Implementation of Secure Authorized Redundancy Elimination by using Hybrid Cloud Approach

M. Yamuna
M.Tech (CSE)
Annamacharya Institute of Technology (AITK)
Kadapa

***Abstract*— In this paper, it present a scheme that permits a more fine-grained trade-off. The intuition is that outsourced data may require different levels of protection, depending on how popular it is: content shared by many users, such as a popular song or video, arguably requires less protection than a personal document, the copy of a payslip or the draft of an unsubmitted scientific paper. As more corporate and private users outsource their data to cloud storage providers, recent data breach incidents make end-to-end encryption an increasingly prominent requirement. Unfortunately, semantically secure encryption schemes render various cost-effective storage optimization techniques, such as data deduplication, ineffective. it present a novel idea that differentiates data according to their popularity. Based on this idea, designs an encryption scheme that guarantees semantic security for unpopular data and provides weaker security and better storage and bandwidth benefits for popular data. This way, data deduplication can be effective for popular data, whilst semantically secure encryption protects unpopular content. It show that our scheme is secure under the Symmetric External Decisional Diffe-Hellman Assumption in the random oracle model.**

KeyTerms — Deduplication, authorized duplicate check, confidentiality, hybrid cloud

I. INTRODUCTION

Cloud computing provides seemingly unlimited "virtualized "resources to users as services across the whole Internet, while hiding platform an implementation details. Today's cloud service providers offer both highly available storage and massively parallel computing resources at relatively low costs. As cloud computing becomes prevalent, an increasing amount of data is being stored in the cloud and shared by users with specified privileges, which define the access rights of the stored data. One critical challenge of cloud storage services is the management of the ever-increasing volume of data. To make data management scalable in cloud computing, deduplication has been a well-known technique and has attracted more and more attention recently. Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data in storage. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. Instead of keeping multiple data copies with the same content, deduplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy. Deduplication can take place at either the file level or the block level. For file level deduplication, it eliminates duplicate copies of the same file. Deduplication can also take place at the block level, which eliminates duplicate blocks of data that occur in non-identical files. Although data deduplication brings a lot of benefits, security and privacy concerns arise as users' sensitive data are susceptible to both insider and outsider attacks. Traditional encryption, while providing data confidentiality, is incompatible with data deduplication. Specifically, traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different cipher texts, making deduplication impossible.

Convergent encryption has been proposed to enforce data confidentiality while making deduplication feasible. It encrypts/ decrypts a data copy with a convergent key, which is obtained by computing the cryptographic hash value of the content of the data copy. After key generation and data encryption, users retain the keys and send the ciphertext to the cloud. Since the encryption operation is deterministic and is derived from the data content, identical data copies will generate the same convergent key and hence the same cipher text. To prevent unauthorized access, a secure proof of ownership protocol is also needed to provide the proof that the user indeed owns the same file when a duplicate is found. After the proof, subsequent users with the same file will be provided a pointer from the server without needing to upload the same file. A user can download the encrypted file with the pointer from the server, which can only be decrypted by the corresponding data owners with their convergent keys. Thus, convergent encryption allows the cloud to perform deduplication on the ciphertexts and the proof of ownership prevents the unauthorized user to access the file. However, previous deduplication systems cannot support *differential Authorization duplicate check*, which is important in many

applications. In such an authorized deduplication system, each user is issued a set of privileges during system initialization. Each file uploaded to the cloud is also bounded by a set of privileges to specify which kind of users is allowed to perform the duplicate check and access the files. Before

submitting his duplicate check request for some file, the user needs to take this file and his own privileges as inputs. The user is able to find a duplicate for this file if and only if there is a copy of this file and a matched privilege stored in cloud. For example, in a company, many different privileges will be assigned to employees.

In order to save cost and efficiently management, the data will be moved to the storage serverprovider (SCSP) in the public cloud with specified privileges and the deduplication technique will be applied to store only one copy of the same file. Because of privacy consideration, some files will be encrypted and allowed the duplicate check by employees with specified privileges to realize the access control. Traditional deduplication systems based on convergent encryption, although providing confidentiality to some extent, do not support the duplicate check with differential privileges. In other words, no differential privileges have been considered in the deduplication based on convergent encryption technique. It seems to be contradicted if we want to realize both deduplication and differential authorization duplicate check at the same time.

