $e(H(F), y_f) \overset{?}{=} e(\sigma, g)$

Figure 4. Authorized convergent key generation based on blind BLS signature

$$e(\sigma, g) = e(\bar{\sigma}, y_f^{-r}, g) = e((H(F) \cdot g^r)^{s_f}) \cdot (g^{s_f})^{-r}, g)$$
$$= e((H(F)^{s_f} \cdot (g^{s_f})^r \cdot (g^{s_f})^{-r}, g)$$
$$= e(H(F), y_f)$$

If the verification is successful, the user calculates the convergent encryption key *K* = *G*(σ).

Thus, BLS blind signature based scheme can improve the problems of RSA blind signature based scheme. The user can verify the correctness of the received value $\bar{\sigma}$. And an algebraic attack by colluding among the users against RSA blind signature based scheme cannot be applied to BLS blind signature based scheme. From the computational point of view, BLS blind signature based scheme has higher computational overhead than RSA blind signature based scheme.

However, the communication overhead is lowered. The length of BLS blind signature based scheme is shorter than RSA blind signature based scheme.

4.3 Analysis of Security and Complexity

4.3.1 Security Analysis

Our scheme provides an authorized deduplication, confidentiality and integrity of data stored in cloud. The security of the proposed scheme is ensured by the security of the used cryptographic primitives such as hash functions, symmetric and asymmetric encryption. In the security of the proposed scheme, the authorization server plays an important role. Because it is possible to carry out a brute force attack if the authorization server colludes with an attacker, it is assumed that the authorization server does not collude with an attacker. Also it assumes that the authorization server has similar conditions as the key server in DupLESS.

The first feature to prove is the correctness of the functionality so-called the authorized deduplication.

Lemma 1. *The proposed scheme provides an authorized deduplication.*

Proof. The proposed method is possible to deduplicate the same files of users with the same permissions. The same files derive the same value $h = H(F)$, and the same privilege secret key k_{pf} is applied when the users have the same privilege. So, the same convergent key K is derived and deduplication is possible. An unauthorized user cannot obtain the same convergent key K because k_{pf} is kept securely only in AS. Also, users cannot obtain any information on k_{pf} from y due to security of AS's private key d. Thus, our scheme provides an authorized deduplication under the assumption of the security of RSA signature and the privilege secret keys.

Comparing with existing schemes which do not apply access privileges, the efficiency of deduplication of the proposed scheme will be reduced. Thus, the appropriate trade-off between the security of the sensitive files and the efficient use of storage space must be taken into account.

To protect the privacy of users' data stored in CSP's storage, all data will be stored in an encrypted form, which means that the confidentiality of data stored in cloud is one of important security requirements. In the below, we will prove that the proposed scheme achieves the feature.

Lemma 2. *The proposed scheme guarantees that only eligible users can access the plain data. That is, it provides the confidentiality of data stored in cloud.*

Proof. If the CSP colludes with an attacker, the attacker can access to an encrypted data C_F and metadata C_U. However, the proposed method can ensure security of a plaintext data even with a low entropy because C_F has encrypted with a key K issued by the authorization server and also the key K has computed by applying the AS's private key and a privilege secret key corresponding to the user's privilege attribute. As long as an attacker does not know the user's secret key sk, he cannot decrypt C_U because it has encrypted with sk.

A legitimate user behaving as a malicious attacker can threaten the AS and the CSP. When attacking the AS, an attacker has to address the factorization problem to obtain the private information of the AS to address the factorization problem by the property of the RSA blind signature-based OPRF protocol, and so it can ensure security of the AS. When attacking an encrypted data stored in the CSP, an attacker has to perform a proofs of ownership protocol for the data. To do this in the proposed scheme, an attacker has to construct Merkle Tree for the encrypted data in the proofs of ownership process and has to submit a sibling path for block indexes randomly selected by the CSP. The security of Merkle Tree based proofs of ownership scheme is presented in [13, 15].

For convergent encryption schemes, the confidentiality can be easily broken if the adversary correctly guesses the secret key. Since the key is derived from the data itself, it is possible to find the secret key by guessing the encrypted data. To improve the security against guessing attacks, DupLESS has been proposed. Due to the invention of DupLESS, in an adversary's view point, the only way to find the key is to perform online brute-force attacks. Hence, the security against the online brute-force is also important issue for the confidentiality of data stored in cloud. To provide resistance to online brute-force attack, DupLESS restricts the number of key issuing requests per a client by the rate limiting strategy. However, DupLESS has a problem that online brute-force attack is possible when an attacker colludes with multiple clients. The proposed scheme can provide higher security than DupLESS because the proposed scheme applies access privileges to the key issue procedure and so has a restriction that an attacker has to collude only with the users with the same privilege.

