6-2016

Deduplication on encrypted big data in cloud

Zheng YAN

Wenxiu DING

Xixun YU

Haiqi ZHU

DENG, Robert H.
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1109/TBDATA.2016.2587659

Citation

YAN, Zheng; DING, Wenxiu; YU, Xixun; ZHU, Haiqi; and DENG, Robert H.. Deduplication on encrypted big data in cloud. (2016). *IEEE Transactions on Big Data*. 2, (2), 138-150. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/3345

Published in IEEE Transactions on Big Data, 2016 June, Volume 2, Issue 2, Pages 138-150
https://doi.org/10.1109/TBDATA.2016.2587659

Deduplication on Encrypted Big Data in Cloud

Zheng Yan, *Senior Member, IEEE*, Wenxiu Ding, Xixun Yu, Haiqi Zhu, and Robert H. Deng, *Fellow, IEEE*

Abstract—Cloud computing offers a new way of service provision by re-arranging various resources over the Internet. The most important and popular cloud service is data storage. In order to preserve the privacy of data holders, data are often stored in cloud in an encrypted form. However, encrypted data introduce new challenges for cloud data deduplication, which becomes crucial for big data storage and processing in cloud. Traditional deduplication schemes cannot work on encrypted data. Existing solutions of encrypted data deduplication suffer from security weakness. They cannot flexibly support data access control and revocation. Therefore, few of them can be readily deployed in practice. In this paper, we propose a scheme to deduplicate encrypted data stored in cloud based on ownership challenge and proxy re-encryption. It integrates cloud data deduplication with access control. We evaluate its performance based on extensive analysis and computer simulations. The results show the superior efficiency and effectiveness of the scheme for potential practical deployment, especially for big data deduplication in cloud storage.

Index Terms—Access control, big data, cloud computing, data deduplication, proxy re-encryption

1 Introduction

Cloud computing offers a new way of Information Technology services by rearranging various resources (e.g., storage, computing) and providing them to users based on their demands. Cloud computing provides a big resource pool by linking network resources together. It has desirable properties, such as scalability, elasticity, fault-tolerance, and pay-per-use. Thus, it has become a promising service platform.

The most important and popular cloud service is data storage service. Cloud users upload personal or confidential data to the data center of a Cloud Service Provider (CSP) and allow it to maintain these data. Since intrusions and attacks towards sensitive data at CSP are not avoidable [35], it is prudent to assume that CSP cannot be fully trusted by cloud users. Moreover, the loss of control over their own personal data [44], [45] leads to high data security risks, especially data privacy leakages [43]. Due to the rapid development of data mining and other analysis technologies, the privacy issue becomes serious [42]. Hence, a good practice is to only outsource encrypted data to the cloud in order to ensure data security and user privacy [36]. But the same or different users may upload duplicated data in encrypted form to CSP, especially for scenarios where data are shared among many users. Although cloud storage space is huge, data duplication greatly wastes network resources, consumes a lot of energy, and complicates data management. The development of numerous services further makes it urgent to deploy efficient resource management mechanisms [37], [38]. Consequently, deduplication becomes critical for big data storage and processing in the cloud.

Deduplication has proved to achieve highcost savings, e.g., reducing up to 90-95 percent storage needs for backup applications [9] and up to 68 percent in standard file systems [10]. Obviously, the savings, which can be passed back directly or indirectly to cloud users, are significant to the economics of cloud business. How to manage encrypted data storage with deduplication in an efficient way is a practical issue. However, current industrial deduplication solutions cannot handle encrypted data. Existing solutions for deduplication suffer from brute-force attacks [7], [11], [12], [13], [14]. They cannot flexibly support data access control and revocation at the same time [16], [18], [19], [20]. Most existing solutions cannot ensure reliability, security and privacy with sound performance [23], [24], [25].

In practice, it is hard to allow data holders to manage deduplication due to a number of reasons. First, data holders may not be always online or available for such a management, which could cause storage delay. Second, deduplication could become too complicated in terms of communications and computations to involve data holders into deduplication process. Third, it may intrude the privacy of data holders in the process of discovering duplicated data. Forth, a data holder may have no idea how to issue data access rights or deduplication keys to a user in some situations when it does not know other data holders due to data super-distribution. Therefore, CSP cannot cooperate with data holders on data storage deduplication in many situations.

In this paper, we propose a scheme based on data ownership challenge and Proxy Re-Encryption (PRE) to manage encrypted data storage with deduplication. We aim to solve the issue of deduplication in the situation where the data holder is not available or difficult to get involved. Meanwhile, the performance of data deduplication in our scheme

- *Z. Yan is with the State Key Laboratory on Integrated Services Networks, Xidian University, Xi'an 710071, China and with the Department of Communications and Networking, Aalto University, Espoo 02150, Finland. E-mail: zyan@xidian.edu.cn.*
- *W. Ding and X. Yu are with the State Key Laboratory on Integrated Services Networks, Xidian University, Xi'an 710071, China. E-mail: {wenxiuding_1989, Gabrielyu}@126.com.*
- *H. Zhu is with the Baidu Online Network Technology, Beijing 100085, China. E-mail: zhuhaiqi_xdu@163.com.*
- *R.H. Deng is with the School of Information Systems, Singapore Management University, Singapore. E-mail: robertdeng@smu.edu.sg.*

Manuscript received 15 Feb. 2016; revised 15 May 2016; accepted 25 June 2016. Date of publication 12 July 2016; date of current version 29 July 2016. Recommended for acceptance by H. Wang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TBDATA.2016.2587659

is not influenced by the size of data, thus applicable for big data. Specifically, the contributions of this paper can be summarized as below:

- We motivate to save cloud storage and preserve the privacy of data holders by proposing a scheme to manage encrypted data storage with deduplication. Our scheme can flexibly support data sharing with deduplication even when the data holder is offline, and it does not intrude the privacy of data holders.
- We propose an effective approach to verify data ownership and check duplicate storage with secure challenge and big data support.
- We integrate cloud data deduplication with data access control in a simple way, thus reconciling data deduplication and encryption.
- We prove the security and assess the performance of the proposed scheme through analysis and simulation. The results show its efficiency, effectiveness and applicability.

The rest of the paper is organized as follows. Section 2 gives a brief overview of related work. Section 3 introduces system and security models, preliminaries and notation. Section 4 gives the detailed description of our scheme, followed by security analysis and performance evaluation in Section 5. Finally, a conclusion is presented in the last section.

2 Related Work

2.1 Encrypted Data Deduplication

Cloud storage service providers such as Dropbox [2], Google Drive [3], Mozy [4], and others perform deduplication to save space by only storing one copy of each file uploaded. However, if clients conventionally encrypt their data, storage savings by deduplication are totally lost. This is because the encrypted data are saved as different contents by applying different encryption keys. Existing industrial solutions fail in encrypted data deduplication. For example, DeDu [17] is an efficient deduplication system, but it cannot handle encrypted data.

Reconciling deduplication and client-side encryption is an active research topic [1]. Message-Locked Encryption (MLE) intends to solve this problem [5]. The most prominent manifestation of MLE is Convergent Encryption (CE), introduced by Douceur et al. [6] and others [7], [11], [12]. CE was used within a wide variety of commercial and research storage service systems. Letting M be a file's data, a client first computes a key $K \leftarrow H(M)$ by applying a cryptographic hash function H to M, and then computes ciphertext $C \leftarrow E(K, M)$ via a deterministic symmetric encryption scheme. A second client B encrypting the same file M will produce the same C, enabling deduplication. However, CE is subject to an inherent security limitation, namely, susceptibility to offline brute-force dictionary attacks [13], [14]. Knowing that the target data M underlying the target ciphertext C is drawn from a dictionary $S = \{M_1, \ldots, M_n\}$ of size n, an attacker can recover M in the time for $n = |S|$ off-line encryptions: for each $i = 1, \ldots, n$, it simply CE-encrypts M_i to get a ciphertext denoted as C_i and returns M_i such that $C = C_i$. This works because CE is deterministic and keyless. The security of CE is only possible when the target data is drawn from a space too large to exhaust.

Another problem of CE is that it is not flexible to support data access control by data holders, especially for data revocation process, since it is impossible for data holders to generate the same new key for data re-encryption [18], [19]. An image deduplication scheme adopts two servers to achieve verifiability of deduplication [18]. The CE-based scheme described in [19] combines file content and user privilege to obtain a file token with token unforgeability. However, both schemes directly encrypt data with a CE key, thus suffer from the problem as described above. To resist the attack of manipulation of data identifier, Meye et al. proposed to adopt two servers for intra-user deduplication and inter-deduplication [20]. The ciphertext C of CE is further encrypted with a user key and transferred to the servers. However, it does not deal with data sharing after deduplication among different users. ClouDedup [16] also aims to cope with the inherent security exposures of CE, but it cannot solve the issue caused by data deletion. A data holder that removes the data from the cloud can still access the same data since it still knows the data encryption key if the data is not completely removed from the cloud.

Bellare et al. [1] proposed DupLESS that provides secure deduplicated storage to resist brute-force attacks. In DupLESS, a group of affiliated clients (e.g., company employees) encrypt their data with the aid of a Key Server (KS) that is separate from a Storage Service (SS). Clients authenticate themselves to the KS, but do not leak any information about their data to it. As long as the KS remains inaccessible to attackers, high security can be ensured. Obviously, DupLESS cannot control data access of other data users in a flexible way. Alternatively, a policy-based deduplication proxy scheme [15] was proposed but it did not consider duplicated data management (e.g., deletion and owner management) and did not evaluate scheme performance.

As stated in [21], reliability, security and privacy should be taken into considerations when designing a deduplication system. The strict latency requirements of primary storage lead to the focus on offline deduplication systems [22]. Recent studies proposed techniques to improve restore performance [23], [24], [25]. Fu et al. [23] proposed History-Aware Rewriting (HAR) algorithm to accurately identify and rewrite fragmented chunks, which improved the restore performance. Kaczmarczyk et al. [24] focused on inter-version duplication and proposed Context-Based Rewriting (CBR) to improve the restore performance for latest backups by shifting fragmentation to older backups. Another work [25] even proposed to forfeit deduplication to reduce the chunk fragmentation by container capping. In our previous work [33], we proposed using PRE for cloud data deduplication. This scheme applies the hash code of data to check ownership with signature verification, which is unfortunately insecure if $H(M)$ is disclosed to a malicious user. In this paper, we propose a new ownership verification approach to improve our previous work and aim to support big data deduplication in an efficient way.