CONTRIBUTIONS

In this paper, aiming at efficiently solving the problem of deduplication with differential privileges in cloud computing, we consider a hybrid cloud architecture consisting of a public cloud and a private cloud. Unlike existing data deduplication systems, the private cloud is involved as a proxy to allow data owner/users to securely perform duplicate check with differential privileges. Such an architecture is practical and has attracted much attention from researchers. The data owners only outsource their data storage by utilizing public cloud while the data operation is managed in private cloud. A new deduplication system supporting differential duplicate check is proposed under this hybrid cloud architecture where the S-CSP resides in the public cloud. The user is only allowed to perform the duplicate check for files marked with the corresponding privileges. Furthermore, we enhance our system in security. Specifically, we present an advanced scheme to support stronger security by encrypting the file with differential privilege keys. In this way, the users without corresponding privileges cannot perform the duplicate check. Furthermore, such unauthorized users cannot decrypt the ciphertext even collude with the S-CSP. Security analysis demonstrates that our system is secure in terms of the definitions specified in the proposed security model. Finally, we implement a prototype of the proposed authorized duplicate check and conduct testbed experiments to evaluate the overhead of the prototype. We show that the overhead is minimal compared to the normal convergent encryption and file upload operations.

II. PRELIMINARIES

In this section, we first define the notations used in thispaper, review some secure primitives used in our secure deduplication. The notations used in this paper are listed in TABLE 1.

Acronym	Description
S-CSP	Storage-cloud service provider
PoW	Proof of Ownership
(pk_U, sk_U)	User's public and secret key pair
k_F	Convergent encryption key for file F
P_U	Privilege set of a user U
P_F	Specified privilege set of a file F
$\phi'_{F,p}$	Token of file F with privilege p

TABLE 1
Notations Used in This Paper

Symmetric encryption. Symmetric encryption uses a common secret key κ to encrypt and decrypt information. A symmetric encryption scheme consists of three primitive functions: • KeyGenSE(1_) *!* κ is the key generation algorithm that generates κ using security parameter 1_;

EncSE(κ,*M*) *!* *C* is the symmetric encryption algorithm that takes the secret κ and message *M* and then outputs the ciphertext *C*; and

• DecSE(κ,*C*) *!* *M* is the symmetric decryption algorithm that takes the secret κ and ciphertext *C* and then outputs the original message *M*.

Convergent encryption.

Convergent encryption provides data confidentiality in deduplication. A user (or data owner) derives a convergent key from each original data copy and encrypts the data copy property holds, i.e., if two data copies are the same, then their tags are the same. To detect duplicates, the user first sends the tag to the server side to check if the identical copy has been already stored. Note that both the convergent key and the tag are independently derived, and the tag cannot be used to deduce the convergent key and compromise data confidentiality. Both the encrypted data copy and its corresponding tag will be stored on the server side. Formally, a convergent encryption scheme can be defined with four primitive functions:with the convergent key. In addition, the user also derives a *tag* for the data copy, such that the tag will be used to detect duplicates. Here, we assume that the tag correctness property holds, i.e., if two data copies are the same, then their tags are the same. To detect duplicates, the user first sends the tag to the server side to check if the identical copy has been already stored. Note that both the convergent key and the tag are independently derived, and the tag cannot be used to deduce the convergent key and compromise data confidentiality. Both the encrypted data copy and its corresponding tag will be stored on the server side. Formally, a convergent encryption scheme can be defined with four primitive functions:

- KeyGenCE(*M*) *!* *K* is the key generation algorithm that maps a data copy *M* to a convergent key *K*;

• EncCE(*K*,*M*) *!* *C* is the symmetric encryption algorithm that takes both the convergent key *K* and the data copy *M* as inputs and then outputs a cipher text *C*;

• DecCE(K,C) *!* M is the decryption algorithm that takes both the ciphertext C and the convergent key K as inputs and then outputs the original data copy M; and

• TagGen(M) *!* T(M) is the tag generation algorithm that maps the original data copy M and outputs a tag T(M)

Proof of ownership. The notion of proof of ownership (PoW) enables users to prove their ownership of data copies to the storage server. Specifically, PoW is implemented as an interactive algorithm (denoted by PoW) run by a prover (i.e., user) and a verifier (i.e., storage server). The verifier derives a short value $\phi(M)$ from a data copy M. To prove the ownership of the data copy M, the prover needs to send ϕ' to the verifier such that $\phi' = \phi(M)$. The formal security definition for PoW roughly follows the threat model in a content distribution network, where an attacker does not know the entire file, but has accomplices who have the file. The accomplices follow the "bounded retrieval model", such that they can help the attacker obtain the file, subject to the constraint that they must send fewer bits than the initial min-entropy of the file to the attacker.

