Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 7840917, 11 pages
https://doi.org/10.1155/2019/7840917

Research Article

Authorized Client-Side Deduplication Using CP-ABE in Cloud Storage

Taek-Young Youn,[1] **Nam-Su Jho**,[1] **Kyung Hyune Rhee**,[2] **and Sang Uk Shin**[2]

[1]*Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, Republic of Korea*
[2]*Department of IT Convergence and Application Eng., Pukyong National University, Busan 48513, Republic of Korea*

Correspondence should be addressed to Sang Uk Shin; shinsu@pknu.ac.kr

Received 17 January 2019; Revised 11 March 2019; Accepted 15 April 2019; Published 15 May 2019

Academic Editor: Ilsun You

Since deduplication inevitably implies data sharing, control over access permissions in an encrypted deduplication storage is more important than a traditional encrypted storage. Therefore, in terms of flexibility, data deduplication should be combined with data access control techniques. In this paper, we propose an authorized deduplication scheme using CP-ABE to solve this problem. The proposed scheme provides client-side deduplication while providing confidentiality through client-side encryption to prevent exposure of users' sensitive data on untrusted cloud servers. Also, unlike existing convergent encryption schemes, it provides authorized convergent encryption by using CP-ABE to allow only authorized users to access critical data. The proposed authorized deduplication scheme provides an adequate trade-off between storage space efficiency and security in cloud environment and is very suitable for the hybrid cloud model considering both the data security and the storage efficiency in a business environment.

1. Introduction

Cloud computing greatly facilitates data providers who want to outsource their data to the cloud without disclosing their sensitive data to external parties and would like users with certain credentials to be able to access the data [1]. This requires that the data be stored in an encrypted form that supports access control policies so that no one but a user with a particular type of attributes (privilege information) can decrypt the encrypted data. On the other hand, the amount of data stored in the storage of cloud storage providers (CSPs) is growing very rapidly, especially at the age of big data. Therefore, one of the important issues of the CSP is how to efficiently manage the ever-increasing data. One of the important techniques to solve this problem is deduplication. Deduplication is a special data compression technique to remove redundant copies of repeated data and can be used to effectively reduce data storage space and communication overhead.

However, in the corporate environment, employees have different permissions to information depending on their job or department. That is, different department employees have different access rights to information according to the various access control systems used [2]. Each employee can only access files that correspond to its privileges. If the deduplication technique does not check file privileges, it would violate file access rights, which would bring some security problems. Since deduplication inevitably implies data sharing, control over access permissions in encrypted deduplication storage is more important than a traditional encrypted storage. Therefore, in terms of flexibility, data deduplication should be combined with data access control techniques. That is, even if encrypted, the same data must be stored only once in the cloud, and it must be able to control access by other users based on the policy of the data owner. Existing secure deduplication techniques do not support access policy based encryption. In the current cloud computing environments, access policy-based encryption (typically ABE (Attribute Based Encryption) scheme) and secure deduplication are widely adopted separately. However, the standard ABE technique does not support secure deduplication, a technique that helps save storage space and network

bandwidth by removing redundant copies of encrypted data in the cloud. Therefore, a design of a cloud storage system supporting both of these properties is required.

In this paper, we propose an authorized deduplication scheme using CP-ABE (Ciphertext-Policy Attribute-Based Encryption) to solve this problem. The proposed scheme provides client-side deduplication while providing confidentiality through client-side encryption to prevent exposure of users' sensitive data on untrusted cloud servers. Also, unlike existing convergent encryption schemes, it provides authorized convergent encryption by using CP-ABE to allow only authorized users to access critical data. The proposed scheme satisfies security requirements and has advantages over existing schemes. The proposed authorized deduplication scheme provides an adequate trade-off between storage space efficiency and security in cloud environment and is very suitable for the hybrid cloud model considering both the data security and the storage efficiency in a business environment.

This paper is organized as follows. Section 2 describes related works. In Section 3, we propose and analyse an authorized client-side deduplication using CP-ABE. And Section 4 describes the optimization and discussions. Finally, Section 5 is the conclusions.

2. Related Works

2.1. Secure Deduplication. Deduplication can be classified into client-side deduplication and server-side deduplication, and client-side deduplication is advantageous from the viewpoint of the efficient use of bandwidth. Therefore, many studies on client-side deduplication have been made. Despite the many advantages of deduplication technique, deduplication technique for the important data has caused several new security problems. In particular, securing the confidentiality of outsourced data is a very important issue, and it may be considered to perform an encryption operation before outsourcing. When a conventional encryption is used, each user has a different key, so that different ciphertexts are computed for the same plaintext. Therefore, deduplication cannot be achieved in this case. To solve this problem, Douceur et al. proposed a convergent encryption using a hash value of a plaintext as an encryption key [3]. Here, the encryption scheme $E()$ is a deterministic algorithm, and a convergent key K depends only on the input data file F. In most cases, however, a convergent encryption is vulnerable to offline brute-force attacks because the plaintext space of a given ciphertext C is not large enough (messages are often predictable) [4]. An attacker can perform the encryption operation for all possible plaintexts at the offline stage to find the corresponding plaintext information. To solve this problem, Bellare et al. proposed a technique called DupLESS that generates a convergent key through the interaction with a key server [5]. Duan et al. proposed a technique for generating a convergent key with the help of other users without a key server [6]. Miao et al. also proposed a method by modifying a single key server in DupLESS as a group of key servers [7].

In [8], Li et al. first proposed a deduplication scheme that applies the privilege information. In this scheme, the privilege information is applied when calculating an authentication tag of the file for 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, which are not distributed to the user. Therefore, users with the same privilege information can generate the authentication tag identifying deduplication, so deduplication is possible. Shin et al. applied an access privilege to compute a convergent key [9]. Privilege information is applied in RSA blind signature based oblivious PRF protocol in order to allow only the authorized user to access to data.