2.2 Data Ownership Verification and Others

Halevi et al. [39] first introduced the practical implementation of Proofs of Ownership (PoW) based on Merkle tree for deduplication, which realized client-side deduplication. They proposed to apply an erasure coding or hash function

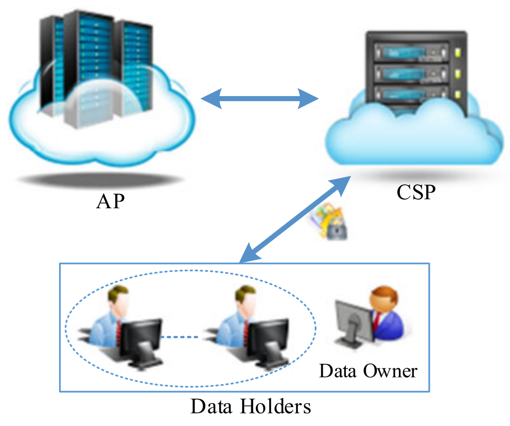


Fig. 1. System model.

over the original file first and then use Merkle tree on the pre-processed data to generate the verification information. When challenging a prover, a verifier randomly chooses several leaves of the tree and obtains the corresponding sibling-paths of all these leaves. Only when all paths are valid, will the verifier accept the proof. This construction can identify deduplication at a client to save network bandwidth and guarantee that the client holds a file rather than some part. Pietro et al. [40] chose the projection of a file onto some randomly selected bit-positions as proof to realize the PoW. But both schemes above do not pay attention to data privacy and the server for data storage could be aware of the file content. Ng et al. [41] adapted the PoW to manage the deduplication of encrypted data. This scheme also generates verification information for deduplication check based on Merkle trees. Each leaf value is generated based on several data blocks, while each interactive proof protocol can only challenge one leaf of the Merkle tree. In order to achieve higher security by checking more data, the protocol should be executed multiple times which introduces much overhead.

Yang et al. also proposed a cryptographically secure and efficient scheme to check the ownership of a file, in which a client proves to the server that it indeed possesses the entire file without uploading the file [28]. By relying on dynamic spot checking, a data holder only needs to access small but dynamic portions of the original file to generate the proof of possession of the original file, thus greatly reducing the burden of computation on the data holder and minimizing the communication cost between the data holder and CSP. At the same time, by utilizing dynamic coefficients and randomly chosen indices of the original files, the scheme mixes the randomly sampled portions of the original file with the dynamic coefficients to generate the unique proof in every challenge. The work focuses on ownership proof of the uploaded data during data deduplication. In this paper, we use Eillptic Curve Cryptography (ECC) to verify data ownership with the support of an authorized party.

Yuan and Yu attempted to solve the issue of supporting efficient and secure data integrity auditing with storage deduplication for cloud storage [31]. They proposed a novel scheme based on techniques including polynomial-based authentication tags and homomorphic linear authenticators. Their design allows deduplication of both files and their corresponding authentication tags. Data integrity auditing and storage deduplication are achieved simultaneously. Public auditing and batch auditing are both supported. But feasibility of supporting deduplication big data was not discussed in this work.

In order to reduce workloads due to duplicate files, Wu et al. proposed Index Name Servers (INS) to manage not only file storage, data deduplication, optimized node selection, and server load balancing, but also file compression, chunk matching, real-time feedback control, IP information, and busy level index monitoring [29]. To manage and optimize storage nodes based on a client-side transmission status by the proposed INS, all nodes must elicit optimal performance and offer suitable resources to clients. In this way, not only can the performance of a storage system be improved, but the files can also be reasonably distributed, decreasing the workload of the storage nodes. However, this work cannot deduplicate encrypted data.

Fan et al. proposed a hybrid data deduplication mechanism that provides a practical solution with partial semantic security [30]. This solution supports deduplication on plaintext and ciphertext. But this mechanism cannot support encrypted data deduplication very well. It works based on the assumption that CSP knows the encryption key of data. Thus it cannot be used in the situation that the CSP cannot be fully trusted by the data holders or owners.

In this paper, we apply ECC, PRE and symmetric encryption to deduplicate encrypted data. Our scheme can resist the attacks mentioned above in CE and achieve good performance without keeping data holders online all the time. Meanwhile, it also ensures the confidentiality of stored data and supports digital rights management. We aim to achieve deduplication on encrypted big data in cloud.

3 Problem Statements

3.1 System and Security Model

We propose a scheme to deduplicate encrypted data at CSP by applying PRE to issue keys to different authorized data holders based on data ownership challenge. It is applicable in scenarios where data holders are not available for deduplication control.

As shown in Fig. 1, the system contains three types of entities: 1) CSP that offers storage services and cannot be fully trusted since it is curious about the contents of stored data, but should perform honestly on data storage in order to gain commercial profits; 2) data holder (u_i) that uploads and saves its data at CSP. In the system, it is possible to have a number of eligible data holders (u_i, $i = 1, \ldots, n$) that could save the same encrypted raw data in CSP. The data holder that produces or creates the file is regarded as data owner. It has higher priority than other normal data holders, which will be presented in Section 4; 3) an authorized party (AP) that does not collude with CSP and is fully trusted by the data holders to verify data ownership and handle data deduplication. In this case, AP cannot know the data stored in CSP and CSP should not know the plain user data in its storage.

In theory it is possible that CPS and its users (e.g., data holders) can collude. In practice, however, such collusion could make the CSP lose reputation due to data leakage. A negative impact of bad reputation is the CSP will lose its

TABLE 1
System Notation

Key	Description
$(pk_i,\ sk_i)$	The public key and secret key of user u_i for PRE
DEK_i	The symmetric key of u_i
$H()$	The hash function
CT	The ciphertext
CK	The cipherkey
M	The user data
KG	The key generation algorithm of PRE
RG	The re-encryption key generation algorithm of PRE
E	The encryption algorithm of PRE
R	The re-encryption algorithm of PRE
D	The decryption algorithm of PRE
$Encrypt(DEK, M)$	The symmetric encryption function on M with DEK
$Decrypt(DEK, CT)$	The symmetric decryption function on CT with DEK
$(V_i,\ s_i)$	The key pair of user u_i in Eillptic Curve Cryptography (ECC) for duplication check
P	The base point in ECC
$x = H(H(M) * P)$	The token used for data duplication check

users and finally make it lose profits. On the other hand, the CSP users (e.g., data holders) could lose their convenience and benefits of storing data in CSP due to bad reputation of cloud storage services. Thus, the collusion between CSP and its users is not profitable for both of them. Concrete analysis based on Game Theory is provided in [26]. Therefore, we hold such an assumption as: CSP does not collude with its users, e.g., performing re-encryption for unauthorized users to allow them to access data.

Additional assumptions include: data holders honestly provide the encrypted hash codes of data for ownership verification. The data owner has the highest priority. A data holder should provide a valid certificate in order to request a special treatment. Users, CSP and AP communicate through a secure channel (e.g., SSL) with each other. CSP can authenticate its users in the process of cloud data storage. We further assume that the user policy $Policy(u)$ for data storage, sharing and deduplication is provided to CSP during user registration.

3.2 Preliminary and Notation

3.2.1 Preliminaries

1) Proxy Re-Encryption

A PRE scheme is represented as a tuple of (possibly probabilistic) polynomial time algorithms $(KG; RG; E; R; D)$:

$(KG; E; D)$ are the standard key generation, encryption, and decryption algorithms. On input the security parameter 1^k, KG outputs a public and private key pair $(pk_A; sk_A)$ for entity A. On input pk_A and data M, E outputs a ciphertext $C_A = E(pk_A; M)$. On input sk_A and ciphertext C_A, D outputs the plain data $M = D(sk_A; C_A)$.

On input $(pk_A; sk_A; pk_B)$, the re-encryption key generation algorithm RG, outputs re-encryption key $rk_{A\to B}$ for a proxy.

On input $rk_{A\to B}$ and ciphertext C_A, the re-encryption function R, outputs $R(rk_{A\to B}; C_A) = E(pk_B; m) = C_B$ which can be decrypted with private key sk_B.

2) Symmetric Encryption

Encrypt(DEK, M). The Encrypt algorithm takes as input data M, the symmetric key DEK. It encrypts M with DEK and outputs the ciphertext CT. This process is conducted at user u to protect its data stored at CSP with DEK.

Decrypt(DEK, CT). The Decrypt algorithm takes as input the encrypted data CT, the symmetric key DEK. The algorithm decrypts CT with DEK and outputs the plain data M. A user (data holder) conducts this process to gain the plaintext of stored data at CSP.

3.2.2 Notation

Table 1 summarizes the notation used in the proposed scheme.

3.2.3 System Setup

There are two groups G_1, G_T of prime order q with a bilinear map $e: G_1 \times G_1 \to G_T$. The system parameters are random generators $g \in G_1$ and $Z = e(g, g) \in G_T$.

During system setup, every data holder u_i generates sk_i and pk_i for PRE: $sk_i = a_i,\ pk_i = g^{a_i}$ where $a_i \in \mathbb{Z}_p$. The public key pk_i is used for generating the re-encryption key at AP for u_i. Assuming that $Eq(a, b)$ is an elliptic curve over $GF(q)$, P is a base point that is shared among system entities, $s_i \in_R \{0, \ldots, 2^\sigma - 1\}$ is the ECC secret key of user u_i and $V_i = -s_i P$ is the corresponding public key and σ is a security parameter. The keys $(pk_i;\ sk_i)$ and $(V_i;\ s_i)$ of u_i are bound to a unique identifier of the user, which can be a pseudonym. This binding is crucial for the verification of the user identity. AP independently generates pk_{AP} and sk_{AP} for PRE and broadcast pk_{AP} to CSP users.

4 Scheme

Our scheme contains the following main aspects:

Encrypted Data Upload. If data duplication check is negative, the data holder encrypts its data using a randomly selected symmetric key DEK in order to ensure the security and privacy of data, and stores the encrypted data at CSP together with the token used for data duplication check. The data holder encrypts DEK with pk_{AP} and passes the encrypted key to CSP.

Data Deduplication. Data duplication occurs at the time when data holder u tries to store the same data that has been stored already at CSP. This is checked by CSP through token comparison. If the comparison is positive, CSP contacts AP for deduplication by providing the token and the data holder's PRE public key. The AP challenges data ownership, checks the eligibility of the data holder, and then issues a re-encryption key that can convert the encrypted DEK to a form that can only be decrypted by the eligible data holder.

