This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

Heterogeneous Data Storage Management with Deduplication in Cloud Computing

Zheng Yan, *Senior Member, IEEE,* Lifang Zhang, Wenxiu Ding, and Qinghua Zheng, *Member, IEEE*

Abstract—Cloud storage as one of the most important services of cloud computing helps cloud users break the bottleneck of restricted resources and expand their storage without upgrading their devices. In order to guarantee the security and privacy of cloud users, data are always outsourced in an encrypted form. However, encrypted data could incur much waste of cloud storage and complicate data sharing among authorized users. We are still facing challenges on encrypted data storage and management with deduplication. Traditional deduplication schemes always focus on specific application scenarios, in which the deduplication is completely controlled by either data owners or cloud servers. They cannot flexibly satisfy various demands of data owners according to the level of data sensitivity. In this paper, we propose a heterogeneous data storage management scheme, which flexibly offers both deduplication management and access control at the same time across multiple Cloud Service Providers (CSPs). We evaluate its performance with security analysis, comparison and implementation. The results show its security, effectiveness and efficiency towards potential practical usage.

Index Terms—Data Deduplication, Cloud Computing, Access Control, Storage Management

1 Introduction

Cloud computing allows centralized data storage and online access to computer services or resources. It offers a new way of Information Technology (IT) services by re-arranging various resources and providing them to users based on their demands. Cloud computing has greatly enriched pervasive services and become a promising service platform due to a number of desirable properties [40, 41], such as scalability, elasticity, fault-tolerance, and pay-per-use.

Data storage service is one of the most widely consumed cloud services. Cloud users have greatly benefited from cloud storage since they can store huge volume of data without upgrading their devices and access them at any time and in any place. However, cloud data storage offered by Cloud Service Providers (CSPs) still incurs some problems.

First of all, various data stored at the cloud may request different ways of protection due to different data sensitivity. The data stored at the cloud include sensitive personal information, publicly shared data, data shared within a group, and so on. Obviously, crucial data should be protected at the cloud to prevent from any access of unauthorized parties. Some unimportant data, however, have no such a requirement. As outsourced data could disclose personal or even sensitive information, data owners sometimes would like to control their data by themselves, while on some occasion, they prefer to delegate their control to a third party since they cannot be always online or have no idea how to perform such a control. How to make cloud data access control adapt to various scenarios and satisfy different user demands becomes a practically important issue. Access control on encrypted data has been widely studied in the literature [10-17, 33]. However, few of them can flexibly support various requirements on cloud data protection in a uniform way, especially with economic deduplication management.

Second, flexible cloud data deduplication with data access control is still an open issue. Duplicated data could be stored at the cloud [39] in an encrypted form by the same or different users, in the same or different CSPs. From the standpoint of compatibility, it is highly expected that data deduplication can cooperate well with data access control. That is the same data (either encrypted or not) are only stored once at the cloud, but can be accessed by different users based on the policies of data owners or data holders (i.e., the eligible data users who hold original data). Although cloud storage space is huge, duplicated data storage could greatly waste networking resources, consume plenty of power energy, increase operation costs, and make data management complicated. Economic storage will greatly benefit CSPs by decreasing their operation costs and reversely benefit cloud users with reduced service fees. Obviously, cloud data deduplication is particularly significant for big data storage and management. However, the literature still lacks studies on flexible cloud data deduplication across multiple CSPs. Existing work cannot offer a generic solution to support both deduplication and access control in a flexible and uniform way over the cloud [18, 22-24, 29-38].

- *Z. Yan is with the State Key Laboratory of Integrated Services Networks, School of Cyber Engineering, Xidian University, POX 91, No. 2 South Taibai Road, 710071, Xi'an, China, and also with the Department of Communications and Networking, Aalto University, Otakaari 5, 02150, Espoo, Finland (e-mail: zyan@xidian.edu.cn).*
- *L. Zhang is with the Department of Communications and Networking, Aalto University, Otakaari 5, 02150, Espoo, Finland (e-mail: lifang.zhang@aalto.fi).*
- *W. Ding is with the School of Cyber Engineering, Xidian University, No. 2 South Taibai Road, 710071, Xi'an, China (Email: wenxiuding_1989@126.com).*
- *Q. Zheng is with the Xi'an Jiaotong University, Xi'an, China (Email: qhzheng@xjtu.edu.cn)*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

In this paper, we propose a holistic and heterogeneous data storage management scheme in order to solve the above problems. The proposed scheme is compatible with the access control scheme proposed in [33]. It further realizes flexible cloud storage management with both data deduplication and access control that can be operated by either the data owner or a trusted third party or both or none of them. Moreover, the proposed scheme can satisfy miscellaneous data security demands and at the same time save storage spaces with deduplication across multiple CSPs. Thus it can fit into various data storage scenarios. Our scheme is original and different from the existing work. It is a generic scheme to realize encrypted cloud data deduplication with access control, which supports the cooperation between multiple CSPs. Specifically, the contributions of this paper are:

- We motivate to save cloud storage across multiple CSPs and preserve data security and privacy by managing encrypted data storage with deduplication in various situations.
- We propose a heterogeneous data management scheme to support both deduplication and access control according to the demands of data owners, which can adapt to different application scenarios. Our scheme can support data sharing among eligible users in a flexible way, which can be controlled by either the data owners or other trusted parties or both of them.
- We justify the performance of the proposed scheme through security analysis, comparison with existing work and implementation based performance evaluation. The results show its security, advantages, efficiency and potential applicability.

The rest of the paper is organized as below. We give a brief review on related work in Section 2. In Section 3, we present a system and security model, and introduce notations and preliminaries that are used in our scheme. We present the detailed design of the proposed scheme in Section 4, followed by security analysis, comparison with existing work and performance evaluation in Section 5. Finally, the last section concludes the paper.

2 Related Work

2.1 Access Control on Encrypted Data

Existing researches [1-3] proposed to encrypt data before outsourcing it to the cloud in order to prevent data privacy from being invaded at CSP. Access control on encrypted data requests that only authorized entities can decrypt the encrypted data. An ideal approach is to encrypt each data once and issue relevant keys to authorized entities only once. However, due to the changeability of trust relationships, key management becomes complicated due to frequent key update.

Access Control Lists (ACLs) were applied to ensure data security in a distrusted or semi-trusted party (e.g., CSP). Before uploading data to CSP, the data owner first classifies the data into different groups, and then encrypts each group with a symmetric key, which is only distributed to the users in the ACL of the group. In this way, this group of data is only accessible by the users in the ACL [4]. The shortcoming of this scheme mainly comes from the fact that the number of symmetric keys increases linearly with the number of groups. Moreover, the trust relationship change between one individual user and the data owner could cause essential update of relevant symmetric keys, which impacts other users in the same ACL. Thereby, this approach is impractical to be applied in many real applications where the trust relationship between different users changes frequently. Combining a traditional symmetric cryptosystem and an asymmetric cryptographic system was proposed for cloud data access control [5]. However, the computation cost of key encryption increases linearly with the number of users in the ACL.

Attribute-Based Encryption (ABE) [6-9] was proposed to achieve access control on encrypted cloud data. It specifies a set of attributes to identify users and encrypts data based on an access structure specified by attributes. Thus, encrypted data can only be decrypted by the users that hold such attributes that can satisfy the access structure. ABE is classified into two divisions: key-policy ABE (KP-ABE) [7] and ciphertext-policy ABE (CP-ABE) [6, 8] according to how the attributes link to ciphertexts and decryption keys. ABE has such advantages as scalability and high flexibility in terms of attributes based access policies and fine-grained access control. It has been widely applied to secure cloud data storage in recent years [10-17]. However, all above existing solutions about access control on encrypted data did not consider how to solve the issue of duplicated data storage in cloud computing in a holistic and comprehensive manner, especially for encrypted data in various data storage scenarios. This issue is practically significant for big data secure storage over the cloud.

2.2 Encrypted Data Deduplication

It is a hot research topic to reconcile deduplication and client-side encryption [18]. Existing industrial solutions fail to perform deduplication on encrypted data, e.g., Dropbox [19], Google Drive [20], and Mozy [21]. Message-Locked Encryption (MLE) was proposed to resolve this tension [22]. Convergent Encryption (CE), the most prominent manifestation of MLE, was introduced [23, 24]. In CE, a user computes the key of data M based on its hash code $K \leftarrow H(M)$ and encrypts M with K. Another user holding the same data can produce the same encrypted data, thus realizing deduplication. The CE suffers from offline brute-force dictionary attacks. As a result, CE can ensure high security only when the underlying data is drawn from a large space that is too big to exhaust. In addition, CE cannot support data access controlled by data owners, as well as other authorized parties. It is hard to support data revocation because generating a same new encryption key is hard to achieve for both the data owners and the data holders to re-encrypt the data.

A number of schemes were proposed to overcome the weakness of CE. Bellare et al. proposed DupLESS to resist the above-mentioned brute-force attacks [18]. In DupLESS, users encrypt their data using the keys obtained from a Key Server (KS). They are generated based on the data with an oblivious Pseudo Random Function (PRF) protocol. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

KS is separated from a Storage Service (SS). Users authenticate themselves to the KS without leaking any information about their data. Thus, high security can be assured if the KS is not accessible to attackers. Even though both KS and SS are compromised, DupLESS can still preserve the security of stored data based on the guarantee of MLE. But some data owners do not like to authorize a third party like KS to control their data, since in some specific situations they prefer to manage the storage and access of their data by themselves and keep track of data storage and usage status. However, DupLESS cannot support this desirable feature. Li et al. presented an efficient and reliable convergent key management scheme by splitting a convergent key and distributing its shares among multiple servers [35]. However, it still cannot avoid the innate drawbacks of CE. Wen et al. constructed a session-key-based convergent key management scheme and a convergent key sharing scheme to solve the issue that encrypted data blocks and data ownership are frequently changed [36]. But this work requests all data owners communicate with each other to manage their session key. The problem of CE still exists. Liu et al. proposed a secure cross-user deduplication scheme that supports client-side encryption without requiring any additional independent servers by applying a password authenticated key exchange protocol [38]. But this scheme requests that the data owner is always online for data ownership check and deduplication. Thus this approach cannot handle the situation that the data owner is not available, which is very common in practice. Cross-CSP was not discussed in this work. The above schemes cannot flexibly manage data deduplication in various situations and across multiple CSPs. They cannot solve the issues as described in the introduction. Neither can they support the management of digital rights.