2.2. CP-ABE. Sahai and Waters introduced Attribute Based Encryption (ABE) which is the public key cryptography of one-to-many algorithm and encrypts the data based on the set of attributes [10]. There are two kinds of ABE schemes: Ciphertext-Policy ABE (CP-ABE) and Key-Policy ABE (KP-ABE) [11]. In KP-ABE schemes, the ciphertext is associated with a set of attributes while the user's private key is generated based on his corresponding access policy, while in CP-ABE schemes, a user's private key is associated with a set of attributes, and ciphertext is encrypted under a specified access structure [12]. A user is able to decrypt a ciphertext only if the attributes associated with the ciphertext/private key satisfy the access policy related to his private key/ciphertext. Most ABE schemes [11, 12] are constructed based on bilinear pairing. We also use CP-ABE scheme based on bilinear pairing in [12].

In ABE scheme, attributes can be defined with different types of privilege properties. An access structure $\wp$ will contain some authorized sets of attributes. So an access policy can be defined as an access structure. An access structure can be represented by an access tree [11, 12]. In this paper, we use the definitions of an access structure and an access policy in [11, 12], and the identities of parties may be replaced by the attributes.

3. Authorized Client-Side Deduplication Using CP-ABE

In this section, we propose an authorized client-side deduplication scheme based on the CP-ABE. The proposed scheme provides the confidentiality through the client-side encryption to prevent the exposure of users' sensitive data on untrusted cloud servers. It also enables deduplication of the encrypted data through the convergent encryption, thereby saving the storage space in the cloud server. Unlike existing convergent encryption schemes, it provides an authorized convergent encryption based on CP-ABE to allow only authorized users to access critical data.

3.1. The System and Threat Model. The proposed scheme consists of User (U), Authorization Server (AS), and Cloud Service Provider (CSP), as shown in Figure 1. Here, the AS can be regarded as a private cloud and the CSP can be considered as a public cloud. The AS is an entity that helps a user to securely use a CSP. Also, the AS generates and manages to the user's privilege secret key corresponding to the access

privilege and computes an authorized convergent key for a file by applying the privilege through the interaction with a user. The *CSP* provides data storage services to users, and deduplication technology is applied to save storage space and cost.

The proposed scheme consists of the system setup process, the authorized convergent key generation process, and the authorized deduplication process. In the system setup process, the system parameters are generated and the key pair of each entity is securely generated. The authorized convergent key generation process performs an interaction for the CP-ABE-based convergent key distribution between the user and the authorization server. Finally, the authorized deduplication process supports the client-side deduplication of the encrypted data using the generated convergent key, and the proofs of ownership (PoW) [13] protocol for verifying the file retention of the user are performed in this process (a concrete PoW protocol is not discussed in this paper, but a PoW protocol such as Merkle tree-based scheme [13] can be used).

We assume that the *AS* assumes an *honest-but-curious* trust model and the *AS* and the *CSP* do not collude. It is assumed that the user U can act maliciously. We consider that the *CSP* may act maliciously due to insider/outsider attacks, software/hardware malfunctions, intentional saving of computational resources, etc. [14]. According to this assumption, we consider the following types of adversaries:

(i) Outside adversary: it tries to obtain sensitive information from the cloud storage or tries to access the file beyond its access privileges.

(ii) Insider adversary: it can access the cloud storage easily and try to get information out of the user's encrypted data or file tags.

We consider the following security requirements that must be satisfied according to the threat model:

(i) The confidentiality of data stored in cloud: except for the information about duplication, no information about the outsourced data is revealed to an adversarial party.

(ii) The unforgeability of the file tag: a user without appropriate privileges should be prevented from generating file tags.

(iii) Secure deduplication: secure deduplication is supported without revealing any information except for the information about duplication.

3.2. The Proposed Scheme. As shown in Figure 2, the proposed scheme consists of the system setup phase, the authorized convergent key generation phase, and the file transfer phase. In the system setup phase, the user and the AS set their own keys. The authorized convergent key generation phase generates a user's convergent key for a file by reflecting the user's privilege information. The authorized file transfer phase contains the file upload and the file download, and the file upload is composed of the first upload module and the deduplication module depending on whether or not the files are duplicated. The detailed protocols are as follows.

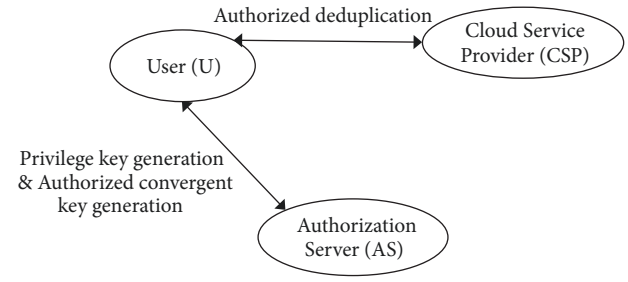


Figure 1: System model.

3.2.1. The System Setup Phase. Users and the *AS* set their own keys by the security parameter λ. First, the *AS* selects the following as the system-wide public parameters:

(i) A cyclic group G of a prime order q generated by g

(ii) A bilinear paring $e : G \times G \longrightarrow G_T$ (a group G_T of order q)

(iii) A hash function $H : \{0,1\}^* \longrightarrow G$

(iv) A universe of privileges $\wp = \{p_1, p_2, \cdots, p_n\}$

The *AS* selects values $a, b \in \mathbb{Z}_q$ randomly. Then a master secret key (MSK) of the *AS* is (g^a, b) and a public key (PK) is $(g, g^b, e(g,g)^a)$. In addition, the *AS* generates a private key and a public key for the signing as follows:

(i) Private key for the signing, $x \xleftarrow[R]{} \mathbb{Z}_q$

(ii) Public key for the signing, $y = g^x$

It is assumed that the public parameters and the public key information are securely distributed throughout the system.

A user U obtains the privilege secret key SK_U corresponding to his access privilege $A_U = \{p_i\}$ ($A_U \subseteq \wp$) from the *AS* over the secure channel. To do this, the user U sends (ID_U, A_U) to the *AS* to request the key generation. After choosing $t \in_R \mathbb{Z}_q$ and $t_j \in_R \mathbb{Z}_q$ for each attribute $j \in A_U$, the *AS* constructs the user's privilege secret key $SK_U = (D = g^{(a+t)/b}, \{D_j = g^t \bullet H(p_i)^{t_j}, g^{t_j}\}_{\forall j \in A_U})$ to securely transmit it to the user U (see [12] for details on the privilege secret key generation for CP-ABE scheme). The user U securely stores the privilege secret key SK_U. The user also generates a private key and a public key pair (spk, ssk) for a digital signature scheme such as DSA [15].