Data Deletion. When the data holder deletes data from CSP, CSP firstly manages the records of duplicated data

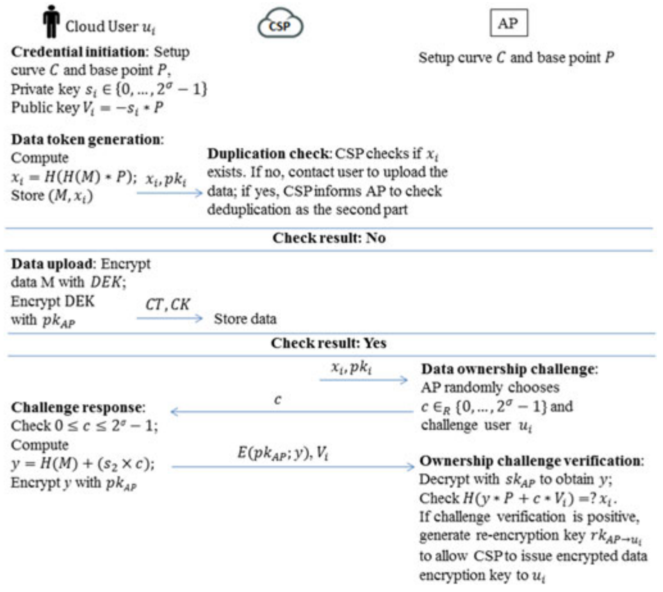

Cloud User u_i
CSP
AP
Credential initiation: Setup curve C and base point P, Private key $s_i \in \{0,\ldots,2^\sigma-1\}$ Public key $V_i = -s_i * P$
Setup curve C and base point P
Data token generation: Compute $x_i = H(H(M) * P)$; Store (M, x_i)
x_i, pk_i
Duplication check: CSP checks if x_i exists. If no, contact user to upload the data; if yes, CSP informs AP to check deduplication as the second part
Check result: No
Data upload: Encrypt data M with DEK; Encrypt DEK with pk_{AP}
CT, CK
Store data
Check result: Yes
x_i, pk_i
Data ownership challenge: AP randomly chooses $c \in_R \{0,\ldots,2^\sigma-1\}$ and challenge user u_i
c
Challenge response: Check $0 \le c \le 2^\sigma - 1$; Compute $y = H(M) + (s_2 \times c)$; Encrypt y with pk_{AP}
$E(pk_{AP}; y), V_i$
Ownership challenge verification: Decrypt with sk_{AP} to obtain y; Check $H(y * P + c * V_i) =? x_i$. If challenge verification is positive, generate re-encryption key $rk_{AP\to u_i}$ to allow CSP to issue encrypted data encryption key to u_i

Fig. 2. Data ownership verification.

holders by removing the duplication record of this user. If the rest records are not empty, the CSP will not delete the stored encrypted data, but block data access from the holder that requests data deletion. If the rest records are empty, the encrypted data should be removed at CSP.

Data Owner Management. In case that a real data owner uploads the data later than the data holder, the CSP can manage to save the data encrypted by the real data owner at the cloud with the owner generated DEK and later on, AP supports re-encryption of DEK at CSP for eligible data holders.

Encrypted Data Update. In case that DEK is updated by a data owner with DEK' and the new encrypted raw data is provided to CSP to replace old storage for the reason of achieving better security, CSP issues the new re-encrypted DEK' to all data holders with the support of AP.

4.1 Scheme to Verify Data Ownership

In order to check duplication, we first propose an ownership verification protocol based on a cryptoGPS identification scheme [47]. As shown in Fig. 2, we let AP to challenge data holder u_i to ensure that it is the real party that possesses data M. CSP just checks if duplication happens by verifying if the token $x_i = H(H(M) \times P)$ of data M has existed already. This design ensures that CSP cannot gain s_i and disclose $H(M)$ to a malicious party. In case that the same token exists, CSP passes the further verification to AP that randomly chooses $c \in_R \{0, \ldots, 2^\sigma - 1\}$ and challenges u_i by c. u_i checks c to make sure that $0 \le c \le 2^\sigma - 1$, computes $y = H(M) + (s_i \times c)$ and sends y and V_i to AP. AP computes $H(yP + cV_i)$ and compares it with x_i. If challenge verification is positive, i.e., $H(yP + cV_i) = x_i$, AP generates reencryption key $rk_{AP\to u_i}$ and issues it to CSP. CSP re-encrypts $E(pk_{AP}, DEK)$ and provides the re-encrypted key to u_i, refer to Section 4.2.

4.2 Scheme to Grant Access to Duplicated Data

Data holder u_i encrypts its secret key DEK_i with $E(pk_{AP}, DEK_i)$ and publishes it along with encrypted data $Encrypt(DEK_i, M)$ to CSP. If another data holder u_j uploads the same raw data encrypted using a different key, CSP contacts AP to challenge ownership and grant DEK_i to the eligible duplicated data holder u_j as described in Algorithm 1.

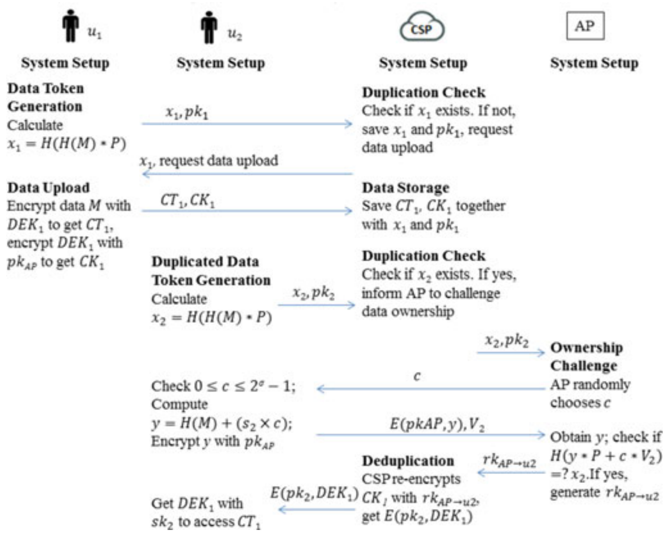

u_1
u_2
CSP
AP
System Setup
System Setup
System Setup
System Setup
Data Token Generation Calculate $x_1 = H(H(M) * P)$
x_1, pk_1
Duplication Check Check if x_1 exists. If not, save x_1 and pk_1, request data upload
x_1, request data upload
Data Upload Encrypt data M with DEK_1 to get CT_1, encrypt DEK_1 with pk_{AP} to get CK_1
CT_1, CK_1
Data Storage Save CT_1, CK_1 together with x_1 and pk_1
Duplicated Data Token Generation Calculate $x_2 = H(H(M) * P)$
x_2, pk_2
Duplication Check Check if x_2 exists. If yes, inform AP to challenge data ownership
x_2, pk_2
Ownership Challenge AP randomly chooses c
c
Check $0 \le c \le 2^\sigma - 1$; Compute $y = H(M) + (s_2 \times c)$; Encrypt y with pk_{AP}
$E(pkAP, y), V_2$
Obtain y; check if $H(y * P + c * V_2) =? x_2$.If yes, generate $rk_{AP\to u2}$
$rk_{AP\to u2}$
Deduplication CSP re-encrypts CK_1 with $rk_{AP\to u2}$, get $E(pk_2, DEK_1)$
$E(pk_2, DEK_1)$
Get DEK_1 with sk_2 to access CT_1

Fig. 3. A procedure of data deduplication.

Algorithm 1. Grant Access to Duplicated Data

Input: pk_j, $Policy(u_i)$, $Policy(AP)$

- CSP requests AP to challenge ownership and grant access to duplicated data for u_j by providing pk_j.
- After ensuring data ownership through challenge, AP checks $Policy(AP)$ and issues CSP $rk_{AP\to u_i} = RG\ (pk_{AP}; sk_{AP}; pk_j)$ if the check is positive.
- CSP transforms $E(pk_{AP}; DEK_i)$ into $E(pk_j; DEK_i)$ if $Policy(u_i)$ authorizes u_j to share the same data M encrypted by DEK_i: $R(rk_{AP\to u_i}; E(pk_{AP}; DEK)) = E(pk_j;\ DEK_i)$.

Note: $rk_{AP\to u_i}$ calculation can be skipped if it has been executed already and the value of $(pk_j; sk_j)$ and $(pk_{AP}; sk_{AP})$ remain unchanged.

- Data holder u_j obtains DEK_i by decrypting $E(pk_j;\ DEK_i)$ with sk_j: $DEK_i = D(sk_j; E(pk_j;\ DEK_i))$, and then it can access data M at CSP.

CSP operates as the proxy in the proposed scheme. It indirectly distributes DEK_i for data decryption to duplicated data holders without learning anything about these secrets (e.g., DEK_i and M). Note that AP itself is not allowed by the CSP to access the user's personal data due to business reasons.

4.3 Procedures

4.3.1 Data Deduplication

Fig. 3 illustrates the procedure of data deduplication at CSP with the support of AP based on the proposed scheme. We suppose that user u_1 saves its sensitive data M at CSP with protection using DEK_1, while user u_2 is a data holder who tries to save the same data at CSP. The detailed procedure of data deduplication is presented below:

Step 1 – System setup: as described in Section 3.

Step 2 – Data token generation: User u_1 generates data token of M, $x_1 = H(H(M) \times P)$ and sends $\{x_1,\ pk_1, Cert(pk_1)\}$ to CSP.

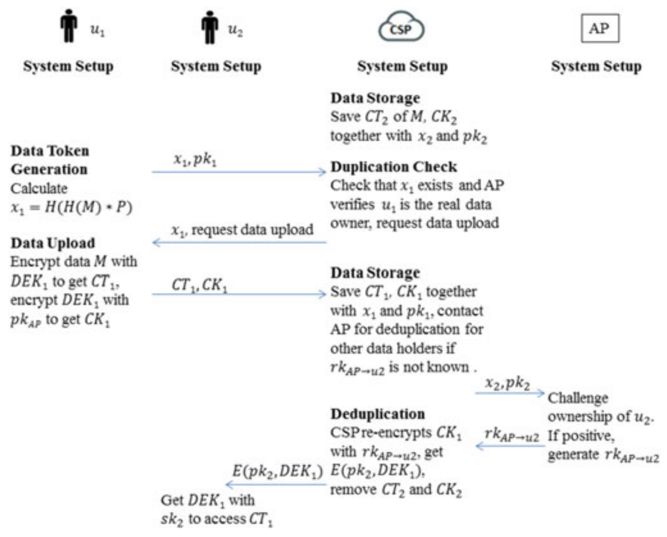

u_1
u_2
CSP
AP
System Setup
System Setup
System Setup
System Setup
Data Storage
Save CT_2 of M, CK_2 together with x_2 and pk_2
Data Token Generation
Calculate $x_1 = H(H(M) * P)$
x_1, pk_1
Duplication Check
Check that x_1 exists and AP verifies u_1 is the real data owner, request data upload
x_1, request data upload
Data Upload
Encrypt data M with DEK_1 to get CT_1, encrypt DEK_1 with pk_{AP} to get CK_1
CT_1, CK_1
Data Storage
Save CT_1, CK_1 together with x_1 and pk_1, contact AP for deduplication for other data holders if $rk_{AP\rightarrow u2}$ is not known .
x_2, pk_2
Challenge ownership of u_2. If positive, generate $rk_{AP\rightarrow u2}$
Deduplication
CSP re-encrypts CK_1 with $rk_{AP\rightarrow u2}$, get $E(pk_2, DEK_1)$, remove CT_2 and CK_2
$rk_{AP\rightarrow u2}$
$E(pk_2, DEK_1)$
Get DEK_1 with sk_2 to access CT_1

Fig. 4. A procedure of data owner management.

Step 3 – Duplication check: CSP verifies $Cert(pk_1)$ and checks if the duplicated data is stored by finding whether x_1 exists. If the check is negative, it requests data upload. User u_1 encrypts data M with DEK_1 to get CT_1 and encryptd DEK_1 with pk_{AP} to get CK_1. u_1 sends CT_1 and CK_1 to CSP, which saves them together with x_1 and pk_1. If the check is positive and the pre-stored data is from the same user, it informs the user about this situation. If the same data is from a different user, refer to Step 6 for deduplication.