Existing schemes realized deduplication in either server-side or owner-side. Seldom, a hybrid solution was proposed to gain advantages of both approaches. In [29], the authors proposed a method to solve deduplication controlled by data owner only. The access control of other data holders is based on predefined metadata that describes eligible users and is shared with CSP. Applying public key encryption in this method results in high computation complexity, which is linearly increased with the number of users and lacks flexibility to support various data storage scenarios. Hur et al. proposed a novel server-side deduplication scheme for encrypted data [34]. It allows the cloud server to control access to outsourced data even when the ownership changes dynamically by exploiting randomized convergent encryption and secure ownership group key distribution. This scheme prevents data leakage not only to revoked users but also to an honest-but-curious cloud storage server.

Yan et al. [30, 31] proposed a deduplication scheme based on PRE, but it completely relied on an authorized party to control data deduplication. It cannot flexibly adapt to different scenarios, especially the data access controlled by the data holders. In another line of our previous work [32], we applied ABE to realize deduplicated data access control managed by data owners. Similarly, this scheme cannot solve the issue about flexible data access control with deduplication across multiple CSPs, which can be managed by any trusted parties based on real application demands.

2.3 Other Related Work

Yang et al. proposed a scheme called Provable Ownership of the File (POF) [25], which allows a user to prove to a server that it really possesses a file without the need to upload the entire file. Data ownership proof is an essential process of data deduplication, especially for encrypted data. But this scheme does not consider flexible deduplication control across multiple CSPs. Yuan and Yu proposed a scheme to achieve data deduplication and secure data integrity auditing at the same time [28]. It supports both public and batch auditing. This work applied different technologies (i.e., polynomial-based authentication tags and homomorphic linear authenticators) from ours and focused on solving a different research issue. Wu et al. developed Index Name Servers (INS) to reduce the workload caused by duplicated data. But this work cannot support the deduplication on encrypted data. A hybrid data deduplication mechanism was proposed by Fan et al. [27]. It can deduplicate both plaintext and ciphertext. However, this mechanism has such a drawback that CSP knows the key that is used for data encryption. Therefore, it cannot be applied into such a situation that the CSP cannot be fully trusted by data owners. Li et al. formally addressed the problem of authorized data deduplication [37]. Different from traditional deduplication systems, the differential privileges of users are further considered in duplicate check besides the data itself in a hybrid cloud architecture. All above work focused on solving different research issues from ours.

3 Problem Statement

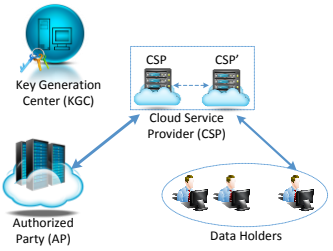


Fig. 1. System model

3.1 System and Security Model

Figure 1 shows the system that the proposed scheme can be applied into. It contains four types of entities:

1) Key Generation Center (KGC) that is fully trusted and responsible for system parameter generation and certification issuance.

2) The Cloud Service Provider (CSP) that offers a data storage service. Multiple CSPs could exist in the system. Thus, a cloud user can choose one of them to manage its uploaded data and seek advanced usage experiences. In addition, CSPs can cooperate with each other under a business agreement to save storage spaces through deduplication;

3) The Data Owner or the Data Holder that uploads and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

stores data at CSPs. Different CSPs may serve the data holders. Multiple eligible data holders or a single cloud user could store the same encrypted or plain data at one CSP or across CSPs;

4) The Authorized Party (AP) that is responsible for controlling data access as a delegate of data owners as they expect to support deduplication.

In this system, AP is trusted by all entities. All CSPs cannot be fully trusted. That is, they are curious about the raw data of cloud users but follow system design and protocols strictly. We hold such an assumption that the AP would never collude with the CSPs due to different business incentive and interests. Any collusion would worsen the reputation of the CSPs, which lead to final loss of their business.

We additionally hold following assumptions. The data holder provides the correct hash code set of its data for data ownership verification. The first eligible data holder that uploads the data is regarded as the data owner. Multiple APs could exist in the system and can be supported by the underlying scheme. For simplification, we assume one AP in the system for easy presentation. CSP, AP and data owners/holders use secure channels to communicate with each other. Backup of stored data is generally performed by CSP and this kind of data duplication for erasing storage risk is out of the discussion of this paper. Delegation agreement could be negotiated and signed among data holders during ownership check for data access management. If a data holder does not want any delegation, the procedure of our scheme will go to this data holder about data access, which means the access will be jointly controlled online by this data holder and the data owner. For simplifying system process, we assume delegation can be agreed among all data holders.

3.2 Notations and Preliminaries

1) Notations

Table 1 summarizes the notations used in this paper.

TABLE 1. NOTATIONS

Key	Description	Usage
PK_u	The public key of u about ABE;	The unique ID of user u and the key for user attribute verification; it is used to generate personalized secret attribute key for u.
SK_u	The secret key of u about ABE;	For decryption in ABE.
PK'_u	The public key of u about Public-Key Cryptosystem (PKC);	For PKC encryption and signature verification.
SK'_u	The secret key of u about PKC;	For PKC decryption and signature generation.
DEK_u	The symmetric key of u;	User data encryption.
$DEK_{1,u}$	The partial key 1 of DEK_u;	
$DEK_{2,u}$	The partial key 2 of DEK_u;	
$pk_{ID,u}$	The public key of u regarding attribute ID;	For encrypting $DEK_{2,u}$.
$sk_{ID,u,u'}$	The secret key of u' regarding attribute ID issued by u;	$DEK_{2,u}$.
DEK'_u	The renewed symmetric key of u;	
$H(*)$	The hash function;	
CT_u	The ciphertext of u;	
CK_u	The cipherkey of u;	
pk_u	The public key of u about PRE;	Generation of re-encryption key for u.
sk_u	The secret key of u about PRE;	Decryption in PRE.
M	The duplicated data;	
$HC(M)$	The hash code set of data M.	

2) Proxy Re-Encryption (PRE)

PRE transforms the ciphertext of m encrypted with the public key of entity A into one that can be decrypted with the private key of entity B at a proxy.

$\boldsymbol{E(pk_A, m)}$ outputs ciphertext $Cipher_A = E(pk_A, m)$ by taking input pk_A and data m.

$\boldsymbol{RG(sk_A, pk_B)}$, the re-encryption key generation algorithm outputs re-encryption key $rk_{A\to B}$ for a proxy (e.g., CSP) by taking input (sk_A, pk_B).

$\boldsymbol{R(rk_{A\to B}, Cipher_A)}$, the re-encryption algorithm outputs $R(rk_{A\to B}, Cipher_A) = E(pk_B, m) = Cipher_B$ by taking input $rk_{A\to B}$ and $Cipher_A$. $Cipher_B$ can be decrypted with sk_B.

$\boldsymbol{D(sk_B, Cipher_B)}$ outputs plain data m by taking input sk_B and $Cipher_B$.

Each user has a key pair for PRE, which is applied when AP involves in taking charge of data deduplication and access control. PRE allows AP to grant data access right to an eligible user through re-encryption at CSP, while the plain data cannot be gained by the CSP.

3) Attribute-Based Encryption

We also resort to control data access during deduplication based on user identity by applying ABE [6-9]. The advance of adopting ABE is a data owner only encrypts a data encryption key once when it issues access rights to a number of eligible data holders. Because the issued decryption keys are personalized for the data holders, they cannot collude with each other. We can also easily realize fine-grained access control with ABE, which further enhances the flexibility of our scheme. We can use either CP-ABE to simplify key management or KP-ABE to gain the efficiency of data encryption. In our implementation as described in Section 5.3, we applied CP-ABE to demonstrate the scheme. In the proposed scheme, each user maintains a secret key SK_u about ABE. SK_u and the identities of other users are used to generate the ABE decryption key of the users based on attribute ID, named as secret attribute key. ID denotes the identity attribute, which can be an anonymous identifier of the user. $pk_{ID,u}$ is the public key used to encrypt a partial key of DEK_u. Data owner u issues a personalized secret attribute key $sk_{ID,u,u'}$ to eligible data holder u' through a secure channel for decrypting the part of cipher-key encrypted with $pk_{ID,u}$.

Our scheme is heterogonous and flexible. In some scenarios, data owners would like to directly control data deduplication, e.g., in the case that they know the data

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

holders, which is also the scenario that was referred by the work in [36]. The scheme proposed in our paper is more advanced than existing work because it can adapt to various application scenarios. For example, the data owner can manage deduplication directly or it does not know how to manage it thus delegates this task to a third party, or it would like to perform dual control or no control. All above scenarios can be supported by our scheme.

Notably, our scheme is a framework that can adapt to various user policies on data deduplication. The data owner adopts the ABE algorithm to directly manage its data deduplication and sharing. In different scenarios, the access policy would differ from each other, which is based on the data sensitivity and the willingness of the data owner. For simplicity, we directly regard user identity as a basic attribute in ABE and data owners are responsible for the ABE setup and key management. If higher security is required and fine-grained access control is expected, more complicated access policy can be designed and this can be realized based on the properties of ABE.

4 System Design

4.1 Overview

We propose a scheme for heterogeneous data storage management with deduplication. It can be flexibly applied into such scenarios that cloud data deduplication is handled 1) only by the data owner; 2) by any trusted third party; 3) by both the data owner and the trusted third party; 4) by nobody (i.e., plain data is stored at the cloud); 5) by either the data owner or the trusted third party.

Concretely, we use the hash code of data M to check data duplication during data storage at the cloud. The data holder signs the hash code of the data for passing the originality verification of CSP. Meanwhile, a number of hash codes of randomly selected specific parts of the data are calculated with their indexes (e.g., the hash code of the first 15.1% of M, the hash code of 21-25% of M). We call these hash codes as the hash code set ($HC(M)$) of data M.