3.2.2. The Authorized Convergent Key Generation Phase. Before requesting the file upload, the user must perform the authorized convergent key generation phase. The user U generates a convergent key CK for encrypting and uploading the file F to the CSP as follows. This process modifies the RSA oblivious PRF-based convergent key generation method of [5] and generates a convergent key reflecting the user's privilege information by applying the CP-ABE scheme (see [12] for details on the encryption and decryption process for CP-ABE scheme). Figure 3 shows the authorized convergent key generation process.

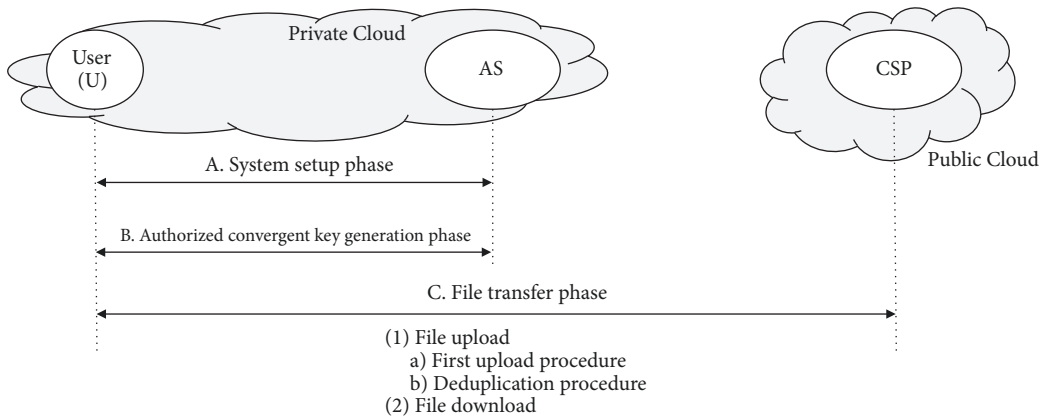


Figure 2: The proposed scheme.

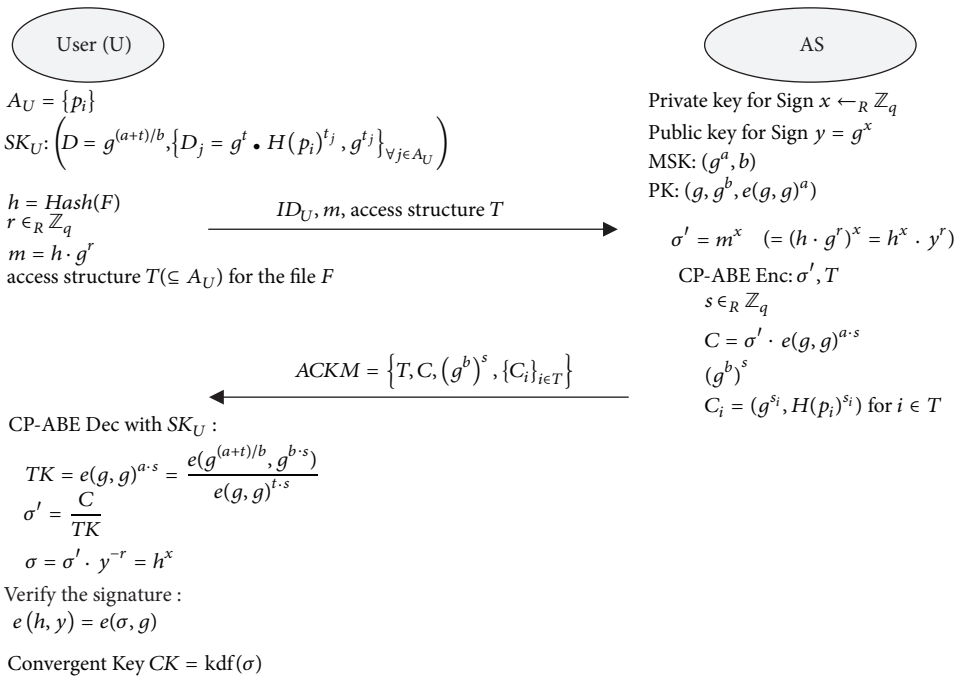


Figure 3: The authorized convergent key generation process.

(i) A user U with access privileges $A_U = \{p_i\}$ constructs a request message for a convergent key generation for file F as follows and sends it to the AS:

(a) compute $h = Hash(F)$, where $Hash()$ is a cryptographically secure hash function, $Hash : \{0,1\}^* \longrightarrow \mathbb{Z}_q$.
(b) select $r \in_R \mathbb{Z}_q$
(c) compute $m = h \cdot g^r$
(d) construct an access structure T ($\subseteq A_U$) for the file F (an access structure T is usually expressed in a tree form; see [12] for details) and send the request message for the convergent key generation to the AS.

(ii) The AS responds to the user U by constructing $ACKM$ (Authorized Convergent Key Material) as follows:

(a) compute $\sigma' = m^x \ (= (h \cdot g^r)^x = h^x \cdot y^r)$
(b) perform $CP\text{-}ABE\ Enc(\sigma', T)$ with PK:
(1) select $s \in_R \mathbb{Z}_q$ and compute $(g^b)^s$ (where s is set to the value corresponding to the root node of the tree access structure T)
(2) compute $C = \sigma' \bullet e(g,g)^{a \bullet s}$
(3) output $C_i = (g^{s_i}, H(p_i)^{s_i})$ for the leaf node $i \in T$ (where s_i is a share of s corresponding to the leaf node i of the tree access structure T, see [12] for details)

(c) send $ACKM = \{T, C, (g^b)^s, \{C_i\}_{i\in T}\}$ to the user U.

(iii) The user U uses $ACKM$ to generate a convergent key CK and the file tag Tag_F.

