Avoidance of Duplication of Encrypted Bigdata on Cloud Storage

Sharmi V
Student, Department of CSE, University College of Engineering, Nagercoil.

Asmitha C
Student, Department of CSE, University College of Engineering, Nagercoil.

Mactalin Godfri M
Student, Department of CSE, University College of Engineering, Nagercoil.

Neela K.L
Assistant Professor, Department of CSE, University College of Engineering, Nagercoil.

Abstract – In this current digital world, data is more importance for individuals as well as organizations. In Cloud computing, dominant part is datacenter where clients/users data are stored. In the datacenters all the data might be uploaded multiple time or data can be hacked so while using the cloud services the data need to be encrypted and stored. When it comes to big data in cloud, the big data means huge datasets so it is very essential to manage the data in secure place. Deduplication is one such storage optimization technique that avoids storing duplicate copies of data. Therefore, few of them can be easily deployed in practice. This scheme, propose an efficient method to deduplicate encrypted data stored in cloud based on possession undertaking and Proxy Re-Encryption (PRE). This scheme also uses the standard AES algorithm for data encryption to outsource the data. This scheme motivates to save cloud storage and preserve the privacy of data holders by proposing a method to manage encrypted data storage with deduplication. In PRE scheme, flexibly support data sharing with deduplication and does not intrude the privacy of data holders and integrates cloud records deduplication with access manages. At last this process will be examined by performance based on tremendous evaluation and computer simulations. The outcomes display the superior performance and effectiveness of the scheme for ability sensible deployment, explicitly for big data deduplication in cloud storage.

Index Terms – 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 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. Deduplication becomes critical for big data storage and processing in the cloud.

Deduplication has proved to achieve high cost 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. The practical issue is how to manage encrypted data storage with deduplication in an efficient way. However, current industrial deduplication solutions cannot handle encrypted data. Existing solutions for deduplication suffer from brute-force attacks [7], [2], They cannot flexibly support data access control and revocation at the same time. Most existing solutions cannot ensure reliability, security and privacy with sound performance.

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 unavailable 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.

This scheme is based on Proxy Re-Encryption (PRE) to manage encrypted data storage with deduplication. The aim is to solve the issue of deduplication in the situation where the data holder is not available or difficult to get involved. Mean-while, the performance of data deduplication in the scheme is not influenced by the size of data, thus applicable for big data. Specifically, the contributions of this paper can be summarized as below:

- This scheme motivates to save cloud storage and preserve the privacy of data holders by proposing a method to manage encrypted data storage with deduplication.
- This scheme proposes an effective approach to verify data ownership and check duplicate storage with secure challenge and big data support
- This scheme integrates cloud data deduplication with data access control in a simple way, thus reconciling data deduplication and encryption.
- This scheme proves 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 Drop box, Google Drive, Mozy 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.

Building a deduplication storage system over cloud computing. Sun et al. proposed data deduplication technique, with more reliability. In data de-duplication process are removing unnecessary copies of data and save memory space. Previously, many de-duplication systems are implemented based on the policies such as, file level, block level deduplication and client-server side de-duplication and it has some drawback they are High reliability provision mechanism. Wang et al. proposed a scheme called RADMAD. It is dynamic and distributed recovery process in the cloud storage. The conflict between deduplication and encryption was first discovered by distributed file system.

Policy-based deduplication in secure cloud storage Liu et al. The user data must be transferred twice, which makes low system efficiency in today's limited bandwidth uses equality predicate encryption scheme and a hybrid approach for de-duplication to prevent information leakage. when encrypting the outsourced data. proposed an approach for secure authorized de-duplication which supports the duplicate check with differential privileges of users, but needs more user involvement. It has some disadvantages like, based on the existing de-duplication technology, it proposes the security proxy and random storage strategy, which separate the security service and storage service. In this way, it resolves the convict between data encryption and de-duplication, resist the attack from outside, and prevent the illegal use of user data and privacy from CSP.

Puzio et al. proposed secure deduplication with encrypted data for cloud storage. The advantages of deduplication unfortunately come with a high cost in terms of new security and privacy challenges. It proposes Clouded up, a secure and efficient storage service which assures block-level deduplication and data confidentiality at the same time. This scheme has some drawbacks like, it copes with the inherent security exposures of convergent encryption and propose ClouDedup, which preserves the combined advantages of deduplication and convergent encryption. The security of ClouDedup relies on its new architecture whereby in addition to the basic storage provider, a metadata manager and an additional server are defined: the server adds an additional encryption layer to prevent well-known attacks against convergent encryption.