When the data owner/holder stores M at CSP, it sends the signed hash code of M to CSP for duplication check. If there is no duplicated data stored at CSP, the data owner encrypts M with a randomly generated symmetric key DEK to get encrypted data CT. It separates DEK into two parts DEK_1 and DEK_2. It encrypts DEK_1 with pk_{AP} by applying PRE to get CK_1 and encrypts DEK_2 with ABE by using pk_{ID} to get CK_2. The encrypted two parts of DEK are passed to CSP together with CT.

If the above duplication check is positive, CSP further verifies the ownership of the data holder by challenging the hash code set of M, concretely some specific hash codes. If the ownership verification is positive, CSP contacts the data owner and/or AP for deduplication.

During deduplication, the data owner issues a personalized secret key through a secure communication channel (e.g., public key cryptosystem) to a data holder for decrypting CK_2 if eligibility verification is positive (i.e., the data holder is allowed by the data owner to store data M at CSP). Meanwhile, AP issues CSP a re-encryption key that is used to re-encrypt CT_1 to make it decryptable by the duplicated data holder in order to get DEK_1. By getting both DEK_1 and DEK_2, the duplicated data holder can gain DEK and access CT at CSP. Data duplication check and data deduplication can be performed among CSPs based on their agreement. One CSP can store data for other CSPs. Duplicated data access from the eligible users of other CSPs can be supported among the CSPs.

Depending on the data management policy set by the data owner, DEK can be randomly divided into multiple parts, which are taken care by different authorized parties (e.g., multiple APs). For simplifying presentation, we illustrate our scheme by dividing DEK into two parts: DEK_1 and DEK_2. The following use cases can be flexibly supported: 1) when DEK_1 is null and $DEK_2 = DEK$, the data owner solely controls data deduplication; 2) when $DEK_1 = DEK$ and $DEK_2 = null$, data deduplication is only controlled by AP; 3) when $DEK_1 \neq null, DEK_2 \neq null$ and $DEK_1 \parallel DEK_2 = DEK$, data deduplication is controlled by both AP and the data owner; 4) when $DEK_1 = DEK_2 = DEK$, data deduplication is managed by either AP or the data owner; 5) when $DEK_1 = DEK_2 = DEK = null$, plaintext is stored at CSP that handles deduplication without any specific control indicated by the data owner.

4.2 Fundamental Algorithms

In this sub-section, we introduce a number of fundamental algorithms of the proposed scheme.

1) System Setup

$\boldsymbol{InitiateSystem.}$ This algorithm is conducted at the KGC. It generates basic system parameters related to ABE and PRE, such as generators and universal attributes, etc.

$\boldsymbol{InitiateNode(u).}$ Based on the system parameters, cloud user u generates its own key pairs including ABE master key pair PK_u and SK_u used for ABE encryption and user decryption key issuance, PKC key pair PK'_u and SK'_u for signing, as well as pk_u and sk_u regarding PRE.

$\boldsymbol{SetupNode(u)}$. With node identity u and public keys as input, this algorithm conducted at KGC outputs a number of user credentials, $Cert(PK_u)$, $Cert(PK'_u)$ and $Cert(pk_u)$, which can be verified by CSPs and their users.

$\boldsymbol{InitiateAP}$. AP initiates itself by generating pk_{AP} and sk_{AP}. pk_{AP} is broadcast to the users of CSPs.

2) ABE Key Generation

$\boldsymbol{CreateIDPK(ID, SK_u)}$. This algorithm checks the policies about ID and outputs $pk_{ID,u}$ for user u to allow u to control its data deduplication and access.

$\boldsymbol{IssueIDSK(ID, SK_u, PK_{u'})}$. This algorithm is run by u to issue $sk_{ID,u,u'}$ to u' if the eligibility check of u' is positive. Otherwise, it outputs $NULL$. Specifically, user u checks the attributes of u'. If they satisfy with the policy defined by u, u issues a secret key to u' for sharing the duplicated data storage and allow its future access. Otherwise, it rejects the request.

For simplifying our presentation, we set user identity as an example attribute rather than complex attributes herein. The access control based on user identity also consists with practice since most of data access over the cloud is based on user identity. Data owner u allows other data holders

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

with $ID = PK_{u'_j}$ $(j = 1,2,3)$ to share its data storage. It encrypts DEK_2 with policy λ: $ID = PK_{u'_1} \vee PK_{u'_2} \vee PK_{u'_3}$. The encryption key algorithm *EncryptKey* as described below iterates over all $j = 1,2,3$, generates a random value for each conjunction and constructs CK_{2_j}. The cipher-key CK_2 is obtained as tuple $CK_2 = < CK_{2_1}, CK_{2_2}, CK_{2_3} >$.

3) Data Encryption and Decryption

$\boldsymbol{Encrypt(DEK_u, M)}$ encrypts M with DEK_u and outputs ciphertext CT_u to protect M stored at CSP.

$\boldsymbol{Decrypt(DEK_u, CT_u)}$ decrypts CT_u with DEK_u and outputs M. It is executed at the data holders to obtain the plain content of CT_u stored at CSP.

4) Symmetric Key Management

$\boldsymbol{SeparateKey(DEK_u)}$. On input DEK_u, this algorithm outputs a number of partial keys, e.g., $DEK_{1,u}$ and $DEK_{2,u}$ based on random separation. Separating DEK_u into multiple parts can also be performed if needed.

$\boldsymbol{CombineKey(DEK_{1,u}, DEK_{2,u})}$. On input partial keys of DEK_u, e.g., $DEK_{1,u}$ and $DEK_{2,u}$, this algorithm outputs the full key DEK_u through combination.

5) Partial Key Control based on ABE Operated by Data Owner

$\boldsymbol{EncryptKey(DEK_{2,u}, \lambda, pk_{ID,u})}$ encrypts $DEK_{2,u}$ with policy λ and outputs cipher-key $CK_{2,u}$ by taking $DEK_{2,u}$, λ and $pk_{ID,u}$ as input. This algorithm is conducted at u.

$\boldsymbol{DecryptKey(CK_{2,u}, \lambda, SK_{u'}, sk_{ID,u,u'})}$ decrypts cipher-key $CK_{2,u}$ and outputs $DEK_{2,u}$ if the policy λ under which $DEK_{2,u}$ was encrypted can be satisfied; otherwise it outputs NULL. This algorithm is conducted at u'.

6) Partial Key Control based on PRE Operated by AP

We employ PRE to enable AP to perform the re-encryption of CK_1. During ciphertext re-encryption, CSP learns nothing about DEK_1. The algorithms related to PRE are represented as below:

$\boldsymbol{E(pk_{AP}, DEK_{1,u})}$ outputs $CK_1 = E(pk_{AP}, DEK_{1,u})$ by taking pk_{AP} and $DEK_{1,u}$, as input.

$\boldsymbol{RG(pk_{AP}, sk_{AP}, pk_{u'})}$ outputs re-encryption key $rk_{AP \to u'}$ for the proxy CSP by taking pk_{AP}, sk_{AP}, and $pk_{u'}$ as input.

$\boldsymbol{R(rk_{AP \to u'}, CK_1)}$ takes input $rk_{AP \to u'}$ and CK_1, and outputs $R(rk_{AP \to u'}, CK_1) = E(pk_{u'}, DEK_{1,u}) = CK'_1$, which can be decrypted with $sk_{u'}$.

$\boldsymbol{D(sk_u, CK'_1)}$ outputs $DEK_{1,u}$ by taking sk_u and CK'_1 as input.

4.3 Flexible Deduplication Scheme

1) Data Deduplication

Figure 2 shows the procedure of data deduplication with heterogeneous control handled by both the data owner and AP. User u_1 is the data owner that stores data M at CSP by encrypting it with DEK_{u_1}, while user u_2 tries to store the same data at CSP. We assume that both the data owner and AP are indicated for deduplication control based on the encryption behavior of u_1. Both u_1 and u_2 are the users of the same CSP.

Step 1 - System Setup: After system parameter generation, each node u_i calls *InitiateNode* to generate three key pairs PK_{u_i} and SK_{u_i}; PK'_{u_i} and SK'_{u_i}; pk_{u_i} and sk_{u_i} $(i = 1, 2, \ldots)$. Meanwhile, u_i gets the certificates of public keys $Cert(PK_{u_i})$, $Cert(PK'_{u_i})$ and $Cert(pk_{u_i})$ from KGC. AP calls *InitiateAP* to generate its key pair pk_{AP} and sk_{AP}.

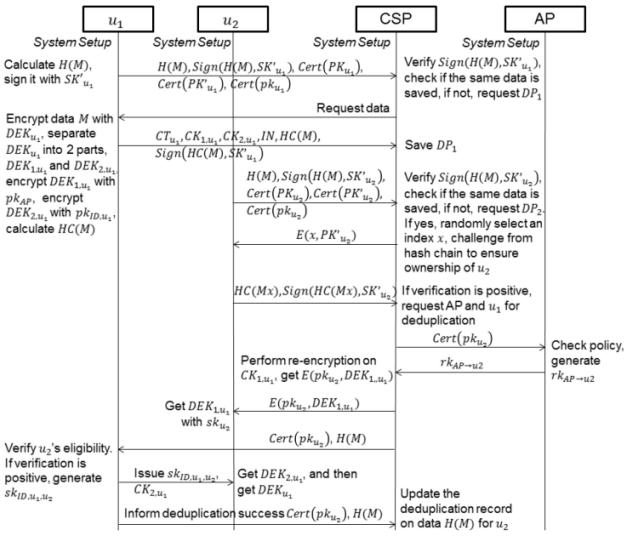


Fig. 2. Data deduplication with heterogeneous control

Step 2 - Duplication Check: User u_1 stores data M at CSP. It calculates $H(M)$, signs $H(M)$ with SK'_{u_1} and sends package $P_1 = \{H(M), Sign(H(M), SK'_{u_1}), Cert(PK_{u_1}), Cert(PK'_{u_1}), Cert(pk_{u_1})\}$ to CSP. CSP checks if the same data has been stored already by verifying the signature and checking if $H(M)$ has existed. The duplication check across multiple CSPs can be supported, refer to next sub-section for details. If the check is positive, go to Step 5. Otherwise, go to Step 3 to request data package.