(a) perform *CP-ABE* $Dec(SK_U, ACKM)$:

(1) compute $TK = e(g,g)^{a\bullet s} = e(g^{(a+t)/b}, g^{b\bullet s})/e(g,g)^{t\bullet s}$

$$\left(= \frac{e(g,g)^{(a+t)/b\bullet(b\bullet s)}}{e(g,g)^{t\bullet s}} = \frac{e(g,g)^{(a+t)\bullet s}}{e(g,g)^{t\bullet s}} = \frac{e(g,g)^{a\bullet s+t\bullet s}}{e(g,g)^{t\bullet s}} = \frac{e(g,g)^{a\bullet s}\bullet e(g,g)^{t\bullet s}}{e(g,g)^{t\bullet s}} \right) \quad (1)$$

where $e(g,g)^{t\bullet s} = \prod_{i\in T}\{e(g^t \bullet H(p_i)^{t_i}, g^{s_i})/e(g^{t_i}, H(p_i)^{s_i})\} = \prod_{i\in T}\{e(g,g)^{t\bullet s_i}\}$ (for details of the computation, see [12])

(2) compute $\sigma' = C/TK$

(3) compute $\sigma = \sigma' \bullet y^{-r} = h^x$

(b) verify that the signature of the AS is correct: $e(h, y) = e(\sigma, g)$

(c) if the signature verification passes, the convergent key $CK = \mathrm{kdf}(\sigma)$ and the file tag $Tag_F = Hash(CK)$ are computed (where the function kdf() is a cryptographically secure key derivation function).

3.2.3. The File Transfer Phase. Suppose that a user $U1$ requests the *CSP* to upload a file F. The *CSP* performs duplication check on the file F and approves the upload of the encrypted file if there is no duplication. If there is duplication (that is, if the same file is already stored), then perform the deduplication process. At this time, the PoW process is performed to confirm whether the user holds the file actually. If the verification is passed, the owner of the file stored in the storage is assigned to the user $U1$ without uploading the actual file.

(1) File Upload (see Figure 4)

(i) Note that the user $U1$ already has $\{Tag_F, r, TK, CK\}$ for the file F through the authorized convergent key generation phase.

(ii) The user $U1$ transmits the upload request message $\{ID_{U1}, Tag_F, T\}$ for the file F to the *CSP* using the file tag Tag_F and the access structure T.

(iii) The *CSP* uses the tag Tag_F and the access structure T to check whether a duplicate file is stored in the storage. If there is no duplication, perform the "First Upload" module. If the duplicate file exists, perform the "Deduplication" module.

(a) First Upload module (see Figure 5)

(i) The *CSP* requests the file transfer to the user $U1$.

(ii) The user $U1$ computes the ciphertext $CT = Enc(CK, F)$ by encrypting the file F using the convergent key CK, where $Enc()$ is an authenticated encryption mechanism such as AES in CTR mode [16]. Also, $U1$ computes $vTK = Hash(TK, CK)$ and $sig = Sign_{ssk}(CT, ACKM, y^r, vTK)$, where $Sign_{ssk}()$ is a signing algorithm using the user's private key ssk. Then $U1$ sends $\{CT, ACKM, y^r, vTK, sig\}$ to the *CSP*. The user $U1$ stores the file tag Tag_F and deletes F, CK, $ACKM$, and so on.

(iii) The *CSP* verifies the signature sig using the user's spk. If valid, store the file record $\{Tag_F, CT, ACKM, y^r, vTK, ID_{U1}\}$.

(b) Deduplication module (see Figure 6)

(i) If the file F already exists, the *CSP* has the file record $\{Tag_F, CT, ACKM, y^r, vTK, ID_{U0}\}$ from the old user $U0$.

(ii) Note that $U1$ has $\{Tag_F, r', TK', CK\}$ for the file F through the authorized convergent key generation phase such that $\{r', TK'\} \neq \{r, TK\}$.

(iii) For the deduplication procedure, the *CSP* first performs PoW protocol with $U1$ for the ciphertext $CT = Enc(CK, F)$. Within this protocol, the *CSP* sends vTK, $ACKM$, and y^r to $U1$. If the user passes PoW protocol, the *CSP* assigns the file reference to the user $U1$ and adds ID_{U1} to the file record.

(iv) Within the PoW protocol, the user $U1$ recovers TK using $ACKM$ and verifies his own access privileges as

$$CK = \mathrm{kdf}\left(\frac{C}{TK} \bullet \frac{1}{y^r}\right) \quad (2)$$

and $vTK == Hash(TK, CK)$.

If the verification passes, $U1$ keeps the file tag Tag_F and deletes all other data from its own storage.

(2) File Download (see Figure 7)

(i) The user $U1$ has attributes $A_{U1} = \{p_i\}$ and its corresponding privilege secret key $SK_{U1} = (D = g^{(a+t)/b}, \{D_j = g^t \bullet H(p_i)^{t_j}, g^{t_j}\}_{\forall j\in A_{U1}})$.

(ii) $U1$ sends the File download request message with $\{ID_{U1}, Tag_F\}$ to the *CSP*.

(iii) The *CSP* finds Tag_F in the file records and load it. The *CSP* checks whether ID_{U1} is in the user lists of the record. If exists, the *CSP* sends $\{CT, ACKM, y^r, vTK\}$ to the user $U1$.

(iv) $U1$ performs *CP-ABE Dec* on $ACKM$ with SK_{U1} as follows and obtains $CK = \mathrm{kdf}(\sigma)$:

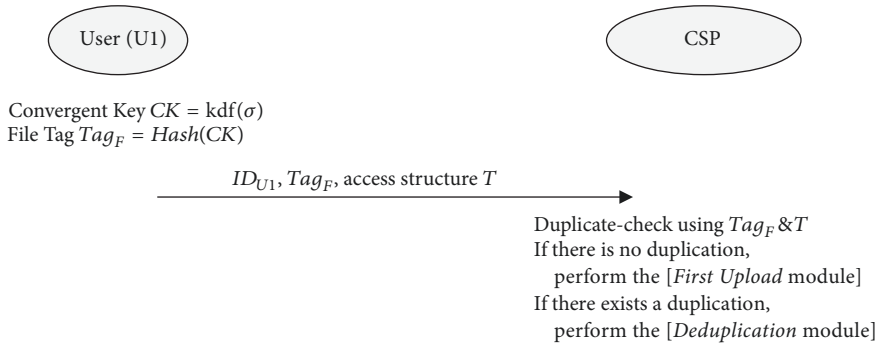


Figure 4: File upload process.