Identification Protocol. An identification protocol _ can be described with two phases: Proof and Verify. In the stage of Proof, a prover/user U can demonstrate his identity to a verifier by performing some identification proof related to his identity. The input of the prover/user is his private key *skU* that is sensitive information such as private key of a public key in his certificate or credit card number etc. that he would not like to share with the other users. The verifier performs the verification with input of public information *pkU* related to *skU*. At the conclusion of the protocol, the verifier outputs either accept or reject to denote whether the proof is passed or not. There are many efficient identification protocols in literature, including certificate-based, identity-based identification etc.

A. SYSTEM MODEL

Hybrid Architecture for Secure Deduplication

At a high level, our setting of interest is an enterprise network, consisting of a group of affiliated clients (for example, employees of a company) who will use the S-CSP and store data with deduplication technique. In this setting, deduplication can be frequently used in these settings for data backup and disaster recovery applications while greatly reducing storage space. Such systems are widespread and are often more suitable to user file backup and synchronization applications than richer storage abstractions. There are three entities defined in our system, that is, *users*, *private cloud* and S-CSP in *public cloud* as shown in Fig. 1.

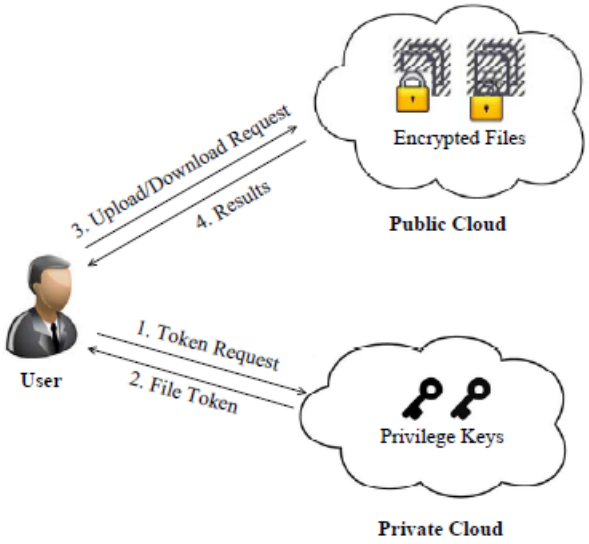

Fig. 1. Architecture for Authorized Deduplication

The S CSP performs deduplication by checking if the contents of two files are the same and stores only one of them. The access right to a file is defined based on a set of *privileges*. The exact definition of a privilege varies across applications. For example, we may define a *rolebased* privilege according to job positions (e.g., Director, Project Lead, and Engineer), or we may define a *time-based* privilege that specifies a valid time period (e.g., 2014-01-01 to 2014-01-31) within which a file can be accessed. A user, say Alice, may be assigned two privileges "Director" and "access right valid on 2014- 01-01", so that she can access any file whose access role is "Director" and accessible time period covers 2014-01- 01. Each privilege is represented in the form of a short message called *token*. Each file is associated with some *file tokens*, which denote the tag with specified privileges (see the definition of a tag in Section 2). A user computes and sends *duplicate-check tokens* to the public cloud for authorized duplicate check. Users have access to the private cloud server, a semitrusted third party which will aid in performing deduplicable encryption by generating file tokens for the requesting users. We will explain further the role of the private cloud server below. Users are also provisioned. with per-user encryption keys and credentials (e.g., user certificates). In this paper, we will only consider the filelevel deduplication for simplicity. In another word, we refer a data copy to be a whole file and file-level deduplication which eliminates the storage of any redundant files. Actually, block-level deduplication can be easily deduced from file-level deduplication, Specifically, to upload a file, a user first performs
the file-level duplicate check. If the file is a duplicate,
then all its blocks must be duplicates as well; otherwise, the user further performs the block-level duplicate check and identifies the unique blocks to be uploaded. Each data copy (i.e., a file or a block) is associated with a token for the duplicate check.

• *S-CSP*. This is an entity that provides a data storage service in public cloud. The S-CSP provides thedata outsourcing service and stores data on behalf of the users. To reduce the

storage cost, the S-CSP eliminates the storage of redundant data via deduplication and keeps only unique data. In this paper, we assume that S-CSP is always online and has abundant storage capacity and computation power.

• *Data Users.* A user is an entity that wants to outsource data storage to the S-CSP and access the data later. In a storage system supporting deduplication,
the user only uploads unique data but does
not upload any duplicate data to save the upload
bandwidth, which may be owned by the same user or different users. In the authorized deduplication system, each user is issued a set of privileges in the setup of the system. Each file is protected with the convergent encryption key and privilege keys to realize the authorized deduplication with differential privileges.