The last security feature to prove is the integrity of stored data. In storage services with deduplication, a user's data will not be stored if the same file is already stored. However, if any spoiled file is stored instead of the original file, the new uploader may lose his file. Hence, we need to guarantee the integrity of stored files. Note that, for cloud storage services, we have to counter the poison attacks to guarantee the integrity. If the CSP does not check the integrity of an uploaded file when a client encrypts it with a randomly selected key and uploads to the cloud storage in case of client-side deduplication, a poison attack can be mounted by an adversary [16]. A poison attack proceeds as follows: A client uploads a file tag and an encrypted file, ($H(F)$, $C=E_k(F)$) to the CSP. When a malicious user stores a file tag and a modified encrypted file, ($H(F)$, C') to the cloud storage, the problem arises in case that the legitimate users thereafter upload the duplicated file *F*. When a legitimate user requests the upload of the file *F*, the cloud server decides to perform the deduplication procedure. Therefore, the user does not need to upload the file and deletes the file *F*. Later if the client wants to download the file *F*, the modified file *C'* will be sent from the server and the user is faced with a problem that the original file *F* is lost. To guarantee the security of the proposed scheme against the above described attack, we give the following lemma which proves the integrity of the proposed scheme.

Lemma 3. *The proposed scheme guarantees resistance to a poison attack. That is, it provides the integrity of data stored in cloud.*

Proof. A poison attack occurs because the cloud server does not perform the integrity check of a received file properly. To deal with this problem, the CSP of the proposed scheme checks the integrity of a received file in the first upload process. Also, the user makes certain whether the value $cTag = H(ID_{U2}, T, AuthTag)$ is valid, for the integrity check of the file stored in the CSP. In order to prevent a poison attack, it runs the deduplication process only if valid. So, the proposed scheme guarantees the integrity of data during the uploading process for both a user and CSP.

Including the above proved main requirements, the proposed improved scheme has many merits than existing techniques. From now, we will compare our scheme with existing techniques.

Merkle Tree based proofs of ownership scheme [10] does not guarantee the freshness of challenges. If an attacker which does not possess the file observes challenged block indexes and several leaf nodes and sibling paths of the proof delivered as the response, there exists a possibility that an attacker can hereafter derive a valid proof. However, the proposed scheme guarantees the freshness by means of the values such as a random number *r*, *cTag*, *rTag* in the proofs of ownership protocol, and so protects against a guessing attack of a valid proof.

The deduplication scheme applying the usage right was first proposed in [2]. This scheme considers the private cloud as the authorization server which maintains the privilege secret keys corresponding to the user's access right. Before uploading a file *F* to the CSP, the user receives two file authentication tags $\phi_{F,p_\tau} = H_0(H(F), k_{p_\tau})$ and $\phi'_{F,p_\tau} = H_1(H(F), k_{p_\tau})$ from the private cloud which both are computed by applying the privilege secret key k_{p_τ}. Then the user sends ϕ'_{F,p_τ} to the CSP and performs the duplication check. If not duplicated, the user selects a random key *k* and computes the encrypted file C_F using this key. Also, the key *k* is encrypted into ciphertexts $\{CK_{k,p_j}\}$ with $\{k_{F,p_j} = \phi_{F,p_j} \oplus H_2(F)\}$. Both the encrypted file C_F and the encrypted keys $\{CK_{k,p_j}\}$ are uploaded to the CSP. The user must save $\{k_{F,p_j}\}$. This scheme performs the deduplication process by applying the privilege information in the generation of the authentication tag. On the other hand, our proposed scheme uses the privilege information in the convergent key generation process. In [2] it encrypts the file with a random key *k* and then generates $\{k_{F,p_j}\}$ to encrypt the key *k*. Thus, there is a disadvantage that it must generate and save new $\{k_{F,p_j}\}$ for each file. That is, the number of keys that the user must maintain securely increases by the number of files. But in the proposed scheme, the user needs to save only one's own secret key because it uploads the convergent key encrypted by the user's secret key to the CSP after generating the convergent key. Therefore, the proposed scheme has the advantage compared to [2] in terms of the key management of the user. Also, it seems that [2] uses Merkle Tree based PoW scheme of [10] as it is instead of suggesting new PoW scheme. In this case, it has to compute an authentication tag applying the privilege and generate a PoW tag based on Merkle Tree, separately. As a result, the proposed scheme has a lower computational overhead compared to [2] in terms of the generation of the authentication tag.