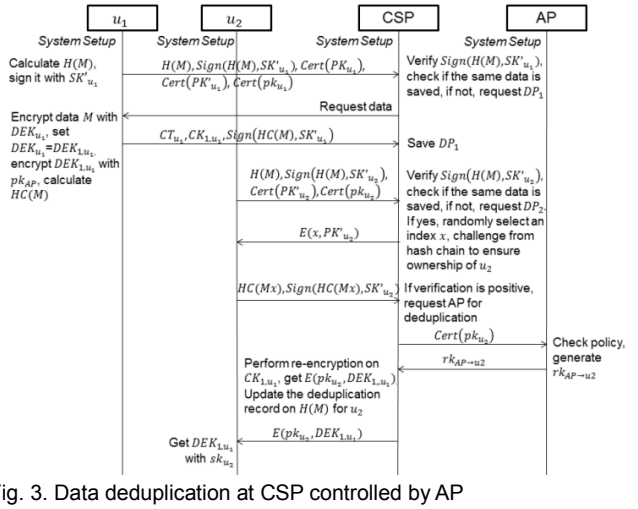


Fig. 3. Data deduplication at CSP controlled by AP

Step 3 - Data Storage: When CSP requests the data package, user u_1 encrypts M with a random symmetric key DEK_{u_1} to get $CT_{u_1} = Encrypt(DEK_{u_1}, M)$. If $DEK_{u_1} = null$, $CT_{u_1} = Encrypt(null, M) = M$. It then calls $SeparateKey(DEK_{u_1})$ to get two random parts of DEK_{u_1}: DEK_{1,u_1} and DEK_{2,u_1}. User u_1 encrypts DEK_{1,u_1} with pk_{AP} to get CK_{1,u_1} by calling $E(pk_{AP}, DEK_{1,u_1})$ and encrypts DEK_{2,u_1}

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

with pk_{ID,u_1} by calling $EncryptKey(DEK_{2,u_1}, \lambda, pk_{ID,u_1})$ to get CK_{2,u_1}, where pk_{ID,u_1} is generated according to data policy λ of u_1. In addition, it randomly selects a number of indexes: $IN = \{In_1, In_2, \dots, In_k\}$ that indicate the special parts of M (e.g., In_1 indicates first 1% of data; In_2 indicates first 3% of data), where k is the total number of indexes. Furthermore, u_1 calculates the hash codes of partial M based on the indexes as $HC(M) = \{H(M_1), H(M_2), \dots, H(M_k)\}$. Then u_1 sends the data package to CSP for storage:
$DP_1 = \{CT_{u_1}, CK_{1,u_1}, CK_{2,u_1}, IN, HC(M), Sign(HC(M), SK'_{u_1})\}$.

Step 4 - Duplicated Data Upload: Later on, user u_2 wants to store the same data M at CSP by sending CSP the data package $P_2 = \{H(M), Sign(H(M), SK'_{u_2}), Cert(PK_{u_2}), Cert(PK'_{u_2}), Cert(pk_{u_2})\}$.

Step 5 - Deduplication: CSP performs duplication check as in step 2. It further checks the correctness of $HC(M)$ by randomly selecting an index x in IN, and challenging u_2. The purpose of performing this additional check is to ensure the data ownership in case that $H(M)$ is eavesdropped or gained by some malicious party. A number of indexes in IN can be selected with regard to the hash code set challenge in order to enhance the security of the ownership check. If the verification of challenge response is positive, CSP performs data storage with deduplication.

If AP is involved into the control of deduplication, CSP contacts AP with $Cert(pk_{u_2})$ (that contains pk_{u_2}). If the verification on the data storage policy regarding u_2 is positive, AP generates $rk_{AP \to u2}$ if not performed before by calling $RG(pk_{AP}, sk_{AP}, pk_{u_2})$ and issues it to CSP to allow it to re-encrypt CK_{1,u_1} by calling $R(rk_{AP \to u2}, CK_{1,u_1})$. CSP sends $E(pk_{u_2}, DEK_{1,u_1})$ to u_2 for decryption with sk_{u_2} to get DEK_{1,u_1}.

If the control of data owner is applied to the stored data, CSP contacts u_1 by sending $H(M)$ and $Cert(PK_{u_2})$ (that contains PK_{u_2}) for deduplication. If the verification on u_2's eligibility for data storage at CSP is positive, u_1 generates SK_{ID,u_1,u_2} by calling $IssueIDSK(ID, SK_{u_1}, PK_{u_2})$, and issues it and CK_{2,u_1} to u_2. Then u_1 informs the success of data deduplication to CSP. After getting this notification, CSP updates corresponding deduplication records.

Through data deduplication, both u_1 and u_2 can access the same data M that is stored only once at CSP. User u_1 uses DEK_{u_1} directly. While u_2 gets DEK_{2,u_1} and DEK_{1,u_1} by calling $DecryptKey(CK_{2,u_1}, \lambda, SK_{u\prime}, sk_{ID,u_1,u_2})$ and $D(sk_{u_2}, CK_{1,u_1})$, respectively. It then combines DEK_{1,u_1} and DEK_{2,u_1} to get DEK_{u_1} by calling $CombineKey(DEK_{1,u_1}, DEK_{2,u_1})$.

Figure 3 describes the procedure of data deduplication at CSP with the control of AP. Figure 4 shows the procedure of data deduplication at CSP with the control of the data owner. The main differences of these two procedures from the one described in Figure 2 are:

1) The separation of DEK is different: in Figure 2, $DEK_1 || DEK_2 = DEK$, where DEK_1 and DEK_2 are not null. In Figure 3, $DEK_1 = DEK$ and DEK_2 is null. In Figure 4, $DEK_2 = DEK$ and DEK_1 is null.

2) CSP requests both the data owner and AP for deduplication in Figure 2, while CSP only requests AP for deduplication in Figure 3 and CSP only requests the data owner for deduplication in Figure 4.

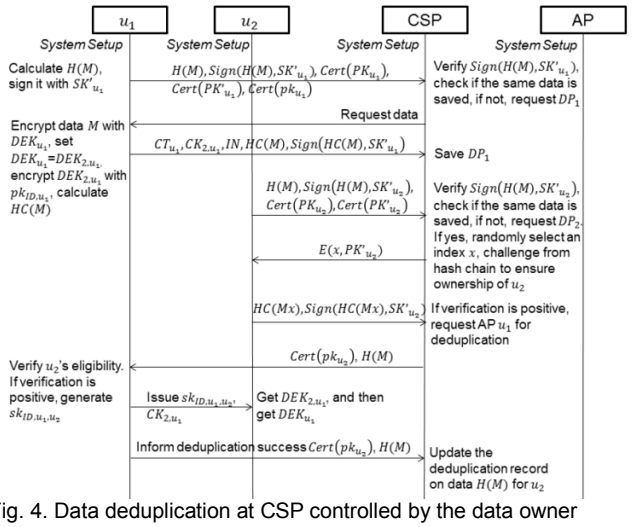


Fig. 4. Data deduplication at CSP controlled by the data owner

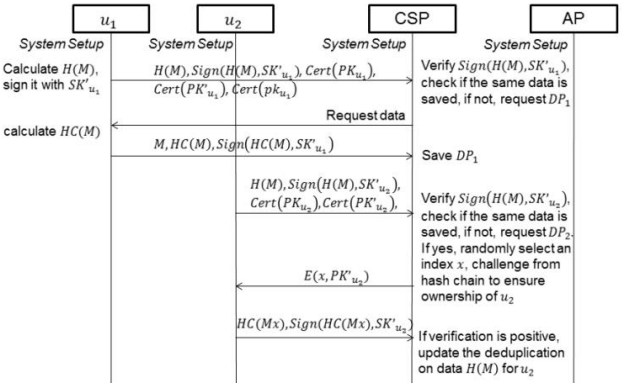


Fig. 5. Data deduplication at CSP without any control of AP or the data owner

Figure 5 shows the procedure of data deduplication at CSP without any control provided by AP and the data owner. In this case DEK is null. Plaintext is stored at CSP. If some data holder would like to store encrypted data at CSP later on, system process is similar to DEK update, which will be described below. Note that re-encryption key $rk_{AP \to u2}$ and sk_{ID,u_1,u_2} should be issued if they are not available by CSP and eligible user u_2 in the case of DEK update.

Though deduplication can help save storage cost, the data owners or holders may prefer to store replicated data in the CSP. A flag is signed by the user to indicate its preference with regard to deduplication, which is checked by CSP before performing any duplication check. On the other hand, CSP can also issue different privileges to its users. Some users could hold a specific privilege to store replicated data in the CSP, but they could be charged more than normal users by the CSP. For this type of users, the data duplication check should be waived.

2) Deduplication Across CSPs

Figure 6 shows the process of deduplication across multiple CSPs.

Step 1. The user requests its local CSP for data storage.

Step 2. The local CSP checks data duplication. If yes, the local CSP performs deduplication by contacting the data

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

owner and/or AP based on the way of data encryption for deduplication. Corresponding keys are generated by the data owner and/or AP and issued to the user if it is an eligible data holder.

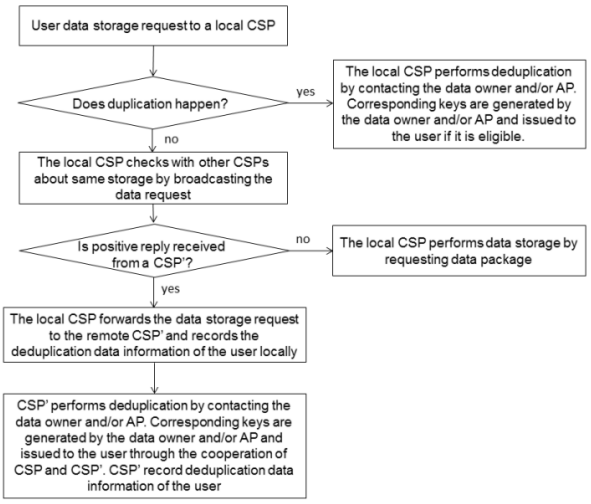


Fig. 6. Data deduplication across multiple CSPs

Step 3. If the local duplication check is negative, CSP will check with other CSPs if the same data is stored by broadcasting the data storage request of the user. If there is no any positive reply from other CSPs, the local CSP performs data storage by requesting data package from the user.