Step 4 – Duplicated data upload and check: User u_2 later on tries to save the same data M at CSP following the same procedure of Step 2 and 3. That is, u_2 sends the data package $\{x_2,\ pk_2, Cert(pk_2)\}$ to CSP. Duplication happens because x_2 exists, so CSP forwards $\{x_2,\ pk_2, Cert(pk_2)\}$ to AP.

Step 5 – Ownership challenge: AP challenges the data ownership of u_2 by randomly choosing $c \in_R \{0, \ldots, 2^\sigma - 1\}$ and sending it to u_2. u_2 checks c to make sure that $0 \leq c \leq 2^\sigma - 1$, computes $y = H(M) + (s_2 \times c)$ and sends $E(pk_{AP}, y)$ to AP. AP gets y, computes $H(yP + cV_2)$ and compares it with x_2. If $H(yP + cV_2) = x_2$, i.e., the ownership challenge is successful, AP generates re-encryption key $rk_{AP\rightarrow u_2}$ by calling $RG(pk_{AP}; sk_{AP}; pk_2)$ if it has not been generated and issued to CSP.

Step 6 – Deduplication: CSP re-encrypts $E(pk_{AP}, DEK_1)$ by calling $R(rk_{AP\rightarrow u_2}; E(pk_{AP}; DEK_1)) = E(pk_2; DEK_1)$ and provides the re-encrypted key $E(pk_2, DEK_1)$ to u_2. Then u_2 can get DEK_1 with its secret key sk_2. u_2 confirms the success of data deduplication to CSP that records corresponding deduplication information in the system after getting this notification.

At this moment, both u_1 and u_2 can access the same data M saved at CSP. User u_1 uses DEK_1 directly, while u_2 gets to know DEK_1 by calling $D(sk_2; E(pk_2; DEK_1))$.

4.3.2 Data Deletion at CSP

When data holder u_2 wants to delete the data from CSP, it sends deletion request to CSP: $Cert(pk_2)$, x_2. CSP checks the validity of the request, then removes deduplication record of u_2, and block u_2's later access to M. CSP further checks if the deduplication record is empty. If yes, it deletes encrypted data CT and related records.

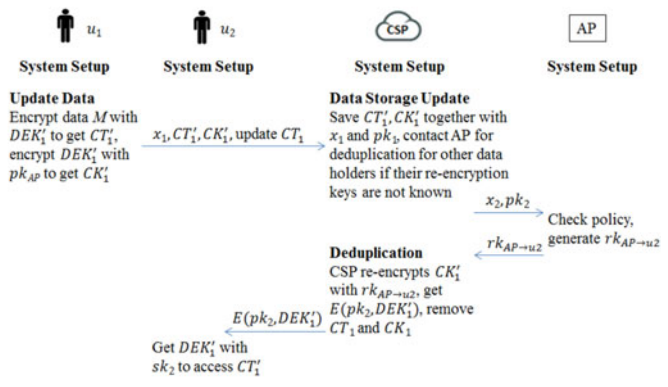

u_1
u_2
CSP
AP
System Setup
System Setup
System Setup
System Setup
Update Data
Encrypt data M with DEK_1' to get CT_1', encrypt DEK_1' with pk_{AP} to get CK_1'
x_1, CT_1', CK_1', update CT_1
Data Storage Update
Save CT_1', CK_1' together with x_1 and pk_1, contact AP for deduplication for other data holders if their re-encryption keys are not known
x_2, pk_2
Check policy, generate $rk_{AP\rightarrow u2}$
$rk_{AP\rightarrow u2}$
Deduplication
CSP re-encrypts CK_1' with $rk_{AP\rightarrow u2}$, get $E(pk_2, DEK_1')$, remove CT_1 and CK_1
$E(pk_2, DEK_1')$
Get DEK_1' with sk_2 to access CT_1'

Fig. 5. A procedure of encrypted data update.

4.3.3 Data Owner Management

In case that real data owner u_1 uploads the data later than data holder u_2, CSP can manage to save the data encrypted by the real data owner at the cloud and allow it to share the storage. The real data ownership can be verified after challenging, e.g., the data owner should provide a specific certificate to show its ownership. The procedure of data owner management is shown in Fig. 4. In this case, CSP contacts AP by providing all data holders' pk_i (e.g., pk_2) if CSP does not know its corresponding re-encryption key $rk_{AP\rightarrow u_i}$ (e.g., $rk_{AP\rightarrow u_2}$). AP issues $rk_{AP\rightarrow u_i}$ to CSP if ownership challenge is positive. CSP re-encrypts CK_1, gets re-encrypted DEK_1 (e.g., $E(pk_2, DEK_1)$), sends it to all related data holders (e.g., u_2), deletes CT_2 and CK_2 by replacing it with u_1's encrypted copy CT_1 and CK_1, and finally updates corresponding deduplication records.

4.3.4 Encrypted Data Update

In some cases, a data holder could update encrypted data stored at CSP by generating a new DEK' and upload the newly encrypted data with DEK' to CSP. As illustrated in Fig. 5, u_1 wants to update encrypted data stored at CSP with new symmetric key DEK_1'. User u_1 sends an update request: $\{x_1, CT_1', CK_1', update\ CT_1\}$. CSP saves CT_1', CK_1' together with x_1 and pk_1. CSP contacts AP for deduplication for other data holders if their re-encryption keys are not known. AP checks its policy for generating and sending corresponding re-encryption keys (e.g., $rk_{AP\rightarrow u2}$), which are used by CSP to perform re-encryption on CK_1' for generating re-encrypted keys that can be decrypted by all eligible data holders (e.g., $E(pk_2, DEK_1')$). The re-encrypted keys are then sent to the eligible data holders for future access on data M. Any data holder can perform the encrypted data update. Based on storage policy and service agreement between the data holder and CSP, CSP decides if such an update can be performed.

4.3.5 Valid Data Replication

As stated above, the savings through deduplication can be passed back directly or indirectly to cloud users, which can help them save storage costs. But sometimes data holders or owners do not care about the storage costs, but want to hold

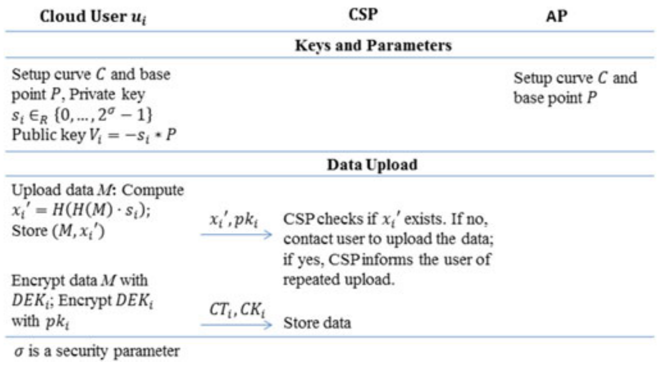

Cloud User u_i
CSP
AP
Keys and Parameters
Setup curve C and base point P, Private key $s_i \in_R \{0, \ldots, 2^{\sigma}-1\}$ Public key $V_i = -s_i * P$
Setup curve C and base point P
Data Upload
Upload data M: Compute $x_i{}' = H(H(M) \cdot s_i)$; Store $(M, x_i{}')$
$x_i{}', pk_i$
CSP checks if $x_i{}'$ exists. If no, contact user to upload the data; if yes, CSP informs the user of repeated upload.
Encrypt data M with DEK_i; Encrypt DEK_i with pk_i
CT_i, CK_i
Store data
σ is a security parameter

Fig. 6. A procedure of valid replicated data upload.

the full control over their data. Hence, they upload and store their own data at CSP, even when it has been uploaded by other entities. As illustrated in Fig. 6, the user u_i generates a token with secret key s_i for deduplication check of its own uploaded data. If it has not been stored by u_i, then u_i encrypts its data with DEK_i and encrypts DEK_i with pk_i. Different from the operation in deduplication, u_i generates CK_i with pk_i rather than pk_{AP}. Thus u_i can control its own data fully, such as data sharing, data deletion and data update. For example, u_i can generate $rk_{u_i \to u_j}$ for sharing the data with u_j.

5 Security Analysis and Performance Evaluation

5.1 Security Analysis

Our scheme provides a secure approach to protect and deduplicate the data stored in cloud by concealing plaintext from both CSP and AP. The security of the proposed scheme is ensured by PRE theory, symmetric key encryption and ECC theory.

Proposition 1. *The cooperation of CSP and AP without collusion guarantees that only eligible users can access plain data M and the data can be deduplicated in a secure way.*

Proof. CSP has no way to know M since it is always in an encrypted form. CSP knows CK encrypted with pk_{AP}, but AP does not share its own secret key sk_{AP} with CSP. Thus CSP cannot know DEK and then M. AP has no way to access M since its access is blocked by CSP although AP could obtain DEK. In addition, we apply proper management protocols to support data storage management and data owner management to achieve deduplication at the same time. □

Data confidentiality is achieved by symmetric key encryption (e.g., AES) and PRE. Data M could be disclosed in two different ways. One is to obtain it from $H(H(M) \times P)$. The hash function is assumed to be secure, i.e., it is difficult to find collision attack. Therefore, M is impossible to be obtained through the hash code. Another is to disclose M by breaking the ciphertext of symmetric key encryption $CT = Encrypt(DEK, M)$. Assuming that the symmetric key encryption is secure, DEK becomes the key point of data security. In the proposed scheme, DEK is encrypted with pk_{AP} through PRE as $E(pk_{AP}, DEK)$. Because CSP and AP would not collude with each other, CSP cannot gain DEK to get M because it knows nothing about sk_{AP} although it stores CK and CT. The re-encryption with $pk_{AP \to u_i}$ transforms the cipherkey under pk_{AP} into the cipherkey under pk_i. Similarly, CSP can get nothing from the cipherkey for knowing nothing about sk_i. The authorized user u_i can obtain the data M by decrypting $E(pk_i, DEK)$ with its own secret key sk_i, which can guarantee that M is secure and can be only accessed by eligible users. About a security proof on PRE, refer to [27].

In data replication, data are encrypted with the secret key of data holder u_i. Thus it can only be obtained with sk_i. The re-encryption key for anyone can only be issued by u_i rather than AP, which guarantees its full control over the data. In addition, only u_i can generate the token $x_i{}'$ with its own secret key s_i. Thus no one can guess its content and produce the same $x_i{}'$ even though it holds the same data. Thus, applying $x_i{}'$ for deduplication check can support valid data replication for an individual user. Similar to the analysis above, the data confidentiality is also achieved by symmetric key encryption and PRE.