Table 2 shows the comparison with the scheme of [2] which has same goals as the proposed scheme in terms of security features provided.

Table 2. Comparison of provided security features

	Authorized deduplication	Confidentiality of data	Integrity of data
[2]	Provided by authorized file tag	Not guaranteed	Vulnerable to Poison attack
Proposed scheme	Provided by authorized convergent key	Guaranteed	Guaranteed

Both [2] and our scheme provide an authorized deduplication. In case of [2], this feature is provided by an authorized file token corresponding to access privileges, while our scheme supports it by distributing an authorized convergent key corresponding to access privileges.

To provide data confidentiality in [2], a user uploads an encrypted file C_F and encrypted keys $\{CK_{k,p_j} = \mathrm{Enc}_{\mathrm{SE}}(k_{F,p_j}, k)\}$ to CSP, where a random file encryption key *k* is encrypted by $\{k_{F,p_j} = \phi_{F,p_j} \oplus H_2(F)\}$ corresponding to each priviledge attribute p_j. In this process, an attacker may obtain k_{F,p_j} if he obtains one of ϕ_{F,p_j} through observation of the protocol because an attacker can obtain the hash value $H_2(F)$ of the file *F* by the guessing attack. So, an attacker may compute the random file key k by decrypting CK_{k,p_j} with the obtained k_{F,p_j}. It is also possible to obtain the key through another method. If an attacker does not have same set of privildges but has at least one common attribute among priviledge attributes, he is also able to obtain k_{F,p_j}. Therefore, the scheme of [2] does not guarantee the confidentiality of the file, unlike the authors' claims. In the proposed scheme, a user who does not have a proper set of access rights cannot generate a corresponding convergence key. Also, since the convergence key for file encryption is encrypted by the user's secret key *sk* and stored in CSP and stored in the CSP, an attacker who does not know *sk* can not access the plain file. Therefore, the proposed scheme provides the data confidentiality by lemma 1 and 2.

The scheme of [2] is vulnerable to the poison attack because CSP can not verify the proper association of the received C_F with the file tag/token in the uploading procedure and a user does not perform the integrity verification for duplicate files stored in CSP in the deduplication process. However, the proposed scheme guarantees the integrity of data because CSP verifies the received C_F in the uploading procedure and a user also performs the integrity check of the duplicate file stored in CSP in the improved MT based PoW process.

4.3.2 Complexity Analysis

We compare computational complexity and storage overhead with [2] whose goal is the same with our scheme. First, Table 3 shows comparison of computational complexity.

Table 3. Comparison of computational complexity

Entity	Process	[2] (except for PoW)	Proposed Scheme (RSA based, except for PoW)
User	Tag & MLE key Generation	2*H	3*H + 1*Exp + 1*ModInv + 2*ModMul
	File Encryption	1*SEnc	1*SEnc
	MLE key Encryption	n*SEnc + n*H	1*SEnc
AS (Private cloud in [2])	Tag & MLE key Generation	4n*H	1*Exp + n*ModMul
	File Uploading	1*SigVer + 4n*H + 1*SigGen	-
CSP	First upload	1*SigGen	1*H
	Deduplication	1*SigGen <+ PoW>	<PoW>

Notes. n: number of access privileges; SEnc: symmetric encryption; H: hash function; SigGen & SigVer: digital signature generation and verification; Exp: exponentiation; ModInv: Modular Inversion; ModMul: Modular Multiplication; PoW: Proofs of Ownership procedure; <>: omission of a procedure.

To compare the computation overhead, since the private cloud in [2] plays a similar role to AS in the proposed method, it considers both entities as the same and compares them. Also, in order to simplify the comparison, we compare the computational complexity except for the PoW process in both schemes because [2] does not explicitly describe the PoW process. Basically, [2] is based on hash function, symmetric cryptographic primitive and digital signature, while our scheme uses a hash function, a symmetric cryptographic primitive, a modular operation, and an RSA blind signature.