• *Private Cloud.* Compared with the traditional deduplication architecture in cloud computing, this is a new entity introduced for facilitating user's secure usage of cloud service. Specifically, since the computing resources at data user/owner side are restricted and the public cloud is not fully trusted in practice, private cloud is able to provide data user/owner with an execution environment and infrastructure working as an interface between user and the public cloud. The private keys for the privileges are managed by the private cloud, who answers the file token requests from the users. The interface offered by the private cloud allows user to submit files and queries to be securely stored and computed respectively.

Notice that this is a novel architecture for data deduplication in cloud computing, which consists of a twin clouds (i.e., the public cloud and the private cloud). Actually, this hybrid cloud setting has attracted more and more attention recently. For example, an enterprise might use a public cloud service, such as Amazon S3, for archived data, but continue to maintain in-house storage for operational customer data. Alternatively, the trusted private cloud could be a cluster of virtualized cryptographic co-processors, which are offered as a service by a third party and provide the necessary hardware based security features to implement a remote execution environment trusted by the users.

Adversary Model

Typically, we assume that the public cloud and private cloud are both "honest-but-curious". Specifically they will follow our proposed protocol, but try to find out as much secret information as possible based on their possessions. Users would try to access data either within or out of the scopes of their privileges. In this paper, we suppose that all the files are sensitive and needed to be fully protected against both public cloud and private cloud. Under the assumption, two kinds of adversaries are considered, that is, 1) external adversaries which aim to extract secret information as much as possible from both public cloud and private cloud; 2) internal adversaries who aim to obtain more information on the file from the public cloud and duplicate-check token information from the private cloud outside of their scopes. Such adversaries may include S-CSP, private cloud server and authorized users. The detailed security definitions against these adversaries are discussed below and in Section 5, where attacks launched by external adversaries are viewed as special attacks from internal adversaries.

Design Goals

In this paper, we address the problem of privacypreserving deduplication in cloud computing and propose a new deduplication system supporting for • *Differential Authorization.* Each authorized user is able to get his/her individual token of his file to perform duplicate check based on his privileges. Under this assumption, any user cannot generate
a token for duplicate check out of his privileges or without the aid from the private cloud server.

• *Authorized Duplicate Check.* Authorized user is able to use his/her individual private keys to generate query for certain file and the privileges he/she owned with the help of private cloud, while the public cloud performs duplicate check directly and tells the user if there is any duplicate. The security requirements considered in this paper lie in two folds, including the security of file token and security of data files. For the security of file token, two aspects are defined as unforgeability and indistinguishability of file token. The details are given below.

• *Unforgeability of file token/duplicate-check token.* Unauthorized users without appropriate privileges or file should be prevented from getting or generating the file tokens for duplicate check of any file stored at the S-CSP. The users are not allowed to collude with the public cloud server to break the unforgeabilityof file tokens. In our system, the S-CSP is honest but curious and will honestly perform the duplicate check upon receiving the duplicate request from users. The duplicate check token of users should be issued from the private cloud server in our scheme.

• *Indistinguishability of file token/duplicate-check token.* It requires that any user without querying the private cloud server for some file token, he cannot get any useful information from the token, which includes the file information or the privilege information.

• *Data Confidentiality.* Unauthorized users without appropriate privileges or files, including the S-CSP and the private cloud server, should be prevented from access to the underlying plaintext stored at S-CSP. In another word, the goal of the adversary is to retrieve and recover the files that do not belong to them. In our system, compared to the previous definition of data confidentiality based on convergent encryption, a higher level confidentiality is defined and achieved.

1) SECURE DEDUPLICATION SYSTEMS

Main Idea. To support authorized deduplication, the tag of a file F will be determined by the file F and the privilege. To show the difference with traditional notation of tag, we call it file token instead. To support authorized access, a secret key kp will be bounded with a privilege p to generate a file token. Let $\phi' F;p$ = TagGen(F, kp) denote the token of F that is only allowed to access by user with privilege p. In another word, the token $\phi' F;p$ could only be computed by the users with privilege p. As a result, if a file has been uploaded by a user with a duplicate token $\phi' F;p$, then a duplicate check sent from another user will be successful if and only if he also has the file F and privilege p. Such a token generation function could be easily implemented as $H(F, kp)$, where $H(\_)$ denotes a cryptographic hash function.

A First Attempt

Before introducing our construction of differential deduplication, we present a straightforward attempt with the technique of token generation TagGen(F, kp) above to design such a deduplication system. The main idea of this basic construction is to issue corresponding privilege keys to each user, who will compute the file tokens and perform the duplicate check based on the privilege keys and files. In more details, suppose that there are N users in the system and the privileges in the universe is defined as $P = fp1, . . . , psg$. For each privilege p in P, a private key kp will be selected. For a user U with a set of privileges PU, he will be assigned the set of keys $fkpigpi \in PU$.