Proposition 2. *$H(M)$that is the key to pass the duplication check cannot be obtained by malicious cloud users.*

Proof. $H(H(M) \times P)$ that plays as the token for data duplication check cannot reveal the value of $H(M) \times P$, especially $H(M)$ due to Elliptic curve discrete logarithm problem. Hence, even when the token is intercepted by some malicious users, $H(M)$ is still protected. CSP cannot get any information except the token. Though AP can get to know the value $y = H(M) + (s_i \times c)$ and the challenge c, it cannot obtain s_i and $H(M)$. Thus, neither CSP nor AP can obtain $H(M)$. □

Based on the above security analysis, $H(M)$ is not transmitted between the involved parties and cannot be obtained by any other parties. Even when CSP colludes with a malicious cloud user, the malicious user can only obtain the token $H(H(M) \times P)$, not $H(M)$, thus impossible to access M stored at cloud by passing the ownership challenge raised by AP.

Proposition 3. *To pass the ownership verification of AP, a cloud user must indeed have data M.*

Proof. With real data M, user u_i can generate correct $H(M)$, compute right $y = H(M) + (s_i \times c)$ with s_i and the challenge c provided by AP, thus AP can successfully compare that $H(yP + cV_i) = H(H(M) \times P + (s_i \times c) \times P - c \times s_i \times P) = H(H(M) \times \mathrm{P}) = x_i$ is equal to x_i, i.e., passing the ownership verification of AP. □

5.2 Computation Complexity

The proposed scheme involves four kinds of system roles: data owner, data holder, CSP and AP. To present the computation complexity in details, we adopt AES for symmetric encryption, ECC and PRE proposed in [8]. We analyze the complexity of uploading one data file as below.

Data Owner. Regarded as the first data uploader, it is in charge of four operations: system setup, data encryption, key encryption, and token $H(H(M) \times P)$ generation. In setup, the key generation of PRE includes 1 exponentiation. The ECC key generation needs one point multiplication. In

TABLE 2
Computation Complexity of System Entities by Comparing with [33]

Entity	Algorithm	Computations [our scheme]	Computations [33]	Complexity
Data Owner	Setup	1*PointMulti + 1* Exp	1*ModInv + 1*Exp	$\mathcal{O}(1)$
	Data upload	2 *Exp + 1*PointMulti	3*Exp	
CSP	Re-encryption	1 *Pair	1*Pair	$\mathcal{O}(n)$
Data Holder	System Setup	1*PointMulti + 1*Exp	1*ModInv + 1* Exp	$\mathcal{O}(1)$
	Challenge Response	2*Exp + 1*PointMulti	–	
	Data upload	–	3*Exp	
	Decryption for access	1*Exp	1*Exp	
AP	System Setup	1*Exp	1*Exp	$\mathcal{O}(n)$
	Ownership challenge and re-encryption key generation	2*Exp + 2*Point Multi	1*Exp	

Notes: Pair: Bilinear Pairing; Exp: Exponentiation; ModInv: Modular Inversion; n*: Number of data holders who share the same data; PointMulti: Point multiplication in ECC.*

addition, system setup takes only once for all data storage operations. The computation complexity of encrypting data using DEK depends on the size of data, which is inevitable in any cryptographic methods for protecting the data. Likewise, the computation complexity of hash depends on the size of data, but it is very fast, which can be ignored. The encryption of DEK using PRE need 2 exponentiations. The first step of data upload for deduplication check involves one token generation, which needs two hashes and one point multiplication. Thus, the computation complexity of data owner is $\mathcal{O}(1)$ at both setup and data upload.

CSP. A user uploads its data to CSP by sending token $H(H(M) \times P)$. CSP should first check if the same token has existed (by comparing the token with the records in CSP, which is inevitable in any deduplication schemes). Then, CSP chooses to save the data if the token does not exist. If the data holder uploads the same data, CSP contacts AP for gaining a re-encryption key if the ownership challenge is positive. In this case, CSP has to finish the re-encryption operation of PRE, which requires 1 pairing. If the same data is uploaded by n data holders, the computational complexity is $\mathcal{O}(n)$. CSP is responsible for allowing the access to the same data for all data holders by avoiding storing the same data in the cloud.

TABLE 3
Test Environments

Hardware Environment	CPU: Intel Core 2 Quad CPU Q9400 2.66 GHZ
	Memory: 4 GB SDRAM
Software Environment	Operating System: Ubuntu v14.04, Windows Home Ultimate editions 64 bit
	Programming Environment: Eclipse Luna CDT, Java
	Secure Protocol: OpenSSL
	Library: Pairing Based Cryptography (http://crypto.stanford.edu/pbc/)
	Database: MySQL v5.5.41

TABLE 4
Operation Time of Each Step in Data Ownership Challenge

	Operation	Time (millisecond)
Cloud User	Credential initiation	0.5
	Data token generation	0.6
	Data upload (10 MB data)	149.8
	Challenge response	4.31
AP	Ownership challenge verification	1.2

Data Holder. When data holder u_i uploads the same data that has been stored in CSP, it generates token $H(H(M) \times P)$ as the data owner has done, which needs one point multiplication. In addition, the data holder has to compute $y = H(M) + (s_i \times c)$ during ownership challenge, perform $E(pk_{AP}, y)$ in order to protect $H(M)$ from disclosure in case y is known by a malicious party, and conduct one more decryption for accessing the data, which involves 2 exponentiations in $E(pk_{AP}, y)$ and 1 exponentiation in $D(pk_i, DEK)$. Note: the data holder has no need to encrypt the original data for data upload. The computational complexity of a data holder is $\mathcal{O}(1)$.

AP. AP is responsible for the re-encryption key management. It challenges data ownership by randomly selecting c, decrypting y and comparing $H(yP + cV_i)$ with x_i. It checks the policy and issues the re-encryption key for authorized user by conducting two point multiplications. The decryption of y needs 1 exponentiation. The re-encryption key generation needs 1 exponentiation. AP needs to issue keys for all authorized data holders that upload the same data. Thus, the computational complexity of AP is $\mathcal{O}(n)$. Notably, if the re-encryption key of a data holder has been generated and issued already, AP will only authorize CSP to perform re-encrytion on CK and will not re-generate the re-encryption key any more.

We should note that the computational burden of system setup, especially the generation of key pairs, can be amortized over the lifetime of entities. Thus, our scheme is very efficient. Table 2 lists the computation operations and complexity of different roles. We also compare the scheme presented in this paper with our previous work [33]. We can see that our scheme is more efficient at the side of data owners and holders than [33] regarding big data deduplication because the point multiplication is more efficient than the exponentiation operation, also refer to Tables 4, 5, and 6. Especially, data holders have no need to encrypt and upload data, which can save much time and bandwidth. Our scheme only introduces a bit more computation complexity at AP. However, AP is a powerful server full of computation capability. Thus additional computation load at AP is acceptable.

TABLE 5
Operation Time of Each Basic Operation in Our Proposed Scheme and [33]

BasicOperation	Pair	Exp(G_1)	Exp(G_T)	ModInv	PointMulti
Time (millisecond)	5.07	3.66	0.55	0.003	0.6

TABLE 6
Total Operation Time of Cloud Users in Our Proposed Scheme and [33] (Unit: millisecond)

Entity	Total time [our scheme]	Total time [33]
Data Owner	150.4	151.2
Data Holder	4.91	151.2

5.3 Communication Cost

The extra communication cost introduced by the proposed scheme is trivial. Refer to Fig. 3, extra cost introduced by our scheme is $\{2x_i,\ pk_i\}$ during data uploading and storage. Its size is 1344 bits if using SHA-1 hash function. Data deduplication also introduces some extra communication cost: $2\{x_i,\ pk_i\}$, c, $E(pk_{AP}, y)$, $rk_{AP\rightarrow u_i}$, $E(pk_i, DEK)$ and V_i. The size is 5984 bits if DEK size is 256 bits and challenge number c is 160 bits. We can see that the communication cost of our scheme is very light and it is not influenced by the size of uploaded data. Thus, the proposed scheme is suitable for supporting big data deduplication with regard to communication cost.

5.4 Performance Evaluation

5.4.1 Implementation and Testing Environment

We implemented the proposed scheme and tested its performance. Table 3 describes our implementation and testing environments. We applied a MySQL database to store data files and related information. In our test, we did not take into account the time of data uploading and downloading. We focused on testing the performance of the deduplication procedure and algorithms designed in our scheme.

5.4.2 Efficiency Test

Test 1: Efficiency of data encryption and decryption

In this experiment, we tested the operation time of data encryption and decryption with AES by applying different AES key sizes (128 bits, 196 bits and 256 bits) and different data size (from 10 megabytes to 600 megabytes). The testing environment was Intel Core i5-3337U CPU 1.80 GHz 4.00 GB RAM, Ubuntu v13.10 2.0 GB RAM, Dual-Core processor, 25.0G Hard disk. As shown in Fig. 7, we observed that even when the data is as big as 600 MB, the encryption/decryption time is less than 13 seconds if applying 256-bit AES key. Applying symmetric encryption for data protection is a reasonable and practical choice. The time spent on AES encryption and decryption is increased with the size of data. This is inevitable in any encryption schemes. Since AES is very efficient on data encryption and decryption, thus it is practical to be applied for big data.

Test 2: Efficiency of PRE

We tested the efficiency of each operation of 1024-bit PRE with different sizes of AES symmetric keys (128 bits, 196 bits and 256 bits). Fig. 8 shows the operation time that is the average time of 500 tests. We observe that our scheme is very efficient. The time spent for PRE key pair generation (KeyGen), re-encryption key generation (ReKeyGen), encryption (Enc), re-encryption (ReEnc) and decryption (Dec) is not related to the length of an input key. For the tested three AES key sizes, the encryption time is less than 5 milliseconds. The decryption time is about 1 millisecond,

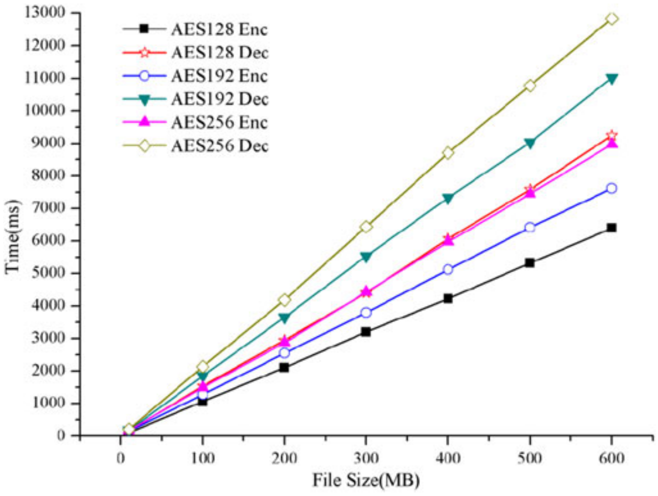


Fig. 7. Operation time of file encryption and decryption with AES.

which implies that our scheme does not introduce heavy processing load to data owners and holders. We also observe that the computation time of each operation does not vary too much with the different length of AES key size. Therefore, our scheme can be efficiently adapted to various security requirements in various scenarios. Obviously, our scheme used for deduplication does not introduce much computation cost. In particular, the PRE related operations for deduplication are not influenced by the size of stored data. This fact implies that the proposed scheme is similarly efficient with regard to different sizes of big data deduplication. This result shows the significance and practical potential of our scheme to support big data storage and deduplication.