According to Table 3, from the user's point of view, the key generation process has overhead of 2*H in case of [2], and the overhead of (3*H + 1*Exp + 1*ModInv + 2*ModMul) in case of our scheme. Also, in the key generation step of AS, [2] has computational overhead of 4n*H and the proposed technique has (1*Exp + n*ModMul). Since the proposed scheme applies the privilege information in the process of generating the convergence key, the computational overhead of the user side in the process of generating the convergence key is higher than [2]. However, in other processes, our scheme is more efficient than [2]. Especially, in the MLE key encryption process, [2] has (n*SEnc + n*H) computations, and in addition, AS must interact with the user during the file upload phase, which has the computational overhead of (1*SigVer + 4n*H + 1*SigGen). In the proposed scheme, AS participates only in the MLE (CE) key distribution process and thereafter does not participate in the protocol. However, [2] has a disadvantage that the private cloud continues to involve in the file uploading process.

Table 4 shows the comparison results in terms of storage overhead. In terms of user-side storage, a user of [2] must store $\{k_{F,p_j}\}$ after the file uploading. It requires (2*n*h_len) storage space for the file token and keys for each file. In the proposed scheme, the storage overhead of the user is very small compared to [2] because a file tag (1*h_len) for each file and only one secret key *sk* need to be stored in each user side.

Table 4. Comparison of storage overhead on a user and CSP

	[2]	Proposed scheme
User	N_u * (2*n*h_len)	1*key_len + N_u*(1*h_len)
CSP	N_{csp} * {n*h_len + 1*ct_len + n*senc_len}	N_{csp} * {1*h_len + 1*ct_len + N_d * senc_len}

Notes. N_u: number of files owned and uploaded by one user; N_{csp}: number of files that are uploaded by all users and managed on CSP; N_d: number of deduplicaed file uploads; h_len: bit-length of the output of the hash function; key_len: bit-length of the secret key of the symmetric encryption; ct_len: bit-length of the encrypted file; senc_len: bit-length of the output of the symmetric encryption.

For CSP, [2] must store (n*h_len) for the file token and the ciphertext C_F and encrypted keys $\{CK_{k,p_j}\}$ of each file. For this, it requires (1*ct_len + n*senc_len) storage space. The proposed scheme needs to store the file tag, the ciphertext and the encrypted key, and it requires storage space of (1*h_len + 1*ct_len + 1*senc_len). Therefore, the proposed scheme is more efficient in terms of CSP-side storage overhead. Also, the proposed technique is more efficient from the perspective of the number of keys the user has to manage.

5 Conclusion

In this paper, we proposed authorized client-side deduplication scheme which provides efficient use of cloud storage while supporting secure data sharing in the cloud. To support deduplication, the proposed scheme used the convergent encryption method, and applied an access privilege to compute a convergent key. Due to this, the user without proper privileges will not be able to generate a convergent key and thus cannot access the shared data. To verify the ownership of the file in client-side deduplication procedure, we also proposed a new proof of ownership protocol based on an existing Merkle Tree-based proofs of ownership protocol. The proposed method provides an adequate trade-off between security and storage space efficiency. By executing the deduplication for users with the same privilege, the effect of deduplication can be reduced, but in view of the data sharing, there is the advantage that only authorized users can access by uploading the encrypted file with the privilege information. The proposed scheme is very suitable for the hybrid cloud model considering both the data security and the storage efficiency.

Acknowledgements

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [17ZH1700, Development of storage and search technologies over encrypted database]. Sang Uk Shin is the corresponding author.

References

[1] J. Gantz, D. Reinsel, The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East, *IDC iView: IDC Analyze the Future*, Vol. 2007, pp. 1-16, December, 2012.

[2] J. Li, Y.-K. Li, X. Chen, P. P. C. Lee, W. Lou, A Hybrid Cloud Approach for Secure Authorized Deduplication, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 26, No. 5, pp. 1206-1216, May, 2015.

[3] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, M. Theimer, Reclaiming Space from Duplicate Files in a Serverless Distributed File System, *Proceedings 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, 2002, pp. 617-624.

[4] D. Harnik, B. Pinkas, A. Shulman-Peleg, Side Channels in Cloud Services: Deduplication in Cloud Storage, *IEEE Security & Privacy*, Vol. 8, No. 6, pp. 40-47, December, 2010.