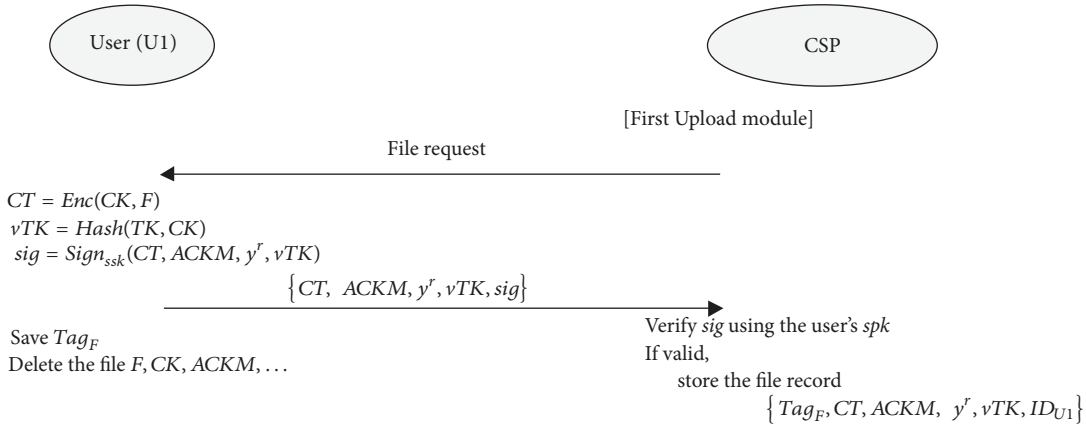


Figure 5: First upload module.

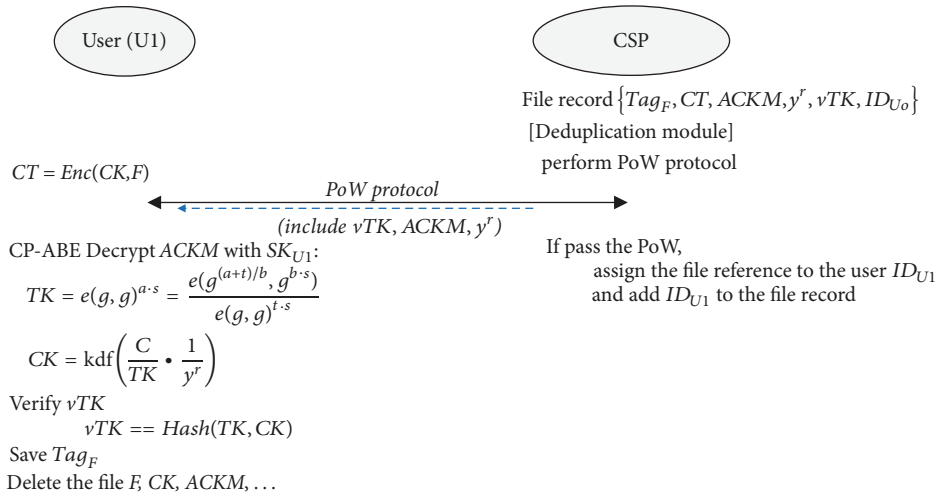


Figure 6: Deduplication module.

(a) compute $TK = e(g,g)^{a \bullet s} = e(g^{(a+t)/b}, g^{b \bullet s})/e(g,g)^{t \bullet s}$
(b) compute $\sigma' = C/TK$
(c) compute $\sigma = \sigma' \bullet y^{-r} = h^x$

(v) $U1$ verifies vTK. If pass, $U1$ obtains the file $F = Dec(CK, CT)$. Finally, verify the file as follows: $e(Hash(F), y) = e(\sigma, g)$.

4. Optimization and Discussions

4.1. Optimization. One problem to be considered is that, in the process of uploading duplicate files, the user with the duplicated file must be able to ensure that the convergent key can be correctly restored through the *ACKM* which is derived by the first uploader and is stored in the *CSP*. One possible solution is to verify the validity of the *ACKM* in the

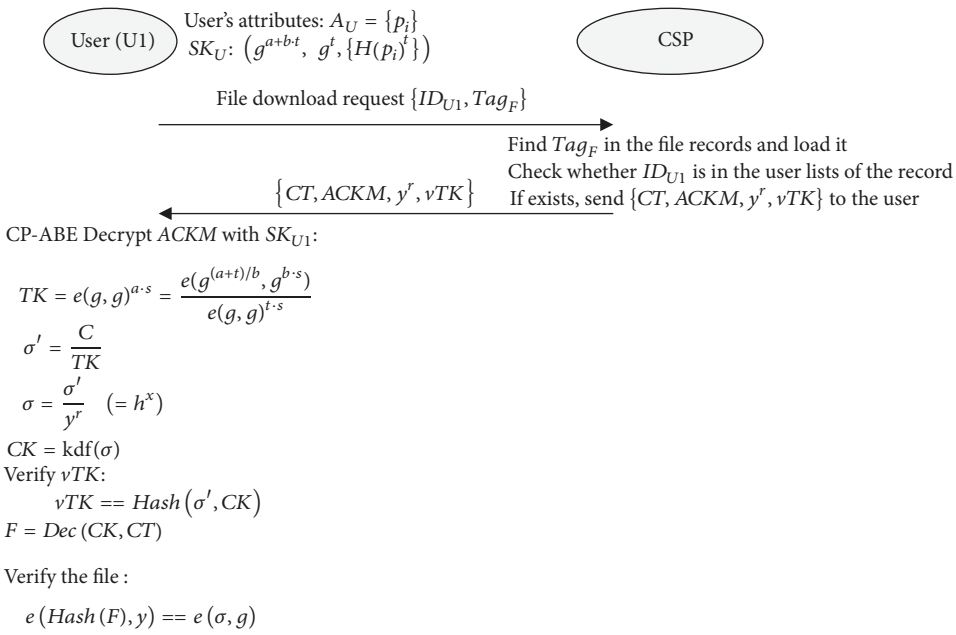


Figure 7: File download process.