Test 3: Data Ownership Challenge

In this experiment, we selected 192-bit field of elliptic curve (160-bit ECC has a security level comparable to 1024-bit RSA), 256-bit AES, 1024-bit PRE and 10M uploaded data. We tested the operation time of each step of data ownership verification as presented in Fig. 2. Obviously, the duplication check operated by CSP depends on the size of the storage. AP initiates the ownership challenge with a randomly chosen number. Hence, we neglect these two steps and

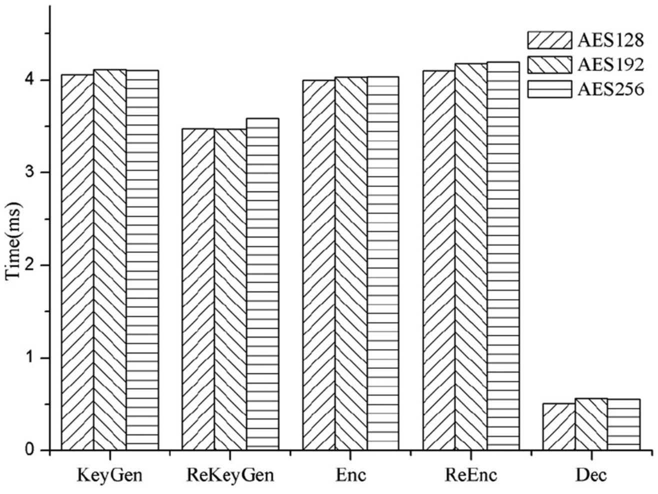
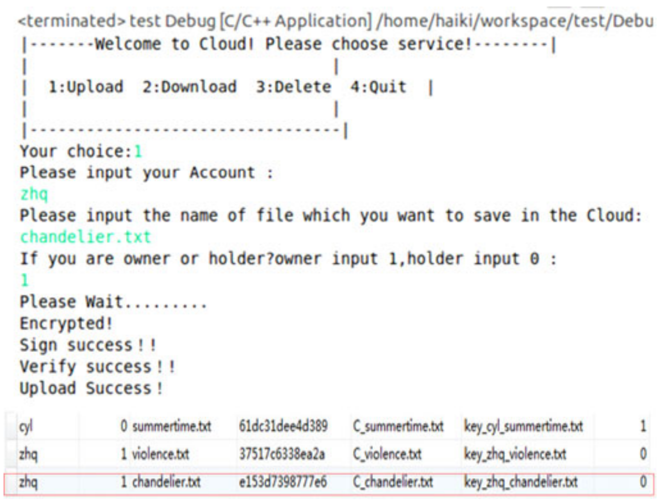


Fig. 8. The execution time of PRE operations.



```
<terminated> test Debug [C/C++ Application] /home/haiki/workspace/test/Debu
|-------Welcome to Cloud! Please choose service!--------|
|                                      |
|  1:Upload  2:Download  3:Delete  4:Quit  |
|                                      |
|--------------------------------|
Your choice:1
Please input your Account :
zhq
Please input the name of file which you want to save in the Cloud:
chandelier.txt
If you are owner or holder?owner input 1,holder input 0 :
1
Please Wait.........
Encrypted!
Sign success!!
Verify success!!
Upload Success!
```

cyl	0	summertime.txt	61dc31dee4d389	C_summertime.txt	key_cyl_summertime.txt	1
zhq	1	violence.txt	37517c6338ea2a	C_violence.txt	key_zhq_violence.txt	0
zhq	1	chandelier.txt	e153d7398777e6	C_chandelier.txt	key_zhq_chandelier.txt	0

Fig. 9. New file upload process.

mainly focus on the computations operated by cloud users and AP, which are presented in Table 4.

We can observe that data upload is the most time-consuming if the file is big, but it is inevitable in all schemes. Compared with [33], the users do not need to process data encryption if another user has uploaded the data already. Therefore, our scheme can save a lot of computation load and communication cost for cloud users. In addition, the data ownership challenge in the proposed scheme is very lightweight, which does not involve much burden to cloud users. This implies that our scheme is very efficient and very suitable for deduplication check on big data.

Test 4: Comparison with [33]

We have analyzed the computation complexity of our scheme in Section 5.2. Here, we further compare its efficieny with [33] by presenting the basic operation time in Table 5. For ease presentation and comparison, the notations are inherited from Table 2.

We can observe that the point multiplication is much more efficient than exponentiation over the group G_1 and approximate to that over G_T, which further proves that our scheme is efficient and does not introduce new overhead.

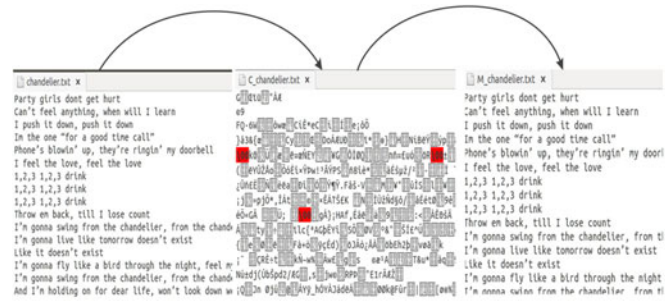
As the time to setup system can be amortized over the lifetime of cloud users, we only compare their time to upload and store data. Based on the implementation results and analysis (1024-bit PRE, 1024-bit RSA [46], and 256-bit AES), we present the time for cloud users to upload 10 MB data in this proposed scheme and [33] respecitively, which is shown in Table 6. We can observe that our scheme has great advantage. This scheme saves much time for those data holders who upload duplicated data.

5.4.3 Functional Test

We set up two users (User 1: zhq; User 2: cyl) in the system for testing purpose in order to illustrate that the designed scheme works well with correct functional implementation.

Case 1: User zhq uploads a new file named chandelier.txt.

User zhq encrypts the encryption key DEK with pk_{AP} and generates key file named as key_zhq_chandelier.txt, As the cloud has not yet stored this data file, the data file is encrypted and successfully uploaded to the cloud as an



```
Party girls dont get hurt
Can't feel anything, when will I learn
I push it down, push it down
In the one "for a good time call"
Phone's blowin' up, they're ringin' my doorbell
I feel the love, feel the love
1,2,3 1,2,3 drink
1,2,3 1,2,3 drink
1,2,3 1,2,3 drink
Throw em back, till I lose count
I'm gonna swing from the chandelier, from the chand
I'm gonna live like tomorrow doesn't exist
Like it doesn't exist
I'm gonna fly like a bird through the night, feel m
I'm gonna swing from the chandelier, from the chand
And I'm holding on for dear life, won't look down w
```

```
[illegible]
```

```
Party girls dont get hurt
Can't feel anything, when will I learn
I push it down, push it down
In the one "for a good time call"
Phone's blowin' up, they're ringin' my doorl
I feel the love, feel the love
1,2,3 1,2,3 drink
1,2,3 1,2,3 drink
1,2,3 1,2,3 drink
Throw em back, till I lose count
I'm gonna swing from the chandelier, from tl
I'm gonna live like tomorrow doesn't exist
Like it doesn't exist
I'm gonna fly like a bird through the night
I'm gonna swing from the chandelier, from tl
```

Fig. 10. The transformation between plaintext and ciphertext of a file.

encrypted file C_chandelier.txt, shown in Fig. 9. When user zhq tried to access the data, it obtains the plaintext file M_chandelier.txt. Fig. 10 shows the transformation between plaintext and ciphertext.

Case 2: User cyl uploads the same file chandelier.txt.

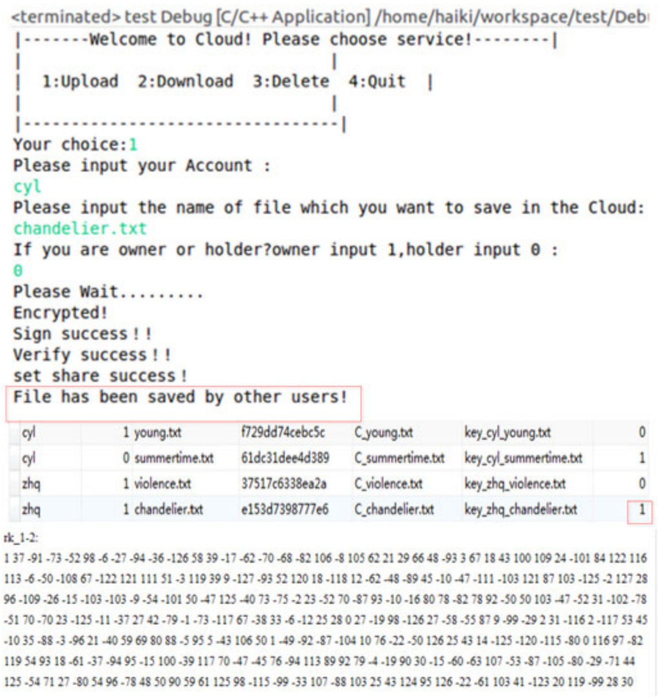
As shown in Fig. 11, CSP found that the file has been uploaded by user zhq through duplication check. Note that no matther what the file name is, CSP can detect duplication if the file content is the same as a file saved already. The CSP would not save the file but ask AP to challenge ownership and issue a re-encryption key to user cyl for accessing the same file. Finally, the AP issues a re-encryption key $rk_{1\rightarrow 2}$ to allow user cyl to share the file chandelier.txt that has been stored already.

Case 3: User cyl accesses the shared file chandelier.txt

User zhq uploaded chandelier.txt and denoted as data owner. User cyl uploaded the same file. Now it tried to access this shared file. As shown in Fig. 12, CSP and AP allow cyl to access the pre-uploaded file. Usr cyl can download the file and obtain the data.