Step 4. If there is a remote CSP′ replying that the same data has been stored therein, the local CSP forwards the data storage request to CSP′ and records user data deduplication information locally. The remote CSP′ performs deduplication by contacting the data owner and/or AP. Corresponding keys are generated by the data owner and/or AP and issued to the user through the cooperation of CSP and CSP′. Meanwhile, CSP′ records the deduplication information of the user.

3) Data Deletion

Figure 7 shows the procedure of data deletion by a data holder in the context of data deduplication.

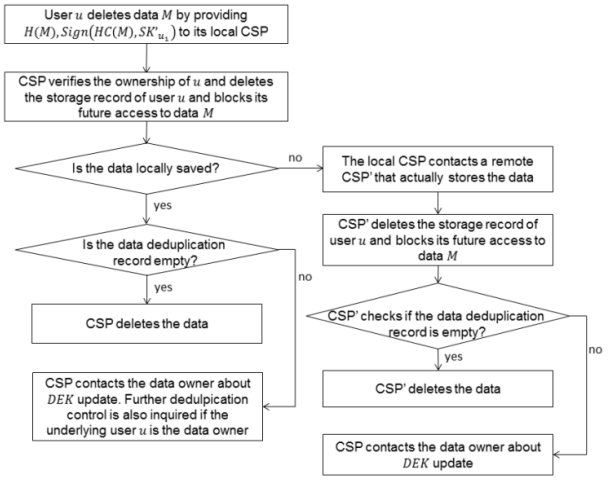


Fig. 7. A procedure of data deletion

Step 1. User u sends a request of data deletion to its local CSP by providing $H(M), Sign(H(M), SK'_u)$.

Step 2. The CSP verifies the ownership of u by randomly selecting an index x in IN and challenging expected hash code set. It deletes the storage record of u and blocks its future access to data M if the verification is positive.

Step 3. The CSP further checks if the data is locally stored. If not, go to Step 4. If yes, it will delete the data in case that the data deduplication record is empty (i.e., no user stores such data in CSP any more). It could contact the data owner about DEK update in case that the deduplication record is not empty. Further deduplication control is also required if the underlying user u is the data owner when the deduplication record is not empty.

Step 4. The local CSP contacts remote CSP′ that really stores the data. The CSP′ deletes the storage record of u and blocks its future data access. It also checks the data deduplication record. If it is empty (i.e., no user stores such data in CSP′ any more), CSP′ deletes the data. Otherwise, it contacts the data owner about DEK update (refer to *Continuous Deduplication Control* as described below).

4) Continuous Deduplication Control

Figure 8 illustrates the procedure when CSP inquires a data owner for continuous deduplication control if the data owner deletes its data at CSP, but still there are other eligible data users storing the same data at CSP.

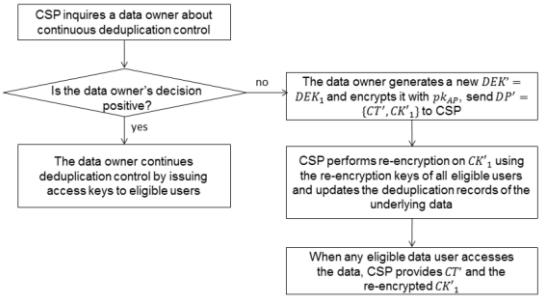


Fig. 8. A procedure of continuous deduplication control

Step 1. CSP inquires a data owner about continuous deduplication control.

Step 2. If the data owner's decision is positive, the data owner continues deduplication control by issuing access keys to eligible users. Else, go to Step 3.

Step 3. The data owner generates a new key $DEK' = DEK'_1$, encrypts it with pk_{AP}, and sends $DP' = \{CT', CK'_1\}$ to CSP. CSP performs re-encryption on CK'_1 using the re-encryption keys of all eligible users and updates the deduplication record of the underlying data. When any eligible data user accesses the data, CSP provides CT' and the re-encrypted CK'_1.

Herein, we only illustrate one solution of continuous deduplication control. Other data holders can also take over the control. In this case, CSP will request a new delegate from existing data holders and select one of them (e.g., based on the duration of data storage and user willingness). The new delegate will perform storage update by applying a newly generated key DEK'. The process is similar to DEK update as described below.

5) DEK and CT Update

Figure 9 illustrates the procedure of DEK and CT update, which is essential for enhancing system security. The data

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

owner (or an eligible data holder) u_1 generates a new key DEK'_{u_1} and encrypts data M with DEK'_{u_1}. It separates DEK'_{u_1} into DEK'_{1,u_1} and DEK'_{2,u_1}, and encrypts DEK'_{1,u_1} with pk_{AP}, and DEK'_{2,u_1} with pk_{ID,u_1}. Then data package $DP'_1 = \{CT'_{u_1}, CK'_{1,u_1}, CK'_{2,u_1}, H(M), Sign(H(M), SK'_{u_1})\}$ is sent to CSP. The CSP validates the eligibility of u_1 and stores DP'_1. CSP requests AP to get the re-encryption keys of current eligible data holders (e.g., u_2) if the re-encryption keys are not available. CSP performs re-encryption on CK'_{1,u_1}, with e.g., $rk_{AP\to u_2}$ to get $E(pk_{u_2}, DEK'_{1u_1})$.

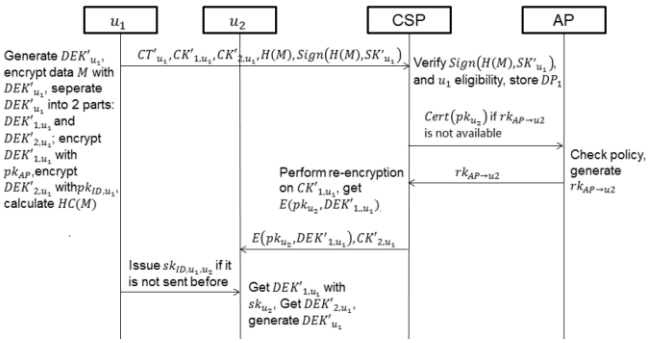


Fig. 9. A procedure of *DEK* and *CT* update

Meanwhile, u_1 also needs to issue sk_{ID,u_1,u_2} through a secure channel if it is not ever sent to eligible users. Any eligible user, e.g., u_2, can get DEK'_{1,u_1} with sk_{u_2} and gain DEK'_{2,u_1} with sk_{ID,u_1,u_2} in order to generate DEK'_{1,u_1} for accessing newly encrypted data CT'_{u_1}.

5 Performance Evaluation

5.1 Security Analysis

The security of our scheme relies on ABE theory, PRE theory, symmetric key encryption and PKC. The security of PRE and ABE was proved in our previous work [33]. Symmetric key encryption and PKC theory play as a security foundation in many security schemes. We assume that the applied key sizes of these two cryptosystems are long enough to satisfy the security requirements of our system. In what follows, we analyze the security of our scheme regarding data ownership verification and data deduplication.

Proposition 1. To pass data ownership verification, a cloud user must really hold data M.

Proof. A cloud user can generate correct $H(M)$ with real data M, thus it can pass duplication check. For ownership challenge, stolen $H(M)$ is useless since an ineligible data holder is hard to provide correct $H(M_x)$ since x is randomly selected and $H()$ is non-invertible. Eavesdropping previous transmitted $H(M_{x'})$ is useless to pass the current challenge. The user holding the real M can get M_x and generate correct $H(M_x)$, thus pass the challenge. The security level of data ownership verification links to the maximum number of k in $IN = \{In_1, In_2, \ldots, In_k\}$ and the number of indexes used to challenge the ownership. In practice, these parameters can be set according to the sensitivity of the stored data and the security requirement of the data owner.

Proposition 2. Data M can be deduplicated in a secure way and only eligible users can access it if data owner u, CSP and AP cooperate without collusion.

Proof. During data deduplication, data confidentiality is ensured by ABE, PRE and symmetric key encryption (e.g., AES). Data M could be disclosed in two ways: obtaining it from $H(M)$ and breaking $CT = Encrypt(DEK, M)$. First, the hash function is assumed hard to suffer from collision attacks. Therefore, it is impossible to obtain M through its hash code. Second, if we select a long enough key size for the symmetric key encryption, breaking CT is hard. Thus, DEK becomes the attack point of data security. In our scheme, DEK is divided into two parts: DEK_1 and DEK_2, which are encrypted with pk_{AP} through PRE and $pk_{ID,u}$ through ABE, respectively. Because AP does not collude with CSP, CSP cannot gain DEK_1 since it knows nothing about sk_{AP} although it stores CK_1. Through the re-encryption with $pk_{AP\to u'}$, CK_1 under pk_{AP} is transformed into the cipherkey under $pk_{u'}$, during which CSP cannot get to know DEK_1 since CSP knows nothing about $sk_{u'}$ and DEK_1 is always in encrypted forms. AP has no way to access M because CSP blocks its access. Even though AP obtains DEK_1 by colluding with CSP, it is still impossible for AP to get M because another part of DEK (i.e., DEK_2) is controlled by the data owner. The data owner and holders have no incentive to collude with CSP considering their personal data profits, thus they would not disclose $sk_{ID,u,u'}$ and DEK to CSP. CSP has no way to gain DEK_2 and DEK. Based on the above analysis, the CSP that stores CT cannot obtain M through DEK. The scheme can guarantee that M is securely stored at CSP during deduplication, which can be only accessed by eligible data holders.

5.2 Comparison with Existing Work

TABLE 2. COMPARISON OF COMPUTATION COMPLEXITY WITH [31][32]

Party	[this paper]	[31]	[32]
Data Owner	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$
CSP	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Data Holder	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
AP	$\mathcal{O}(n)$	$\mathcal{O}(n)$	-

n: the number of data holders.