Reconciling deduplication and client-side encryption is an active research topic. Message-Locked Encryption (MLE) intends to solve this problem [6]. The most prominent manifestation of MLE is Convergent Encryption (CE). Letting M be a file's data, a client first computes a key $K \leftarrow H(M)$ by applying a crypto-graphic hash function H to M, and then computes ciphertext $C \leftarrow E(K,M)$ via a deterministic symmetric encryption scheme.. However, CE is subject to an inherent security limitation, namely, susceptibility to offline brute-force dictionary attacks [3]. 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.

Bellare et al. proposed message-locked encryption and secure deduplication This involves generating the key, followed by encryption and tag generation. It provides definitions of privacy and integrity peculiar to this domain. Now having

created a clear, strong target for designs, it makes contributions that may broadly be divided into two parts: practical and theoretical. This category it analyzes existing schemes and new variants, breaking some and justifying others with proofs in the random-oracle-model (ROM)

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. An image deduplication scheme adopts two servers to achieve verifiability of deduplication [5]. The CE-based scheme described in [3] 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 [6]. 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. Close-up [3] also aims to cope with the inherent security exposures of CE, but it can-not solve the issues 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. [2] proposed DupLESS that provides secure deduplicated storage to resist brute-force attacks. In Dup-LESS, 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, Dup-LESS cannot control data access of other data users in a flexible way. Alternatively, a policy-based deduplication proxy scheme [2] was proposed but it did not consider duplicated data management (e.g., deletion and owner management) and did not evaluate scheme performance.

Previous scheme, 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. This scheme proposes 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

Li et al. [4] 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 over the original file first and then use Merkle tree on the pre-processed data to generate the verification information. When challenging approver, 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.

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 [4]. 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 bur-den 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 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 [8]. To manage and optimize storage nodes based on a client-side transmission status by the proposed I5NS, all nodes must elicit optimal performance and offer suitable resources to clients. 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 [9]. 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.

This scheme apply PRE 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. It aims to achieve deduplication on encrypted big data in cloud

3. PROBLEM FORMULATION

3.1 System and Security Model

This scheme 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; . . . ; 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) a Proxy Server (PS) that does not collude with CSP and is fully trusted by the data holders to verify data ownership and handle data deduplication. This case, PS cannot know the data stored in CSP and CSP should not know the plain user data in its storage upload the data to the cloud.

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 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.

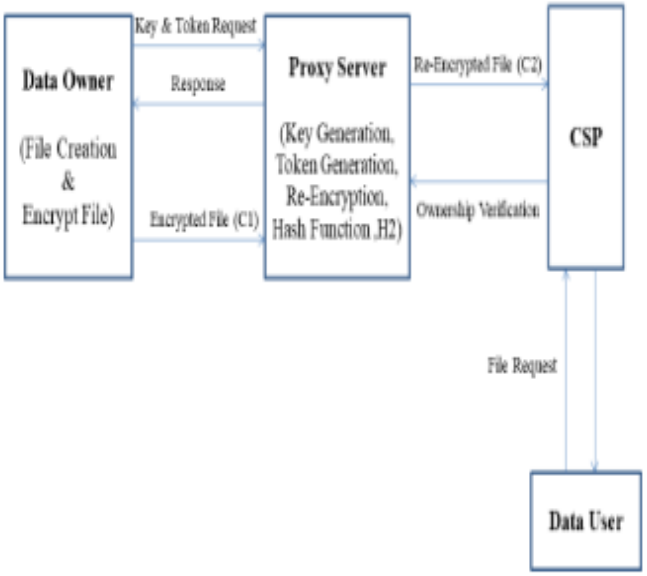


Fig. 1 System Model

System Notation

Key	Description
$(pk_A; sk_A)$	The public key and secret key of user u_A for PRE
DEK	The symmetric key of u_i
H	The hash function
C_A	The cipher text
C_B	The cipher text
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; C_A)	The symmetric decryption function on CT with DEK

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 PS communicate through a secure channel (e.g., SSL) with each other. CSP can authenticate its users in the process of cloud data storage. It further assume that the user policy Policy for data storage, sharing and deduplication is provided to CSP during user registration.