PoW protocol because the PoW protocol must be performed in the deduplication process. We do not describe the specific PoW protocol in this paper, but as a simple solution to this purpose, we introduced the *vTK* value into the PoW protocol. The verification of the validity of the *ACKM* can be performed using the value of *vTK* in the PoW process. To do this, the *vTK* value must be stored together as an element of the *ACKM* in the first upload process. When the *CSP* delivers the *vTK* value to the user in the deduplication process, the user can verify that the value of $\sigma' = C/TK$ $(= C/e(g,g)^{a*s})$ derived from his own privilege information is valid. Then, it can be confirmed that the convergent key can be derived through the *ACKM* stored in the *CSP*. In order to support this, one hash computation overhead is added in the first upload process. In the *CSP*, there is additional storage overhead of *vTK* value. Finally, in the deduplication process, one hash computation overhead for the verification of *vTK* value and the cost for evaluating the convergent key are added.

Also, another consideration is the possibility of optimizing metadata for the duplicated file. If the file F with access structure T is already stored in the *CSP* with Tag_F, a user with a different access structure T' may request the same file F to be uploaded with Tag'_F $(== Tag_F)$. That is, if different access attributes are specified for the same file, there should be some consideration as to whether to deduplicate it. The file upload process considering this can be performed as follows: for the file upload request of the user, the *CSP* checks whether the corresponding file tag Tag'_F exists and performs "First upload module" if it does not exist (that is, if there is no duplication). If the corresponding file tag exists (in case of duplicate upload), the access structure T' is compared. It works differently depending on the inclusion relation between the existing access structure T and the newly requested access structure T'.

(i) In case of $T' \subseteq T$: In this case, since the access structure T' of the file to which the duplicate upload is requested is included in the access structure T of the already stored file, it is not necessary to upload it because it can be restored by the information of the file record already stored in the *CSP*. That is, the PoW protocol is executed within the deduplication module to confirm the retention of the file, and then the access rights to the file record stored in cloud are assigned without uploading the file.

(ii) In case of $T \subseteq T'$: In this case, since the access structure T of the already stored file is included in the access structure T' of the file requested to be uploaded, it is necessary to update the existing file record in order to save only one file in the CSP. To do this, the user uploads related data such as the ciphertext CT' computed with the access structure T'. The *CSP* replaces the information of the existing file record with the uploaded information. Existing users can restore the file through the updated file record.

(iii) If not for the above two cases: This is the case where the access structure T and T' do not satisfy the inclusion relation. In this case, the user uploads $\{ACKM', y^{r'}\}$, and the CSP adds this information to the existing file record corresponding to the file tag Tag_F. In this case, when the user requests to download the file, the *CSP* sends the appropriate $\{ACKM', y^{r'}\}$ corresponding to the requesting user. This allows the user to restore the file.

4.2. Security Analysis. Our scheme provides an authorized deduplication of encrypted data stored in cloud. The security of the proposed scheme depends on the security of the used cryptographic primitives such as hash functions, symmetric and asymmetric encryption algorithms. In the security of the proposed scheme, the *AS* plays an important role, and so it is assumed that the *AS* does not collude with an attacker.

(1) The Proposed Scheme Provides an Authorized Deduplication

Proof. The proposed scheme deduplicates the files with the same contents and the proper privileges. The same files always derive the same value σ $(= h^x$, where $h = H(F))$. However, σ is distributed to the user in encrypted form using CP-ABE with the access structure T by the *AS*. The file tag Tag_F for verifying duplication is derived from the convergent key CK which is computed from σ. So, when the users have the proper privilege the same convergent key CK is derived and therefore deduplication is possible. An unauthorized user cannot obtain the same convergent key CK because of the property of CP-ABE. To get CK, an attacker needs to get σ. To do this, it has to calculate CP-ABE decryption or know the *AS*'s private key x. If an attacker does not have the proper access privileges or know the *AS*'s private key x, he cannot compute CP-ABE $Dec()$ and obtain σ. Thus, our scheme provides an authorized deduplication under the assumption of the security of CP-ABE scheme and the private key.

When a new user $U1$ with privileges equal or higher than an old user $U0$ uploads the same file that is already stored in cloud, the CSP performs the deduplication process. The security of the proposed scheme against offline brute-force attack is guaranteed by the convergent key generation procedure as in [5]. In order for an attacker with lower permissions than an old user $U0$ to perform online brute-force attacks, it is necessary to forge the tag on the file. The unforgeability of the file tag is guaranteed by (2) below. Thus, the proposed scheme provides secure deduplication. □

(2) The Proposed Scheme Guarantees the Unforgeability of the File Tag

Proof. In the proposed scheme, the user generates the file tag $Tag_F = Hash(\mathrm{kdf}(\sigma))$ through the blind signature scheme with the *AS*. The user sends m $(= h \bullet g^r)$ to the *AS*, and the AS blindly signs it, $\sigma' = m^x$ with its own private key x. And then the *AS* encrypts by CP-ABE $Enc()$ with the access structure T. An unauthorized user has to obtain a valid σ in order to forge the file tag. This requires obtaining the *AS*'s private key or breaking the CP-ABE. Therefore, if the private key of the *AS* is kept secure and the CP-ABE algorithm is secure, the unforgeability of the file tag is guaranteed. □

(3) 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. An attacker who colludes with the *CSP* can access an encrypted data CT and metadata Tag_F. However, the proposed method can ensure security of a plaintext data even with a low entropy because CT has encrypted with a key CK derived by interacting with the *AS* through the authorized convergent key generation process. The convergent key CK is derived by a value to which the private key of the *AS* is applied, which is encrypted and distributed by CP-ABE algorithm with the user's privilege attributes. As long as an attacker does not know the AS's private key or cannot break CP-ABE algorithm, he cannot obtain CK and therefore the confidentiality of data is preserved if the encryption algorithm $Enc()$ is secure.

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 discrete logarithm problem to obtain the private information of the *AS*, and so it can ensure the security of the *AS*. When attacking an encrypted data stored in the CSP, an attacker has to first pass a proofs of ownership protocol for the file. To do this in the proposed scheme, an attacker has to construct a proof for the encrypted file in the PoW process. Thus it depends on the security of the PoW protocol used (for example the Merkle Tree-based proofs of ownership scheme presented in [13] can be used). That is, for an attacker who does not own the file, it provides the confidentiality of data stored in cloud. □

4.3. Discussions. Li et al. [8] first proposed the deduplication scheme applying the privilege information. In this scheme, the file authentication tags for the deduplication are computed by applying the privilege information and these tags are generated by the *AS*. The *AS* possesses the privilege private key corresponding to the user's privileges and this privilege private key does not be distributed to the user.