[5] M. Bellare, S. Keelveedhi, T. Ristenpart, DupLESS: Server-aided Encryption for Deduplicated Storage, *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, Washington, DC, pp. 179-194.

[6] Y. Duan, Distributed Key Generation for Encrypted Deduplication: Achieving the Strongest Privacy, *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security*, Scottsdale, AZ, 2014, pp. 57-68.

[7] M. Miao, J. Wang, H. Li, X. Chen, Secure Multi-server-aided Data Deduplication in Cloud Computing, *Pervasive and Mobile Computing*, Vol. 24, pp. 129-137, December, 2015.

[8] S. Yu, C. Wang, K. Ren, W. Lou, Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing, *2010 Proceedings IEEE INFOCOM*, San Diego, CA, 2010, pp. 1-9.

[9] T. Youn, K. Chang, K. H. Rhee, S. U. Shin, Authorized Convergent Encryption for Client-side Deduplication, *IT CoNvergence PRActice (INPRA)*, Vol. 4, No. 2, pp. 9-17, June, 2016.

[10] P. Shah, W. So, Lamassu: Storage-Efficient Host-Side Encryption, *USENIX Annual Technical Conference (USENIX ATC 15)*, Santa Clara, CA, 2015, pp. 333-345.

[11] J. Wei, J. Liu, R. Zhang, X. Niu, Y. Yao, Public Possession Checking for Shared Data with User Revocation in Cloud Computing, *Journal of Internet Technology*, Vol. 17, No. 6, pp. 1141-1149, November, 2016.

[12] S. Shin, T. Kwon, A Survey of Public Provable Data Possession Schemes with Batch Verification in Cloud Storage, *Journal of Internet Services and Information Security (JISIS)*, Vol. 5, No. 3, pp. 37-47, August, 2015.

[13] S. Halevi, D. Harnik, B. Pinkas, A. Shulman-Peleg, Proofs of Ownership in Remote Storage Systems, *Proceedings of the 18th ACM Conference on Computer and Communications Security*, Chicago, IL, 2011, pp. 491-500.

[14] D. Boneh, B. Lynn, H. Shacham, Short Signatures from the Weil Pairing, *Journal of Cryptology*, Vol. 17, No. 4, pp. 297-319, September, 2004.

[15] R. C. Merkle, A Certified Digital Signature, *Advances in Cryptology - CRYPTO'89 Proceedings*, Santa Barbara, CA, 1989, pp. 218-238.

[16] N. Kaaniche, M. Laurent, A Secure Client Side Deduplication Scheme in Cloud Storage Environments, *2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*, Dubai, United Arab Emirates, 2014, pp. 1-7.

Biographies

Taek-Young Youn received his B.S., MS, and Ph.D. from Korea University in 2003, 2005, and 2009, respectively. He is currently a senior researcher at Electronics and Telecommunications Research Institute (ETRI), Korea. His research interests include cryptography, information security, and data privacy.

Ku-Young Chang received his B.S., M.S. and Ph.D. degrees in mathematics from Korea University, Seoul, Korea on 1995, 1997, and 2000, respectively. He is currently a principal researcher of Cryptography Research Section at Electronics and Telecommunication Research Institute, Daejeon, Korea. His research interests include cryptography, data privacy, and finite field theory.

Kyung-Hyune Rhee received his M.S. and Ph.D. degrees from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea in 1985 and 1992, respectively. He worked as a senior researcher in Electronic and Telecommunications Research Institute (ETRI), Daejeon, Korea from 1985 to 1993. He also worked as a visiting scholar in the University of Adelaide in Australia, the University of Tokyo in Japan, the University of California at Irvine in USA, and Kyushu University in Japan. He has served as a Chairman of Division of Information and Communication Technology, Colombo Plan Staff College for Technician Education in Manila, the Philippines. He is currently a professor in the Department of IT Convergence and Application Engineering, Pukyong National University, Busan, Korea. His research interests center on multimedia security and analysis, key management protocols and mobile ad-hoc and VANET communication security.

Sang Uk Shin received his M.S. and Ph.D. degrees from Pukyong National University, Busan, Korea in 1997 and 2000, respectively. He worked as a senior researcher in Electronics and Tele-communications Research Institute, Daejeon Korea from 2000 to 2003. He is currently a professor in Department of IT Convergence and Application Engineering, Pukyong National University. His research interests include digital forensics, e-Discovery, cryptographic protocol, mobile and wireless network security and multimedia content security.