3.2 Preliminary and Notation

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 out-puts 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 $rk_{A->B}$ for a proxy.

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

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.

3.2.1 System Setup

There are two groups G_1; G_T of prime order q with a bilinear map $e : G_1 \times G_1 \rightarrow G_T$. The system parameters are random generators g 2 G_1 and Z =(g; g) 2 G_T.

During system setup, every data holder u_i generates sk_i and pk_i for PRE: $sk_i = a_i$; $pk_i = g^{ai}$ where a_i $2Z_p$. The public key pk_i is used for generating the reencryption key at AP for u_i. Assuming that E(a; b) is an elliptic curve over GF (q), P is a base point that is shared among system entities, $s_{i\in R}\{0; . . . ; 21\}$ is the ECC secret key of user u_i and $V_i = s_iP$ is the corresponding public key and s is a security parameter. This binding is crucial for the verification of the user identity. PS independently generates pk_{PS} and sk_{PS} for PRE and broadcast pk_{PS} to CSP users.

4. SCHEMES

This strategy 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_{PS} 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 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 man-age to save the data encrypted by the real data owner at the cloud with the owner generated DEK and later on, AP sup-ports 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 Procedures

4.1.1 Data Deduplication

The procedure of data deduplication at CSP with the support of PS based on the proposed scheme. It 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.

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 encrypted DEK_1 with pk_{PS} 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 hap-pens because x_2 exists, so CSP forwards $\{x_2; pk_2; Cert(pk_2)\}$ to PS.

Step 5 – Deduplication: CSP re-encrypts $E(pk_{PS} ; DEK_1)$ by calling $R(rk_{PS \rightarrow u2} ; E(pk_{PS} ; DEK_1))= E(pk_2; DEK_1)$ 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 con-firms 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 $D(sk_2; E(pk_2; DEK_1))$

4.2 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. 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 $rk_{PS \rightarrow ui}$ (e.g., $rk_{PS \rightarrow u2}$). AP issues $rk_{PS \rightarrow ui}$ 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.

4.2.1 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. 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 PS for deduplication for other data holders if their re-encryption keys are not known. PS checks its policy for generating and sending corresponding re-encryption keys (e.g., $rk_{ps \rightarrow u2}$), 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. 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.

4.2.2 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 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.

5. SECURITY ANALYSIS AND PERFOMANCE EVALUATION

5.1 Security Analysis

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

Proposition 1. To pass the ownership verification of PS, 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 $H((yP) + cV_i) = H(H(M) \times P)(s_i \times c) \times P$ $(c \times s_i \times P) = H(H(M) \times P) = x_i$ is equal to x_i, i.e., passing the ownership verification of PS.

5.2 Computation Complexity

The proposed scheme involves four kinds of system roles: data owner, User, CSP and PS. To present the computation complexity in details, it adopt AES for symmetric encryption. It analyze the complexity of uploading one data file as below.

Data owner regarded as the first data up loader, it is in charge of four operations: system setup, data encryption, key encryption, and token H(H(M)) .In 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 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 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. 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 $E(pk_{PS}; y)$ 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 $E(pk_{PS}; y)$ 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 O(1).

PS 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. PS needs to issue keys for all authorized data holders that upload the same data. Thus, the computational complexity of PS is O(n). Notably, if the re-encryption key of

a data holder has been generated and issued already, PS will only authorize CSP to perform re-encrytion on CK and will not re-generate the re-encryption key any more.

It 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. It also compare the scheme presented in this paper with our previous work. It can see that our scheme is more efficient at the side of data owners and holders than [3] 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 PS. However, PS is a powerful server full of computation capability. Thus additional computation load at PS is acceptable.