We compare our scheme with the previous work [31][32]. One of them [31] realizes deduplication managed by AP and the other [32] manages deduplication by the online data owner. As shown in Table 2 and further tested in Section 5.4, the proposed scheme can flexibly support various scenarios with similar computation complexity to existing work [31][32]. Thereby, we compare their main properties in Table 3. We can see that the proposed scheme is a heterogeneous solution. It can realize both fine-grained and offline access control, thus it has better flexibility than previous work. In addition, the random hash code challenge is applied to verify data ownership, which can guarantee that the data holders really have the original data rather than its hash code. Though possession proof has been achieved in [31] by applying Elliptic Curve Cryptography (ECC) (with ownership verification time about 1.2 millisecond), hash code set employed in this paper is also very efficient if we make challenged part of data is small. If the challenged part of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

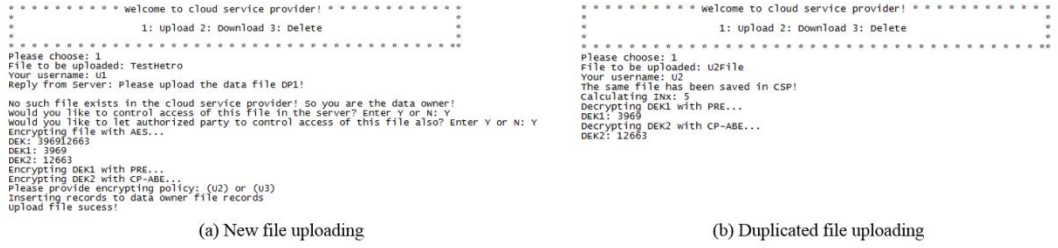
data is very small, e.g., within 1 kilobyte, we can achieve much better performance than [31] considering the fast operation time of the hash function. Moreover, our scheme can cope with the situations of deduplication across multiple CSPs, which was not considered at all in previous work. In general, our scheme has distinct advantages compared with existing work in terms of high flexibility and advanced properties.

TABLE 3. COMPARISON OF FEATURES WITH [31][32]

Properties	[this paper]	[31]	[32]
Basic Algorithm Applied	PRE, ABE	PRE, ECC	ABE
Fine-grained Access Control	√	×	√
Possession Proof	√	√	×
Offline Access Control	√	√	×
Deduplication Across CSPs	√	×	×

5.3 Scheme Implementation

We implemented the scheme based on MIRACL Crypto Library (http://info.certivox.com/docs/miracl), Pairing Based Cryptography (PBC) Library [27], OpenSSL Cryptography, JHU-MIT Proxy Re-cryptography Library (https://isi.jhu.edu/~mgreen/prl/index.html), and SSL / TLS Toolkit (https://www.openssl.org/). In our implementation, we applied AES for symmetric encryption, RSA for PKC and SHA-1 as a hash function to generate hash codes and hash sets. Our implementation was in C++ and adopted MySQL 5.5.46 to build a database. The experiments were conducted in a virtual machine running a 64-bits Ubuntu operating system on Amazon EC2 cloud service with Intel Xeon CPU E5-2670, 2.50GHz processor and 1-GB RAM. We tested the correctness of our implementation in terms of each procedure described in Section 4.3. Herein, we only take the case of data deduplication with heterogeneous control as an example to illustrate our implementation due to paper size limitation.



(a) New file uploading (b) Duplicated file uploading

Fig. 10. File uploading process with heterogeneous control

Modify	HM	Owner	SharedNums	SharedUsers	INs	CK1	CK2	HCM	CT
edit	2A8DC0991F8EA342B92FC31A7AB944516AF87883	U1	0		39, 45, 5, 85, 60, 98, 85, 89, 47, 10	281 bytes	822 bytes	200 bytes	436 bytes

(a) CSP database after a new file *TestHetro* has been uploaded by U1

Modify	HM	Owner	SharedNums	SharedUsers	INs	CK1	CK2	HCM	CT
edit	2A8DC0991F8EA342B92FC31A7AB944516AF87883	U1	1	U2	39, 45, 5, 85, 60, 98, 85, 89, 47, 10	281 bytes	822 bytes	200 bytes	436 bytes

(b) CSP database after a duplicated file *U2File* has been uploaded by U2

Fig. 11. Record of CSP database

Modify	Filename	HM	DEK	Policy	CK2
edit	TestHetro	2A8DC0991F8EA342B92FC31A7AB944516AF87883	396912663	U2 U3 1of2	822 bytes

(a) Record of U1 database after uploading *TestHetro* to CSP

Modify	Filename	HM	DEK1	DEK2
edit	U2File	2A8DC0991F8EA342B92FC31A7AB944516AF87883	3969	12663

(b) Record of U2 database after uploading *U2File* to CSP

Fig. 12. Records of U1 and U2 databases

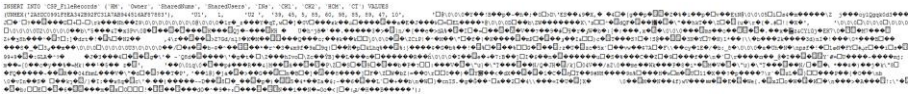


Fig. 13. The detailed data content in CSP

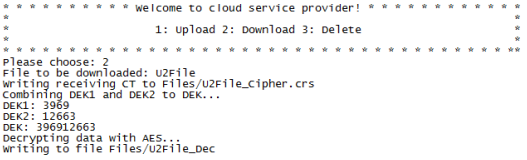


Fig. 14. Duplicated file download with heterogeneous control

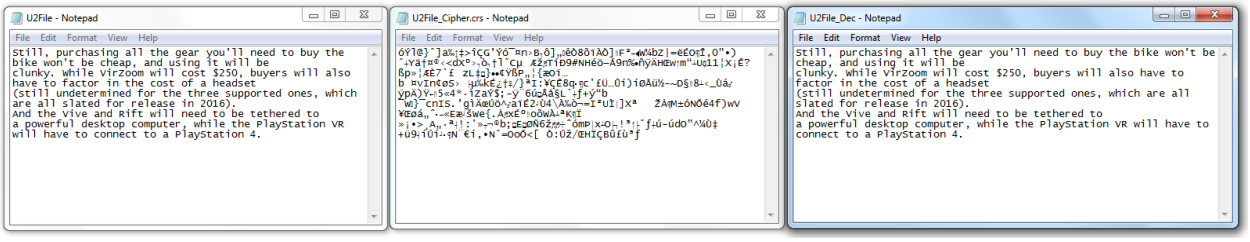


Fig. 15. The content of file *U2File*, CT and the decryption of CT

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

Use Case: Data deduplication with heterogeneous control (as illustrated in Figure 2).

Step 1: $U1$ wants to upload file *TestHetro* to CSP. CSP checks that there is no duplicated file stored, and thus requests DP_1 from $U1$. $U1$ generates DEK randomly, encrypts file *TestHetro* using AES with DEK, divides DEK into two parts, denoted as $DEK1$ and $DEK2$. $U1$ prepares DP_1 and uploads it to CSP. Upon receiving DP_1, CSP stores DP_1 in its database and $U1$ also stores the uploaded file information in its own database. Such uploading process is shown in Figure 10(a) and the records in CSP database and $U1$ database are shown in Figure 11 and Figure 12(a).

Step 2: $U2$ wants to upload file *U2File* whose content is exactly the same as that of file *TestHetro*. CSP identifies duplication and challenges the ownership of $U2$. After passing ownership challenge, $U2$ gets $CK2$, the re-encrypted $CK1$, and related attribute secrete key of $U2$. $U2$ then decrypts $CK1$ and $CK2$ to get DEK_1 and DEK_2, as shown in Figure 10(b). In addition, we can observe that $U2$ can get the correct DEK generated by $U1$. Then, $U2$ stores the received DEK_1 and DEK_2 for file *U2File*, as shown in Figure 12(b). Meanwhile, CSP updates the record of file *TestHetro* to mark $U2$ as a user that holds file *TestHetro*, as shown in Figure 11(b).

Figure 13 shows the detailed data content in CSP. We can see that the file is secure from CSP since only the ciphertexts of $DEK1$, $DEK2$ and file content are stored in CSP.

Step 3: $U2$ wants to download file *U2File*. After checking the eligibility of $U2$, CSP sends CT of file *TestHetro* to $U2$. Upon receiving CT, $U2$ decrypts it with DEK that is combined from $DEK1$ and $DEK2$, as shown in Figure 14. Figure 15 shows the content of the file *U2File* before it is uploaded, its CT and decryption of CT. We can see from Figure 15 that $U2$ can decrypt the file correctly.

5.4 Efficiency Evaluation

Based on the implementation, we performed a number of tests to evaluate the efficiency of our proposed scheme.

Test 1: Efficiency of file encryption and decryption

We tested the time spent to encrypt and decrypt a file with different sizes by applying AES with 3 different key sizes, namely 128 bits, 196 bits and 256 bits. We observe from Figure 16(a) that encrypting or decrypting a file of 500 megabytes (MB) with 256-bit AES takes about 100 seconds. It is a reasonable and practical choice to apply symmetric encryption for data protection.