File Uploading. Suppose that a data owner U with privilege set PU wants to upload and share a file F with users who have the privilege set $PF = fpjg$. The user computes and sends S-CSP the file token $\phi' F;p$ = TagGen(F, kp) for all p 2 PF .
• If a duplicate is found by the S-CSP, the user proceeds proof of ownership of this file with the S-CSP. If the proof is passed, the user will be assigned a pointer, which allows him to access the file. Otherwise, if no duplicate is found, the user computes the encrypted file CF = EncCE(kF , F) with the convergent key kF = KeyGenCE(F) and uploads (CF , $f\phi' F;p$ g) to the cloud server. The convergent key kF is stored by the user locally.

File Retrieving. Suppose a user wants to download a file F. It first sends a request and the file name to the S-CSP. Upon receiving the request and file name, the S-CSP will check whether the user is eligible to download F. If failed, the S-CSP sends back an abort signal to the user to indicate the download failure. Otherwise, the S-CSP returns the corresponding ciphertext CF . Upon receiving the encrypted data from the S-CSP, the user uses the key kF stored locally to recover the original file F.

Problems. Such a construction of authorized deduplication has several serious security problems, which are listed below.
• First, each user will be issued private keys $fkpi gpi \in PU$ for their corresponding privileges, denoted by PU in our above construction. These private keys $fkpi gpi \in PU$ can be applied by the user to generate file token for duplicate check. However, during file uploading, the user needs to compute file tokens for sharing with other users with privileges PF . To compute these file tokens, the user needs to know the private keys for PF , which means PF could only be chosen from PU. Such a restriction makes the authorized deduplication system unable to be widely used and limited. • Second, the above deduplication system cannot prevent the privilege private key sharing among users. The users will be issued the same private key for the same privilege in the construction. As a result, the users may collude and generate privilege private keys for a new privilege set $P*$ that does not belong to any of the colluded user. For example, a user with privilege set $PU1$ may collude with another user with privilege set $PU2$ to get a privilege set $P* = PU1 [PU2$.

• The construction is inherently subject to bruteforce attacks that can recover files falling into a known set. That is, the deduplication system cannot protect the security of predictable files. One of critical reasons is that the traditional convergent encryptionsystem can only protect the semantic security of unpredictable files

Security of Duplicate-Check Token

It consider several types of privacy we need protect, that is
i) unforgeability of duplicate-check token: There are two types of adversaries, that is, external adversary and internal adversary. As shown below, the external adversary can be viewed as an internal adversary without any privilege. If a user has privilege p, it requires that the adversary cannot forge and output a valid duplicate token with any other privilege p' on any file F, where p does not match p'. Furthermore, it also requires that if the adversary does not make a request of token with its own privilege from private cloud server, it cannot forge and output a valid duplicate token with p on any F that has been queried. The internal adversaries have more attack power than the external adversaries and thus we only need to consider the security against the internal attacker, ii) indistinguishability of duplicatecheck token : this property is also defined in terms of two aspects as the definition of unforgeability. First, if a user has privilege p, given a token ϕ', it requires that the adversary cannot distinguish which privilege or file in the token if p does not match p'. Furthermore, it also require that if the adversary does not make a request of token with its own privilege from private cloud server, it cannot distinguish a valid duplicate token with p on any other F that the adversary has not queried. In the security definition of indistinguishablity, we require that the adversary is not allowed to collude with the public cloud servers. Actually, such an assumption could be removed if the private cloud server maintains the tag list for all the files uploaded. Similar to the analysis of unforgeability, the security against external adversaries is implied in the security against the internal adversaries.

CONCLUSION

In this paper, the notion of authorized data deduplication was proposed to protect the data security by including differential privileges of users in the duplicate check. It also presented several new deduplication constructions supporting authorized duplicate check in hybrid cloud architecture, in which the duplicate-check tokens of files are generated by the private cloud server with private keys. Security analysis demonstrates that our schemes are secure in terms of insider and outsider attacks specified in the proposed security model. As a proof of concept, it implemented a prototype of our proposed authorized duplicate check scheme and conduct testbed experiments on our prototype. It shows that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer.

REFERENCES

[1] OpenSSL Project. http://www.openssl.org/.
[2] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In Proc. of USENIX LISA, 2010.
[3] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Serveraided encryption for deduplicated storage. In USENIX Security Symposium, 2013.
[4] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message locked encryption and secure deduplication. In EUROCRYPT, pages 296– 312, 2013.
[5] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. J. Cryptology, 22(1):1–61, 2009.