Case 4: Data owner management

User cyl has uploaded file summertime.txt as a data holder. When the data owner zhq uploads this file, CSP

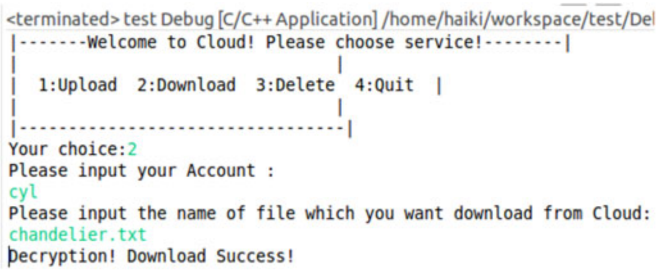


```
<terminated> test Debug [C/C++ Application] /home/haiki/workspace/test/Deb
|-------Welcome to Cloud! Please choose service!--------|
|                                      |
|  1:Upload  2:Download  3:Delete  4:Quit  |
|                                      |
|--------------------------------|
Your choice:1
Please input your Account :
cyl
Please input the name of file which you want to save in the Cloud:
chandelier.txt
If you are owner or holder?owner input 1,holder input 0 :
0
Please Wait.........
Encrypted!
Sign success!!
Verify success!!
set share success!
File has been saved by other users!
```

cyl	1	young.txt	f729dd74cebc5c	C_young.txt	key_cyl_young.txt	0
cyl	0	summertime.txt	61dc31dee4d389	C_summertime.txt	key_cyl_summertime.txt	1
zhq	1	violence.txt	37517c6338ea2a	C_violence.txt	key_zhq_violence.txt	0
zhq	1	chandelier.txt	e153d7398777e6	C_chandelier.txt	key_zhq_chandelier.txt	1

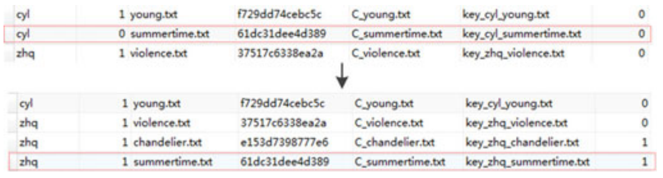
```
rk_1-2:
1 37 -91 -73 -52 98 -6 -27 -94 -36 -126 58 39 -17 -62 -70 -68 -82 106 -8 105 62 21 29 66 48 -93 3 67 18 43 100 109 24 -101 84 122 116
113 -6 -50 -108 67 -122 121 111 51 -3 119 39 9 -127 -93 52 120 18 -118 12 -62 -48 -89 45 -10 -47 -111 -103 121 87 103 -125 -2 127 28
96 -109 -26 -15 -103 -103 -9 -54 -101 50 -47 125 -40 73 -75 -2 23 -52 70 -87 93 -10 -16 80 78 -82 78 92 -50 50 103 -47 -52 31 -102 -78
-51 70 -70 23 -125 -11 -37 27 42 -79 -1 -73 -117 67 -38 33 -6 -12 25 28 0 27 -19 98 -126 27 -58 -55 87 9 -99 -29 2 31 -116 2 -117 53 45
-10 35 -88 -3 -96 21 -40 59 69 80 88 -5 95 5 -43 106 50 1 -49 -92 -87 -104 10 76 -22 -50 126 25 43 14 -125 -120 -115 -80 0 116 97 -82
119 54 93 18 -61 -37 -94 95 -15 100 -39 117 70 -47 -45 76 -94 113 89 92 79 -4 -19 90 30 -15 -60 -63 107 -53 -87 -105 -80 -29 -71 44
125 -54 71 27 -80 54 96 -78 48 50 90 59 61 125 98 -115 -99 -33 107 -88 103 25 43 124 95 126 -22 -61 103 41 -123 20 119 -99 28 30
```

Fig. 11. Reject duplicated saving and issue a re-encryption key.



```
<terminated> test Debug [C/C++ Application] /home/haiki/workspace/test/Del
|-------Welcome to Cloud! Please choose service!--------|
|                                |
|  1:Upload  2:Download  3:Delete  4:Quit  |
|                                |
|--------------------------------|
Your choice:2
Please input your Account :
cyl
Please input the name of file which you want download from Cloud:
chandelier.txt
Decryption! Download Success!
```

Fig. 12. Access shared duplicated data.



cyl	1	young.txt	f729dd74cebc5c	C_young.txt	key_cyl_young.txt	0
cyl	0	summertime.txt	61dc31dee4d389	C_summertime.txt	key_cyl_summertime.txt	0
zhq	1	violence.txt	37517c6338ea2a	C_violence.txt	key_zhq_violence.txt	0

↓

cyl	1	young.txt	f729dd74cebc5c	C_young.txt	key_cyl_young.txt	0
zhq	1	violence.txt	37517c6338ea2a	C_violence.txt	key_zhq_violence.txt	0
zhq	1	chandelier.txt	e153d7398777e6	C_chandelier.txt	key_zhq_chandelier.txt	1
zhq	1	summertime.txt	61dc31dee4d389	C_summertime.txt	key_zhq_summertime.txt	1

Fig. 13. Data owner management.

removes the data record of cyl and replaces it with the corresponding record of zhq, as shown in Fig. 13.

Case 5: Data deletion

User zhq is the data owner of the file chandelier.txt. If user cyl as a data holder wants to delete the file, CSP just blocks the access of cyl, as shown in Fig. 14. But if the data owner zhq wants to delete the file, CSP needs to delete the record of zhq but keep the record of cyl. CSP blocks the access of zhq to the file, as shown in Fig. 15.

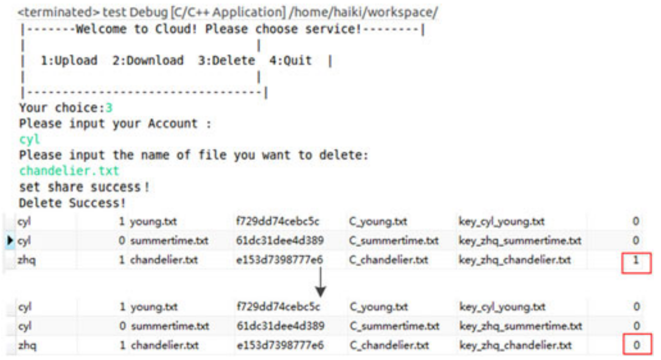
5.5 Further Discussions

The proposed scheme has the following additional advantages.

Flexibility. The proposed scheme can flexibly support access control on encrypted data with deduplication. One data holder can flexibly update DEK. The new key can be easily issued to other data holders or eligible data users by CSP with a low cost, especially when AP has issued the re-encryption key already. Data revocation can be realized by blocking data access at CSP and rejecting key re-encryption on a newly applied key DEK'. The detailed process of data revocation is described in [34]

Low Cost of Storage. The scheme can obviously save the storage space of CSP since it only stores one copy of the same data that is shared by data owner and data holders. Storing deduplication records occupies some storage or memory for saving token pk_i and x_i (only $1024 + 160$ bits). But comparing with the big volume of duplicated data, this storage cost can be ignored.

Big Data Support. The proposed scheme can efficiently perform big data deduplication. First, duplicated big data upload is efficient because only x_i and pk_i are sent to CSP. CSP performs hash comparison and then contacts AP to challenge ownership for issuing a re-encryption key. The computation and communication cost of this process (involving ownership challenge, re-encryption key generation, CK re-encryption and re-encrypted key decryption) is not influenced by the size of big data. Second, uploading ciphertext CT is inevitable in almost all schemes for deduplication. The proposed scheme only introduces a bit extra communication load (i.e., CK) and a little bit additional communication cost for ownership challenge. Compared with big data upload cost and storage cost, they are very trivial and efficient.



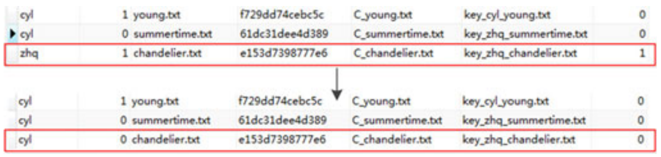
```
<terminated> test Debug [C/C++ Application] /home/haiki/workspace/
|-------Welcome to Cloud! Please choose service!--------|
|                                |
|  1:Upload  2:Download  3:Delete  4:Quit  |
|                                |
|--------------------------------|
Your choice:3
Please input your Account :
cyl
Please input the name of file you want to delete:
chandelier.txt
set share success!
Delete Success!
```

cyl	1	young.txt	f729dd74cebc5c	C_young.txt	key_cyl_young.txt	0
cyl	0	summertime.txt	61dc31dee4d389	C_summertime.txt	key_zhq_summertime.txt	0
zhq	1	chandelier.txt	e153d7398777e6	C_chandelier.txt	key_zhq_chandelier.txt	1

↓

cyl	1	young.txt	f729dd74cebc5c	C_young.txt	key_cyl_young.txt	0
cyl	0	summertime.txt	61dc31dee4d389	C_summertime.txt	key_zhq_summertime.txt	0
zhq	1	chandelier.txt	e153d7398777e6	C_chandelier.txt	key_zhq_chandelier.txt	0

Fig. 14. Data deletion by a data holder.



cyl	1	young.txt	f729dd74cebc5c	C_young.txt	key_cyl_young.txt	0
cyl	0	summertime.txt	61dc31dee4d389	C_summertime.txt	key_zhq_summertime.txt	0
zhq	1	chandelier.txt	e153d7398777e6	C_chandelier.txt	key_zhq_chandelier.txt	1

↓

cyl	1	young.txt	f729dd74cebc5c	C_young.txt	key_cyl_young.txt	0
cyl	0	summertime.txt	61dc31dee4d389	C_summertime.txt	key_zhq_summertime.txt	0
cyl	0	chandelier.txt	e153d7398777e6	C_chandelier.txt	key_zhq_chandelier.txt	0

Fig. 15. Data deletion by a data owner.

6 Conclusion

Managing encrypted data with deduplication is important and significant in practice for achieving a successful cloud storage service, especailly for big data storage. In this paper, we proposed a practical scheme to manage the encrypted big data in cloud with deduplication based on ownership challenge and PRE. Our scheme can flexibly support data update and sharing with deduplication even when the data holders are offline. Encrypted data can be securely accessed because only authorized data holders can obtain the symmetric keys used for data decryption. Extensive performance analysis and test showed that our scheme is secure and efficient under the described security model and very suitable for big data deduplication. The results of our computer simulations further showed the practicability of our scheme. Future work includes optimizing our design and implementation for practical deployment and studying verifiable computation to ensure that CSP behaves as expected in deduplication management.

Acknowledgments

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

References

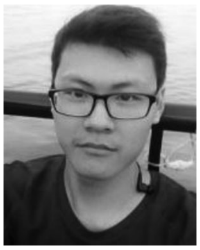
[1] M. Bellare, S. Keelveedhi, and T. Ristenpart, "DupLESS: Server aided encryption for deduplicated storage," in *Proc. 22nd USENIX Conf. Secur.*, 2013, pp. 179–194.