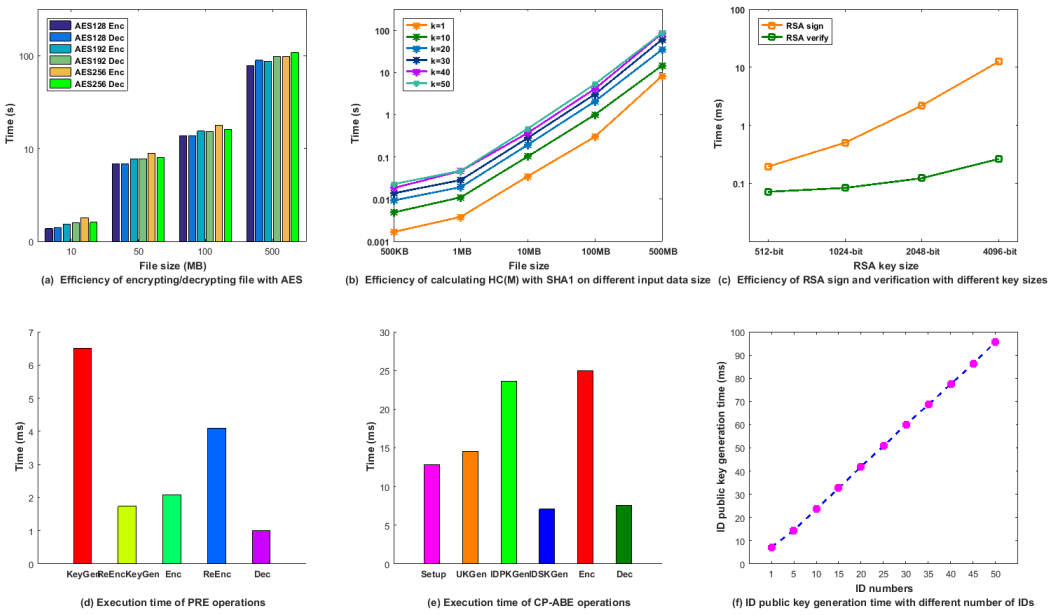


Fig. 16. Efficiency evaluation on basic algorithms

Test 2: Efficiency of calculating hash code set of a file

Figure 16(b) shows the time needed to calculate $H(M)$ $(k = 1)$ and $HC(M)$ $(k > 1)$ of files of different sizes using SHA-1. We can see from Figure 16(b) that the time increases as the file size increases and that the bigger k is, the more time it takes to calculate $HC(M)$. Calculating $H(M)$ is very efficient, which takes less than 10 seconds to calculate $H(M)$ of a file as big as 500MB. When k is small (e.g., k=50), calculating $HC(M)$ with data size 500 kilobytes (KB) is also very efficient, within 50 milliseconds.

Test 3: Efficiency of RSA sign and verification

In our proposed scheme, RSA signature is used during duplication check and performed on the hash code or the hash code set of plaintext data. Signature verification is used at CSP to ensure data ownership during duplication check. We tested the execution time needed to sign a given SHA-1 hash code and verify a given signature using RSA cryptosystem. We observed from Figure 16(c) that both RSA sign and RSA verification are very efficient. Signing with 4096-bit RSA takes only about 10 milliseconds.

Test 4: Efficiency of PRE operations

We tested the operation time of different PRE operations. PRE schemes require that all users in a PRE deployment share a common set of public parameters. These parameters should be fixed, then they need to be generated only once during system setup. We tested that generating these

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2701352, IEEE Transactions on Big Data

parameters takes about 34.79 milliseconds. Each user in a PRE deployment needs to generate a public/secrete key pair. As shown in Figure 16(d), generating a PRE key pair takes only 6.5 milliseconds. We can observe that PRE operations (including re-encryption key generation, encryption, re-encryption and decryption) are quite efficient. Thus, applying PRE to protect data encryption keys is reasonable and practical, especially when it is handled at a server with sufficient resources and processing capability.

Test 5: Efficiency of CP-ABE operations

Figure 16(e) shows the execution times of all CP-ABE operations (UKGen: User key pair generation; IDPKGen: ID public key generation (ID numbers = 10); IDSKGen: ID secret key generation; Enc: ABE encryption (ID numbers in encryption policy = 5); Dec: ABE decryption (ID numbers in encryption policy = 5)). The setup process that is needed only once generates CP-ABE global public key and secret master key, which takes about 12 milliseconds. User key pair generation takes about 14 milliseconds and is needed when a new user is registered into the system. The ID public key generation process varies with different number of IDs. Figure 16(f) shows the ID public key ($pk_{ID,u}$) generation time with different number of IDs, namely eligible users.

Figure 17(a) shows the CP-ABE encryption and decryption time. The encryption time increases with the number of IDs in encryption policy, since the encryption algorithm iterates over all IDs and constructs ciphertext for each ID. The decryption time is consistent around 7.8 milliseconds.

Following tests were carried out by applying 128-bit AES and 2048-bit RSA. The $HC(M)$ was calculated with k=10 and the number of IDs in CP-ABE encryption policy is 5.

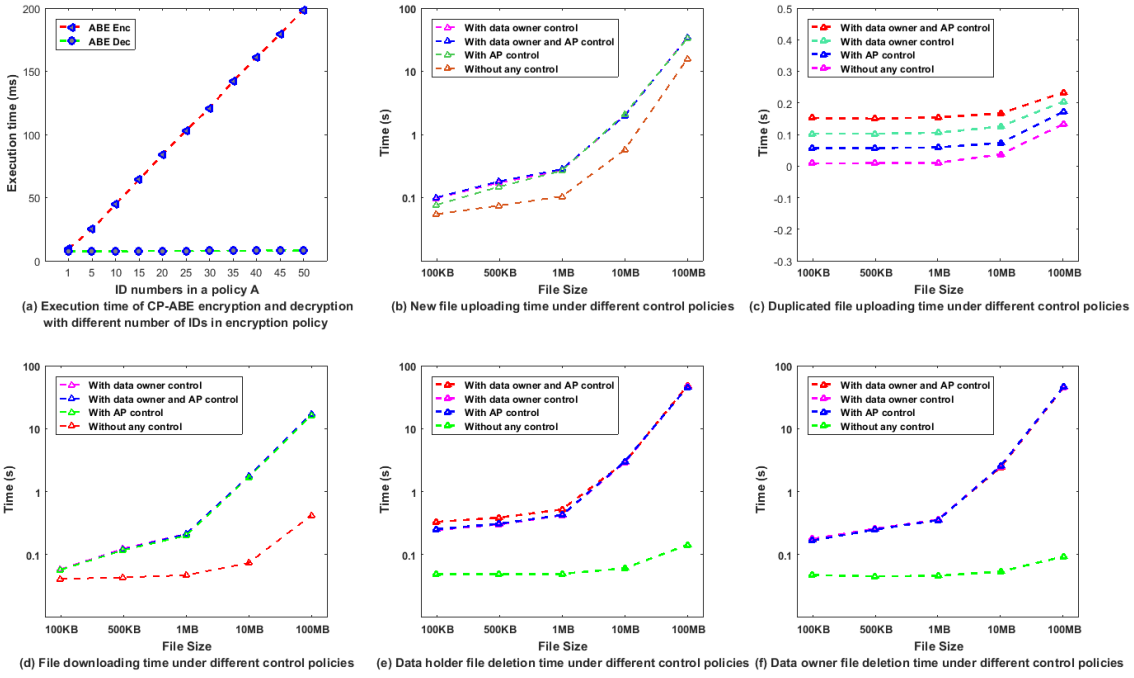


Fig. 17. Efficiency evaluation on main operations

Test 6: Efficiency of file uploading

We tested the efficiency of file uploading process under different control policies. The process includes encrypting data file with AES, calculating $H(M)$ and $HC(M)$, signing and verifying signature. The process may include encrypting $DEK_{1,u}$ with PRE and/or encrypting $DEK_{2,u}$ with ABE according to the access control policy. As shown in Figure 17(b), there is no much difference between uploading a file under three control policies, namely, data owner and AP control, data owner control, and AP control, especially for big files. Since for big files, the time is dominated mainly by AES encryption that increases with file sizes. However, CP-ABE and PRE are quite efficient (less than 1 second) and stays constant for files with different sizes, since the size of DEK stays constant for different files. We can also see from Figure 17(b) that encrypting a file does not introduce too much computation overhead. The result shown in this figure also indicates that the proposed scheme has similar performance to the existing work [31, 32] with regard to file uploading.

Figure 17(c) shows the duplicated file uploading time under different control policies. In this process, CSP will request re-encryption key from AP and use it to re-encrypt CK_1 if needed. CSP also contacts the data owner about issuing the user attribute secret key if the data owner controls data access. We observe that such a process is very efficient, taking less than 0.3 seconds if the data is less than 100MB. The operation time varies slightly with file sizes, which results from $HC(M)$ calculation and challenge. By comparing Figure 17(b) with 17(c), we can see that the proposed deduplication scheme can greatly save data uploading time for duplicated data storage at the cloud.

Test 7: Efficiency of file downloading

We also tested the efficiency of file downloading process

that combines DEK_1 and DEK_2 and decrypts downloaded CT with AES. Because the decryption of CK_1 and CK_2 is very fast (only several milliseconds), there is no much difference between the file downloading time under different control policies, as shown in Figure 17(d). But if no party controls the data access, the downloading process is much faster than that under data owner and/or AP control. Since in this case, AES decryption is not needed. This result indicates that the proposed scheme has similar performance to the existing work [31, 32] with regard to file downloading.

Test 8: Efficiency of file deletion

Figure 17(e) shows the data holder's file deletion time under different control policies. The deletion process involves DEK and CT update if the deleted file is controlled by the data owner and/or AP. The DEK and CT update process is similar to the above mentioned new file uploading process except that it does not need to calculate $HC(M)$. Thus, deleting a file under data owner and/or AP control varies slightly, especially for big files, as shown in Figure 17(e). However, deleting a file without any control by the data owner or AP only needs to update related file records in CSP, thus it is very efficient and takes less than 0.1 seconds for a file with 100 MB.

Figure 17(f) shows the data owner's file deletion time under different control policies. The data owner deletion process involves generating a new DEK and encrypting it with pk_{AP}. Therefore, there is no much difference under different access control policies. For a file without any access control, the deletion just needs to update related CSP file records and thus very efficient.

As can be seen from Figure 17, the proposed scheme achieves similar performance to the existing work [31, 32]. Considering its advanced properties as shown in Table 3 and high flexibility, we conclude that our scheme outperforms the existing work.

6 Conclusion

Data deduplication is important and significant in the practice of cloud data storage, especially for big data storage management. In this paper, we proposed a heterogeneous data storage management scheme, which offers flexible cloud data deduplication and access control. Our scheme can adapt to various application scenarios and demands and offer economic big data storage management across multiple CSPs. It can achieve data deduplication and access control with different security requirements. Security analysis, comparison with existing work and implementation based performance evaluation showed that our scheme is secure, advanced and efficient.