5.3 Communication Cost

The extra communication cost introduced by the proposed scheme is trivial. extra cost introduced by our scheme is $\{x_i; pk_i\}$ during data uploading and storage. Its size is 1344 bits if using SHA-1 hash function. Data deduplication also introduces some extra communication cost: $2\{fx_i; pk_i\}$, c, $E(pk_{PS}$; y), $rk_{PS\rightarrow ui}$, $E(pk_i; DEK)$ and V_i. The size is 5984 bits if DEK size is 256 bits and challenge number c is 160 bits. It 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

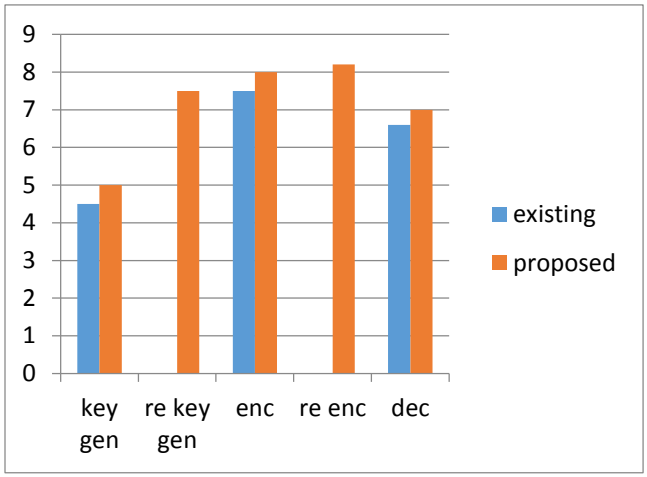
Factors	Existing	Proposed
Key gen	4.5	5
Re-key gen	0	7.5
Enc	7.5	8
Re-enc	0	8.2
Dec	6.6	7

Table 1. Performance evaluation of Proposed System

The table 1 tabulates the percentage value of factors like key generation, Re-Encryption key generation, Encryption, Re-Encryption, Decryption for both existing and proposed model.

The Performance Evaluation is Carried out for both Existing and proposed system model with ASP.NET as a front end and sql as a backend as the software requirements and mouse, keyboard, monitor as the hardware requirements.

Fig.2 Graphical representation of Proposed System



The Fig.2 represents the tabulated values from the table 1. This graph shows that the proposed system has the higher security than the existing system. The factors represent the X axis and the Percentage represents the Y axis.

5.4.1 Implementation and Testing Environment

It implemented the proposed scheme and tested its performance. It applied a MySQL database to store data files and related information. In our test, did not take into account the time of data uploading and downloading. It 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

This experiment, it 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. 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

It tested the efficiency of each operation of 1024-bit PRE with different sizes of AES symmetric keys (128 bits, 196 bits and 256 bits). 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 1millisecond, which implies that our scheme does not introduce heavy processing load to data owners and holders. It also observes 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.

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 PS 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 $DEK^{'}$.

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 PS 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 cipher text 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.

6. CONCLUSION

Managing encrypted data with deduplication is important and significant in practice for achieving a successful cloud storage service, especially for big data storage. In this paper, it proposed a practical scheme to manage the encrypted big data in cloud with deduplication based on ownership challenge and PRE. 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.

REFERENCES

[1] Ali .M, 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

[2] Bellare .M, S. Keelveedhi, and T. Ristenpart, “DupLESS: Server aided encryption for deduplicated storage,” in Proc. 22nd USENIX Conf. Secure., 2013, pp. 179–194.

[3] Bellare .M, 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.

[4] Li .J, Li .Y .K, Chen .X .F, Lee .P .P .C, and Lou .W .J, “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.

[5] Liu .C .Y, Liu .X .J, and Wan .L, “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.

[6] Meye .P, Raipin .P, Tronel .F, and Anceaume .E, “A secure twophase data deduplication scheme,” in Proc. HPCC/CSS/ICESS,2014, pp. 802–809, doi:10.1109/HPCC.2014.134.

[7] Puzio .P, Molva .R, Onen .O .M, and Loureiro .S, “ClouDedup:Secure deduplication with encrypted data for cloud storage,” in Proc. IEEE Int. Cof. Cloud Comput. Technol. Sci., pp370,doi:10.1109/CloudCom.2013.54

[8] Sun .Z, Shen .J, and Yong .J .M, “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.

[9] Wu .T .Y, Pan .J .S, and Lin .C .F, “Improving accessing efficiency of cloud storage using de-duplication and feedback schemes,” IEEE Syst. J., vol. , no. 1, pp. 208–218, Mar. 2014, doi:10.1109/JSYST.2013.2256715.

[10] Yang .C, Ren .J, and Ma .J .F, “Provable ownership of file in deduplication cloud storage,” Proc. IEEE Global Commun. Conf.,2013, pp. 695–700, doi:10.1109/GLOCOM.2013.6831153.