[2] Dropbox, A file-storage and sharing service. (2016). [Online]. Available: http://www.dropbox.com

[3] Google Drive. (2016). [Online]. Available: http://drive.google.com
[4] Mozy, Mozy: A File-storage and Sharing Service. (2016). [Online]. Available: http://mozy.com/
[5] 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 *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2002, pp. 617–624, doi:10.1109/ICDCS.2002.1022312.
[6] G. Wallace, et al., "Characteristics of backup workloads in production systems," in *Proc. USENIX Conf. File Storage Technol.*, 2012, pp. 1–16.
[7] Z. O. Wilcox, "Convergent encryption reconsidered," 2011. [Online]. Available: http://www.mailarchive.com/cryptography@metzdowd.com/msg08949.html
[8] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inform. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, 2006, doi:10.1145/1127345.1127346.
[9] Opendedup. (2016). [Online]. Available: http://opendedup.org/
[10] D. T. Meyer and W. J Bolosky, "A study of practical deduplication," *ACM Trans. Storage*, vol. 7, no. 4, pp. 1–20, 2012, doi:10.1145/2078861.2078864.
[11] J. Pettitt, "Hash of plaintext as key?" (2016). [Online]. Available: http://cypherpunks.venona.com/date/1996/02/msg02013.html
[12] The Freenet Project, Freenet. (2016). [Online]. Available: https://freenetproject.org/
[13] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Proc. Cryptology—EUROCRYPT*, 2013, pp. 296–312, doi:10.1007/978-3-642-38348-9_18.
[14] D. Perttula, B. Warner, and Z. Wilcox-O'Hearn, "Attacks on convergent encryption." (2016). [Online]. Available: http://bit.ly/yQxyvl
[15] C. Y. Liu, X. J. Liu, and L. Wan, "Policy-based deduplication in secure cloud storage," in *Proc. Trustworthy Comput. Serv.*, 2013, pp. 250–262, doi:10.1007/978-3-642-35795-4_32.
[16] P. Puzio, R. Molva, M. Onen, and S. Loureiro, "ClouDedup: Secure deduplication with encrypted data for cloud storage," in *Proc. IEEE Int. Cof. Cloud Comput. Technol. Sci.*, 2013, pp. 363–370, doi:10.1109/CloudCom.2013.54.
[17] Z. Sun, J. Shen, and J. M. Yong, "DeDu: Building a deduplication storage system over cloud computing," in *Proc. IEEE Int. Conf. Comput. Supported Cooperative Work Des.*, 2011, pp. 348–355, doi:10.1109/CSCWD.2011.5960097.
[18] Z. C. Wen, J. M. Luo, H. J. Chen, J. X. Meng, X. Li, and J. Li, "A verifiable data deduplication scheme in cloud computing," in *Proc. Int. Conf. Intell. Netw. Collaborative Syst.*, 2014, pp. 85–90, doi:10.1109/INCoS.2014.111.
[19] 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, May 2015, doi:10.1109/TPDS.2014.2318320.
[20] P. Meye, P. Raipin, F. Tronel, and E. Anceaume, "A secure two-phase data deduplication scheme," in *Proc. HPCC/CSS/ICESS*, 2014, pp. 802–809, doi:10.1109/HPCC.2014.134.
[21] J. Paulo and J. Pereira, "A survey and classification of storage deduplication systems," *ACM Comput. Surveys*, vol. 47, no. 1, pp. 1–30, 2014, doi:10.1109/HPCC.2014.134.
[22] Y.-K. Li, M. Xu, C.-H. Ng, and P. P. C. Lee, "Efficient hybrid inline and out-of-line deduplication for backup storage," *ACM Trans. Storage*, vol. 11, no. 1, pp. 2:1-2:21, 2014, doi:10.1145/2641572.
[23] M. Fu, et al., "Accelerating restore and garbage collection in deduplication-based backup systems via exploiting historical information," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 181–192.
[24] M. Kaczmarczyk, M. Barczynski, W. Kilian, and C. Dubnicki, "Reducing impact of data fragmentation caused by in-line deduplication," in *Proc. 5th Annu. Int. Syst. Storage Conf.*, 2012, pp. 15:1–15:12, doi:10.1145/2367589.2367600.
[25] M. Lillibridge, K. Eshghi, and D. Bhagwat, "Improving restore speed for backup systems that use inline chunk-based deduplication," in *Proc. USENIX Conf. File Storage Technol.*, 2013, pp. 183–198.
[26] L. J. Gao, "Game theoretic analysis on acceptance of a cloud data access control scheme based on reputation," M.S. thesis, Xidian University, State Key Lab of ISN, School of Telecommunications Engineering, Xi'an, China, 2015.
[27] Z. Yan, X. Y. Li, M. J. Wang, and A. V. Vasilakos, "Flexible data access control based on trust and reputation in cloud computing," *IEEE Trans. Cloud Comput.*, vol. PP, no. 99, Aug. 2015, doi:10.1109/TCC.2015.2469662, Art. no. 1.
[28] C. Yang, J. Ren, and J. F. Ma, "Provable ownership of file in deduplication cloud storage," in *Proc. IEEE Global Commun. Conf.*, 2013, pp. 695–700, doi:10.1109/GLOCOM.2013.6831153.
[29] T. Y. Wu, J. S. Pan, and C. F. Lin, "Improving accessing efficiency of cloud storage using de-duplication and feedback schemes," *IEEE Syst. J.*, vol. 8, no. 1, pp. 208–218, Mar. 2014, doi:10.1109/JSYST.2013.2256715.
[30] C. Fan, S. Y. Huang, and W. C. Hsu, "Hybrid data deduplication in cloud environment," in *Proc. Int. Conf. Inf. Secur. Intell. Control*, 2012, pp. 174–177, doi:10.1109/ISIC.2012.6449734.
[31] J. W. Yuan and S. C. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in *Proc. IEEE Int. Conf. Communic. Netw. Secur.*, 2013, pp. 145–153, doi:10.1109/CNS.2013.6682702.
[32] N. Kaaniche and M. Laurent, "A secure client side deduplication scheme in cloud storage environments," in *Proc. 6th Int. Conf. New Technol. Mobility Secur.*, 2014, pp. 1–7, doi:10.1109/NTMS.2014.6814002.
[33] Z. Yan, W. X. Ding, and H. Q. Zhu, "A scheme to manage encrypted data storage with deduplication in cloud," in *Proc. ICA3PP2015*, Zhangjiajie, China, Nov. 2015, pp. 547–561.
[34] Z. Yan, X. Y. Li, and R. Kantola, "Controlling cloud data access based on reputation," *Mobile Netw. Appl.*, vol. 20, no. 6, 2015, pp. 828–839, doi:10.1007/s11036-015-0591-6.
[35] T. T. Wu, W. C. Dou, C. H. Hu, and J. J. Chen, "Service mining for trusted service composition in cross-cloud environment," *IEEE Systems Syst. J.*, vol. PP, no. 99, pp. 1–12, 2014, doi:10.1109/JSYST.2014.2361841.
[36] C. Liu, C. Yang, X. Y. Zhang, and J. J. Chen, "External integrity verification for outsourced big data in cloud and iot: A big picture," *Future Generation Comput. Syst.*, vol. 49, pp. 58–67, 2015.
[37] N. X. Xiong, et al., "Comparative analysis of quality of service and memory usage for adaptive failure detectors in healthcare systems," *IEEE J. Select. Areas Commun.*, vol. 27, no. 4, pp. 495–509, 2009, doi:10.1109/JSAC.2009.090512.
[38] Y. Z. Zhou, Y. X. Zhang, H. Liu, N. X. Xiong, and A. V. Vasilakos, "A bare-metal and asymmetric partitioning approach to client virtualization," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 40–53, Jan.-Mar. 2014, doi:10.1109/TSC.2012.32.
[39] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, 2011, pp. 491–500, doi:10.1145/2046707.2046765.
[40] R. D. Pietro and A. Sorniotti, "Boosting efficiency and security in proof of ownership for deduplication," in *Proc. 7th ACM Symp. Inf. Comput. Commun. Secur.*, 2012, pp. 81–82, doi:10.1145/2414456.2414504.
[41] W. K. Ng, Y. Wen, and H. Zhu, "Private data deduplication protocols in cloud storage," in *Proc 27th Annu. ACM Symp. Appl. Comput.*, 2012, pp. 441–446.
[42] C. W. Tsai, C. F. Lai, H. C. Chao, and A. V. Vasilakos, "Big data analytics: A survey," *J. Big Data*, vol. 2, no. 1, pp. 1–32, 2015, doi:10.1186/s40537-015-0030-3.
[43] L. F. Wei, et al., "Security and privacy for storage and computation in cloud computing," *Inf. Sci.*, vol. 258, pp. 371–386, 2014, doi:10.1016/j.ins.2013.04.028.
[44] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Inf. Sci.*, vol. 305, pp. 357–383, 2015, doi:10.1016/j.ins.2015.01.025.
[45] M. Ali, et al., "SeDaSC: Secure data sharing in clouds," *IEEE Syst. J.*, vol. PP, no. 99, pp. 1–10, 2015, doi: 10.1109/JSYST.2014.2379646.
[46] Z. Yan, W. X. Ding, V. Niemi, and A. V. Vasilakos, "Two schemes of privacy-preserving trust evaluation," *Future Generation Comput. Syst.*, vol. 62, pp. 175–189, Dec. 2015, doi:10.1016/j.future.2015.11.006.
[47] M. O'Neill and M. J. B. Robshaw, "Low-cost digital signature architecture suitable for radio frequency identification tags," *IET Comput. Digital Techn.*, vol. 4, no. 1, pp. 14–26, 2010, doi: 10.1049/iet-cdt.2008.0165.

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, respectively. She is currently a professor with the Xidian University, Xi'an, China and a visiting professor with the Aalto University, Espoo, Finland. She authored more than 140 publications and solely authored two books. She is the inventor and co-inventor of 47 patents and patent applications. Her research interests include trust, security and privacy, social networking, cloud computing, networking systems, and data mining. She is currently an associate edtor of the *IEEE Internet of Things Journal, IEEE Access Journal, Security and Communication Networks* (John Wiley), and the *KIIS Transactions on Internet and Information Systems*. She serves as a leading guest editor for more than 10 reputable journals, including *Information Sciences*, *ACM Multimedia Computing, Communications, and Applications*, *IEEE Systems Journal*, *Information Fusion*, *Future Generation Computer Systems*, etc. She is an organization and program committee member for more than 70 international conferences and workshops. She is a senior member of the IEEE.

Wenxiu Ding received the BEng degree in information security from Xidian University, Xi'an, China, in 2012. She is currently working toward the PhD degree in information security from the School of Telecommunications Engineering, Xidian University, and also a visiting research student in the School of Information Systems, Singapore Management University. Her research interests include privacy preservation, data mining and trust management.



Xixun Yu received the BEng degree in telecommunications engineering from Xidian University, Xi'an, China, in 2015. He is currently working toward the PhD degree in information security from the School of Cyber Engineering, Xidian University. His research interests include cloud security and verifiable computing.

Haiqi Zhu received the BEng degree in telecommunication from Xidian University, Xi'an, China, in 2015. He is currently working in Online Management in the Baidu.Inc, Beijing China. His research interests include big data, and data mining.

Robert H. Deng has been a professor in the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network and system security. He was an associate editor of the *IEEE Transactions on Information Forensics and Security* from 2009 to 2012. He is currently an associate editor of the *IEEE Transactions on Dependable and Secure Computing*, an associate editor of *Security and Communication Networks* (John Wiley). He is the cochair of the Steering Committee of the ACM Symposium on Information, Computer and Communications Security. He received the University Outstanding Researcher Award from the National University of Singapore in 1999 and the Lee Kuan Yew fellow for Research Excellence from the Singapore Management University in 2006. He was named Community Service Star and Showcased Senior Information Security Professional by (ISC)2 under its Asia-Pacific Information Security Leadership Achievements program in 2010. He is a fellow member of the IEEE.