Our scheme supports data privacy of cloud users since the data stored at the cloud is in an encrypted form. One way to support identity privacy is to apply pseudonyms in Key Generation Center (KGC), where a real identity is linked to a pseudonym, which is verified and certified by the KGC. In our future work, we will further enhance user privacy and improve the performance of our scheme towards practical deployment. In addition, we will conduct game theoretical analysis to further prove the rationality and security of the proposed scheme.

Acknowledgment

This work is sponsored by the National Key Research and Development Program of China (grant 2016YFB0800704), the NSFC (grants 61672410 and U1536202), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2016ZDJC-06), the 111 project (grants B16037 and B08038), the PhD grant of the Ministry of Education, China (grant JY0300130104), and Aalto University.

References

[1] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," in *Proc. 2009 ACM Workshop Cloud Comput. Secur.*, pp. 85-90, 2009.

[2] S. Kamara, and K. Lauter, "Cryptographic cloud storage," *Financ. Crypto. Data Secur.*, pp. 136-149, Springer, 2010.

[3] Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Efficient information retrieval for ranked queries in cost-effective cloud environments," in *Proc. 2012 IEEE INFOCOM*, pp. 2581-2585, 2012.

[4] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: scalable secure file sharing on untrusted storage," in *Proc. USENIX Conf. File Storage Technol.*, pp. 29–42, 2003.

[5] E.-J. Goh, H. Shacham, N. Modadugu, and D. Boneh, "SiRiUS: securing remote untrusted storage," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, pp. 131-145, 2003.

[6] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. of IEEE Symp. Secur. Privacy (SP'07)*, pp. 321-334, 2007.

[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data", in *Proc. of 13th ACM Comput. Commun. Secur.*, pp. 89–98, 2006.

[8] S. Muller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," in *Proc. of 11th Annual Int. Conf. Inf. Secur. Crypto.*, pp. 20–36, 2008.

[9] A. Sahai, and B. Waters, "Fuzzy identity-based encryption," in *Proc. of 24th Int. Conf. Theory App. Cryptographic Tech.*, pp. 457–473, 2005.

[10] S. C. Yu, C. Wang, K. Ren, and W. J. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. of IEEE INFOCOM*, pp. 534–542, 2010.

[11] G. J. Wang, Q. Liu, J. Wu, and M. Y. Guo, "Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers," *Comput. Secur.*, vol. 30, no. 5, pp. 320–331, 2011.

[12] S. C. Yu, C. Wang, K. Ren, and W. J. Lou, "Attribute based data sharing with attribute revocation," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, pp. 261–270, 2010.

[13] G. J. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. of 17th ACM Comput. Commun. Secur.*, pp. 735-737, 2010.

[14] M. Zhou, Y. Mu, W. Susilo, M. H. Au, and J. Yan, "Privacy-preserved access control for cloud computing," in *Proc. of IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, pp. 83-90, 2011.

[15] Z. G. Wan, J. E. Liu, and R. H. Deng, "HASBE: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 2, pp. 743-754, 2012.

[16] Z. Yan, Trust Management in Mobile Environments – Usable and Autonomic Models, IGI Global, Hershey, Pennsylvania, 2013.

[17] Y. Tang, P. P. Lee, J. C. Lui, and R. Perlman, "Secure overlay cloud storage with access control and assured deletion," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 6, pp. 903-916, 2012.

[18] M. Bellare, S. Keelveedhi, and T. Ristenpart, "DupLESS: server aided encryption for deduplicated storage," in *Proc. of 22nd USENIX Conf. Secur.*, pp. 179-194, 2013.
[19] Dropbox, "A file-storage and sharing service," http://www.dropbox.com/.
[20] Google Drive. http://drive.google.com.
[21] Mozy, "Mozy: a file-storage and sharing Service," http://mozy.com/.
[22] J. R. Douceur, A. Adya, W. J. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *Proc. of 22nd Int. Conf. Distributed Comput. Syst.*, pp. 617-624, 2002.
[23] G. Wallace, F. Douglis, H. W. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu, "Characteristics of backup workloads in production systems," in *Proc. of USENIX Conf. File Storage Technol.*, pp. 500, 2012.
[24] Z. O. Wilcox, "Convergent encryption reconsidered," 2011. http://www.mail-archive.com/cryptography@metzdowd.com/msg08949.html.
[25] C. Yang, J. Ren, and J. F. Ma, "Provable ownership of file in de-duplication cloud storage," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, pp. 695-700, 2013.
[26] T.-Y. Wu, J.-S. Pan, and C.-F. Lin, "Improving accessing efficiency of cloud storage using de-duplication and feedback schemes," *IEEE Systems J.*, vol. 8, no. 1, pp. 208-218, 2014.
[27] C.-I. Fan, S.-Y. Huang, and W.-C. Hsu, "Hybrid data deduplication in cloud environment," in *2012 Int. Conf. Inf. Secur. Intell. Control (ISIC)*, pp. 174-177, 2012.
[28] J. W. Yuan, and S. C. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in *IEEE 2013 Conf. Commun. Netw. Secur. (CNS)*, pp. 145-153, 2013.
[29] N. Kaaniche, and M. Laurent, "A secure client side deduplication scheme in cloud storage environments," in *2014 6th Int. Conf. New Technol., Mobility Secur. (NTMS)*, pp. 1-7, 2014.
[30] Z. Yan, W. X. Ding, and H. Q. Zhu, "A scheme to manage encrypted data storage with deduplication in cloud," in *Proc. of ICA3PP2015*, pp. 547-561: Springer, 2015.
[31] Z. Yan, W. X. Ding, X. X. Yu, H. Q. Zhu, and R. H. Deng, "Deduplication on encrypted big data in cloud," *IEEE Trans. on Big Data*, vol. 2, no. 2, pp. 138-150, April-June 2016.
[32] Z. Yan, M. J. Wang, Y. X. Li, and A. V. Vasilakos, "Encrypted data management with deduplication in cloud computing," *IEEE Cloud Comput. Mag.*, vol. 3, no. 2, pp. 28-35, 2016.
[33] Z. Yan, X. Y. Li, M. J. Wang, A.V. Vasilakos, "Flexible data access control based on trust and reputation in cloud computing," *IEEE Trans. Cloud Comput.*, 2015. Doi: 10.1109/TCC.2015.2469662.
[34] J. Hur; D. Koo; Y. Shin; and K. Kang, "Secure Data Deduplication with Dynamic Ownership Management in Cloud Storage," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 3113-3125, 2016.
[35] J. Li, X. F. Chen, M. Q. Li, J. W. Li, P. P. C. Lee; and W. J. Lou, "Secure Deduplication with Efficient and Reliable Convergent Key Management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1615-1625, 2014.
[36] M. Wen, K. Ota, H. Li, J. S. Lei, C. H. Gu; and Z. Su, "Secure Data Deduplication With Reliable Key Management for Dynamic Updates in CPSS," *IEEE Trans. Comput. Social Syst.*, vol. 2, no. 4, pp.137-147, 2015.
[37] J. Li, Y. K. Li, X. F. Chen, P. P. C. Lee, and W. J. Lou. "A hybrid cloud approach for secure authorized deduplication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1206-1216, 2015.
[38] J. Liu, N. Asokan, and B. Pinkas. "Secure deduplication of encrypted data without additional independent servers," in *Proc. of 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, pp. 874-885. ACM, 2015.
[39] Q. Liu, G. J. Wang, and J. Wu, "Consistency as a service: Auditing cloud consistency" *IEEE Trans. Netw. Serv. Manage.*, vol.11, no.1, pp. 25-35, 2014.
[40] Q. Duan, "Cloud service performance evaluation: Status, challenges, and opportunities – A survey from the system modeling perspective", *Digital Commun. Netw.*, Available online 23 December 2016, ISSN 2352-8648, http://dx.doi.org/10.1016/j.dcan.2016.12.002.
[41] Q. Liu, C. C. Tan, J. Wu, and G. J. Wang, "Towards differential query services in cost-efficient clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1648-1658, 2014.

Zheng Yan (M'06, SM'14) received the BEng degree in electrical engineering and the MEng degree in computer science and engineering from the Xi'an Jiaotong University, Xi'an, China in 1994 and 1997, respectively, the second MEng degree in information security from the National University of Singapore, Singapore in 2000, and the Licentiate of Science and the Doctor of Science in Technology in electrical engineering from Helsinki University of Technology, Helsinki, Finland in 2005 and 2007. She is currently a professor at the Xidian University, Xi'an, China and a visiting professor at the Aalto University, Espoo, Finland. She authored more than 150 peer-reviewed publications and solely authored two books. She is the inventor and co-inventor of over 50 patents and PCT patent applications. Her research interests are in trust, security and privacy, social networking, cloud computing, networking systems, and data mining. Prof. Yan serves as an associate editor of Information Sciences, Information Fusion, IEEE Internet of Things Journal, IEEE Access Journal, JNCA, Security and Communication Networks, etc. She is a leading guest editor of many reputable journals including ACM TOMM, FGCS, IEEE Systems Journal, MONET, etc. She served as a steering, organization and program committee member for over 70 international conferences. She is a senior member of the IEEE.

Lifang Zhang received the BSc degree in electrical engineering from Beijing Forestry University, Beijing, China. She achieved MSc in the Department of Communications and Networking, Aalto University, Espoo, Finland. Her research interests are in network security and data privacy.

Wenxiu Ding received her BEng degree in information security from the Xidian University, Xi'an, China in 2012. She is currently pursuing her Ph.D. degree in information security, the School of Cyber Engineering in the Xidian University, and also a visiting research student at the School of Information Systems, Singapore Management University. Her research interests are in RFID authentication, privacy preservation, cloud security, data mining and trust management.

Qinghua Zheng received the BSc degree in computer software in 1990, the MSc degree in computer organization and architecture in 1993, and the PhD degree in system engineering in 1997 from Xian Jiaotong University, China. He did postdoctoral research at Harvard University from February 2002 to October 2002 and visiting professor research at HongKong University from November 2004 to January 2005. Since 1995, he has been with the Department of Computer Science and Technology at Xi'an Jiaotong University. He is currently a professor and serves as the vice president of Xi'an Jiaotong University. His research interests include intelligent e-learning, network security, and trusted software. He is a member of the IEEE.