While Li et al. scheme supports only simple access attributes, the proposed scheme can support hierarchical access attributes, and so direct comparisons are difficult. Li et al. scheme [8] has the following shortcomings. First, each access privilege is represented by a private key. If a user has multiple access privileges, the *AS* needs to keep private keys securely as many as the number of access attributes, which may cause a great deal of trouble in the user key management in the *AS*. Also, when a user uploads a file F that has assigned n access privileges, the *AS* needs to generate n file tags for F and sends them to the user. Then, the user must send these tags to the *CSP* for the file uploading, so it causes large network traffic. In addition, Li et al. scheme has a disadvantage in terms of the number of keys to be stored in the cloud server when compared with the proposed scheme. For a duplicated file, Li et al. scheme has to store keys as many as the number of users. In the case of Li et al. scheme, each user U stores the encrypted convergent key eK in the cloud by encrypting the convergent key CK (i.e., $eK = Enc(sk_U, CK)$) using his secret key sk_U, so that the number of keys stored in the cloud is equal to the number of users. However, the proposed scheme only needs to store one encrypted key material, $ACKM$. The authorized user can restore the convergent key via $ACKM$. Meanwhile, Li et al. scheme has another disadvantage that the *AS* is always involved in the file upload process. This adds a lot of overhead to the *AS*. In the proposed scheme, the *AS* participates only in the convergent

Table 1: Comparison of the storage overhead on the user and the *CSP*.

	Li et al.' scheme [8]	Proposed scheme
User	$N_u * (2 * N_{ap} * \text{h\_len})$	$1 * (ep\_len * (N_{ap} + 2)) + N_u * (1 * \text{h\_len})$
CSP	$N_{csp} * \{N_{ap} * \text{h\_len} + 1 * \text{ct\_len} + N_d * (N_{ap} * \text{senc\_len})\}$	$N_{csp} * \{1 * \text{h\_len} + 1 * \text{ct\_len} + 1 * (ep\_len * (N_{ap} + 2)) + 1 * ep\_len + 1 * \text{h\_len}\}$

Notes. N_u: the number of files owned and uploaded by one user; N_{csp}: the number of files that are uploaded by all users and managed on CSP; N_d: the number of deduplicated file uploads; N_{ap}: the number of access privileges assigned to the file by the user; ct_len: the bit-length of the encrypted file; ep_len: the bit-length of the elliptic curve pairings; h_len: the bit-length of the output of the hash function; senc_len: the bit-length of the output of the symmetric encryption.

Table 2: The bit-length overhead on the user and the *CSP*.

	Li et al.' scheme [8]	Proposed scheme
User	512,000 bits	28,672 bits
CSP	115,360,000 bits (110.02 Mbits)	103,840,000 bits (99.03 Mbits)

key generation process and thereafter does not involve in the protocol. This reduces the processing burden of the *AS*.

Of course, the proposed scheme that supports more complex and hierarchical privileges has higher computational complexity than existing schemes. The application of CP-ABE scheme to the convergent key generation process has the advantages of higher security and the utilization of hierarchical privileges, but the computational overhead has increased. This computational overhead does not pose a problem for the practicality of the proposed scheme. According to [17], it is reported that the encryption and decryption take about 3.3 *ms* and 6.2 *ms*, respectively, in the case of the lightweight CP-ABE scheme (the number of user attributes is 600). Thus, in case of applying the lightweight CP-ABE scheme such as [18, 19], this amount of computing costs will not be a big problem for practical use.

We compare the storage complexity of the proposed scheme with the existing scheme. Table 1 shows the storage overhead on the user and the *CSP*.

In order to make the comparison result more clear, suppose we have chosen the following bit-length as key lengths of the cryptographic primitives to ensure 128-bit security (see [20]): 128-bit for the symmetric encryption algorithm, 256-bit for the hash function and 256-bit for the elliptic curve pairings. And let N_u=100, N_{csp}=1000, N_d=10, N_{ap}=10, and *ct_len*=100,000. Table 2 shows the specific storage overhead of both schemes.

On the user side, it can be seen that the storage complexity of the proposed scheme is reduced by 483,328 bits (That is, it can save about 94%). On the CSP, the proposed scheme is about 11,520,000 bits small, so it can save about 10%. These results show that the proposed scheme has advantages in terms of storage space complexity.

The advantages of the proposed scheme are further clarified by the results of the comparison below. Figure 8 shows the storage overhead on the CSP when only the N_d (the number of deduplicated file uploads) value is increased in the above comparison. Figure 9 also shows the storage overhead when the N_{ap} (the number of access privileges assigned to the file by the user) value increases. If the number of duplicate files or the number of access privileges increases, we can see that the proposed scheme is more advantageous than the scheme of [8].

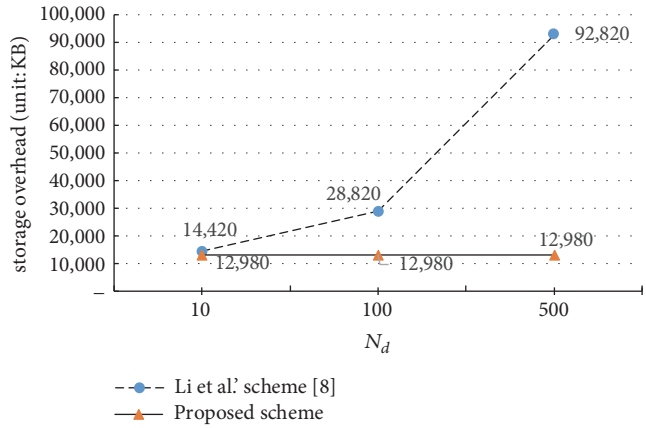
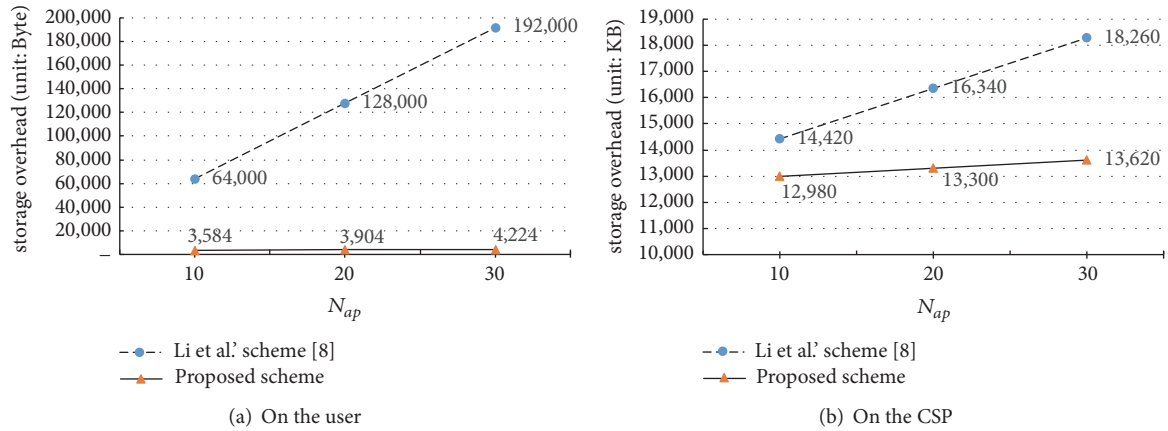


Figure 8: Storage overhead on the CSP as the N_d value changes.

The proposed CP-ABE-based authorized convergent encryption scheme can apply much more complex and various attribute privileges than existing schemes. Because it can utilize a hierarchical access structure that can be used in the CP-ABE techniques, it is very suitable for data sharing services in cloud storage. Since the convergent encryption key generated by the proposed scheme can be derived only by the authorized user, the leakage of the file information by the unauthorized user in the deduplication process can be blocked.

5. Conclusion

In this paper, we proposed the authorized deduplication scheme using CP-ABE. The proposed scheme provides client-side deduplication while providing confidentiality through client-side encryption to prevent exposure of users' sensitive data on untrusted cloud servers. Also, unlike existing convergent encryption schemes, it provides authorized convergent encryption by using CP-ABE to allow only authorized users to access critical data. The proposed CP-ABE-based authorized convergent encryption scheme can apply much more complex and various types of attribute privileges than existing schemes. Also, it satisfies security requirements. And the proposed scheme has advantages over existing scheme in



(a) On the user

(b) On the CSP

Figure 9: Storage overhead as the N_{ap} value changes.

terms of the AS's burden and storage overhead. The proposed scheme is very suitable for data sharing services in the enterprise's hybrid cloud storage model.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [18ZH1200, Core Technology Research on Trust Data Connectome].

References

[1] H. Cui, R. H. Deng, Y. Li, and G. Wu, "Attribute-based storage supporting secure deduplication of encrypted data in cloud," *IEEE Transactions on Big Data*, 2017, Early Access.

[2] P. Mell, J. Shook, R. Harang, and S. Gavrila, "Linear time algorithms to restrict insider access using multi-policy access control systems," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 8, no. 1, pp. 4–25, 2017.

[3] 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 *Proceedings of the 22nd International Conference on Distributed Systems (ICDCS 2002)*, pp. 617–624, IEEE, Vienna, Austria, 2002.

[4] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 40–47, 2010.

[5] S. Keelveedhi, M. Bellare, and T. Ristenpart, "Dupless: Server-aided encryption for deduplicated storage," in *Proceedings of the 22nd USENIX Security Symposium (USENIX Security 13)*, vol. 12, pp. 179–194, USENIX, Washington, DC, USA, 2013.

[6] Y. Duan, "Distributed key generation for encrypted deduplication: Achieving the strongest privacy," in *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security (CCSW'14)*, pp. 57–68, ACM, Scottsdale, Arizona, USA, 2014.

[7] M. Miao, J. Wang, H. Li, and X. Chen, "Secure multi-server-aided data deduplication in cloud computing," *Pervasive and Mobile Computing*, vol. 24, pp. 129–137, 2015.

[8] J. Li, Y. K. Li, X. Chen, P. P. C. Lee, and W. Lou, "A hybrid cloud approach for secure authorized deduplication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1206–1216, 2015.

[9] T.-Y. Youn, K.-Y. Chang, K. H. Rhee, and S. U. Shin, "Authorized client-side deduplication using access policy-based convergent encryption," *Journal of Internet Technology*, vol. 19, no. 4, pp. 1229–1240, 2018.

[10] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (Advances in Cryptology – EUROCRYPT 2005*, vol. 3494, pp. 457–473, Springer-Verlag, LNCS, Aarhus, Denmark, 2005.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 89–98, ACM, Alexandria, Virginia, USA, November 2006.

[12] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*, pp. 321–334, IEEE, Berkeley, CA, USA, 2007.

[13] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS'11*, pp. 491–500, ACM Press, 2011.

[14] T. Youn, K. Chang, K. Rhee, and S. U. Shin, "Efficient client-side deduplication of encrypted data with public auditing in cloud storage," *IEEE Access*, vol. 6, pp. 26578–26587, 2018.

[15] C. F. Kerry, A. Secretary, and C. R. Director, "FIPS PUB 186-4 federal information processing standards publication digital signature standard (DSS)," *National Institute of Standards and Technology (NIST)*, 2013.

[16] M. Dworkin, *Recommendation for Block Cipher Modes of Operation Methods and Techniques*, NIST, USA, 2001, NIST-SP-800-38A.

[17] V. Odelu, A. K. Das, M. Khurram Khan, K.-K. R. Choo, and M. Jo, "Expressive CP-ABE scheme for mobile devices in IoT satisfying constant-size keys and ciphertexts," *IEEE Access*, vol. 5, pp. 3273–3283, 2017.

[18] Y. Zhang, D. Zheng, and X. Chen, "Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts," in *Provable Security*, vol. 8782, pp. 259–273, Springer, 2014.

[19] H. Tsuchida, T. Nishide, and E. Okamoto, "Expressive ciphertext-policy attribute-based encryption with fast decryption," *Journal of Internet Services and Information Security (JISIS*, vol. 8, no. 4, pp. 37–56, 2018.

[20] E. Barker and A. Roginsky, "Transitioning the use of cryptographic algorithms and key lengths," *Draft NIST Special Publication 800-131A Revision 2, National Institute of Standards and Technology (NIST)*, 2018.