An Enhanced Multi-layered Cryptosystem Based Secure and Authorized De-duplicaton Model in Cloud Storage System

Rajshree H. Marshinge[1], Shubhangi D. Sapkal[2], Dr. R. R. Deshmukh[3]

[1]Department of CSE, Research student, Government College of Engineering, Aurangabad, India
[2]Department of MCA, Professor, Government College of Engineering, Aurangabad, India
[3]Department of CSE and IT, Professor, Dr. BAMU, Aurangabad, India

Abstract— *Data de-duplication is one of the essential data compression techniques for eliminating duplicate copies of repeating data, and it has been widely used in cloud storage to reduce the amount of storage space and save bandwidth. To protect the privacy of sensitive data while supporting de-duplication, the salt encryption technique has been proposed to encrypt the data before its outsourcing. To protect the data security in a better way, this paper makes the first attempt to formally address the problem of authorized data de-duplication. Different from traditional de-duplication systems, the derivative privileges of users are further considered in duplicate check besides the data itself. We also present various new de-duplication constructions which supports the authorized duplicate check in hybrid cloud architecture. Security analysis demonstrates that the scheme which we used is secure in terms of the definitions specified in the proposed security model. We enhance our system in security. Specially, we present a forward-looking scheme to support a stronger security by encrypting file with differential privilege keys. We show that our proposed authorized duplicate check scheme incurs minimal overhead compared to normal operations.*

Keywords— De-duplication, , hybrid cloud, authorized duplicate check, confidentiality

NOMENCLATURE

S-CSP	-	Storage- cloud service provider
PoW	-	Proof of Ownership
(pk_U, sk_U)	-	Users public and secrete 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
$\varphi'_{F,p}$	-	Token of file F with privilege P

I. INTRODUCTION

In this paper, we instant a scheme that permits a more fine-grained trade-off. The motive behind this is that outsourced data may require various levels of protection, depending on how popular it is: content shared by many users, such as a famous song or video, arguably requires less protection than a personal document, the copy of a pay-slip or the draft of an un-submitted 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. Regrettably, semantically secure encryption schemes provide various cost-effective storage optimization techniques, such as data de-duplication, ineffective. We present a novel and best idea that differentiates data according to their popularity. Depending upon this idea, we design an encryption scheme which guarantees semantic security for unpopular data and provides weaker and less security and better storage and bandwidth benefits for popular data. Using this method, data de-duplication can be effective for popular data, while semantically secure encryption protects unpopular content.

Although data de-duplication brings a large number of benefits, security and privacy concerns arise as users' sensitive data is susceptible to both attacks i.e. insider as well as outsider attacks. Traditional encryption technique, while providing data confidentiality, is not compatible with data de-duplication. Specifically, traditional encryption needs various users to encrypt their data with their own keys. Thus, identical or similar data copies of various users will lead to different cipher a text which makes de-duplication impossible. Salt encryption has been proposed to enforce data confidentiality while making de-duplication feasible. It encrypts/decrypts a data copy with a salt key, which is obtained by computing the cryptographic hash value of the content of the data copy. After the generation of key and data encryption, users retain the keys and send the cipher text to the cloud. Since the encryption operation is deterministic and it is derived from the data content, identical data copies will generate the same salt key and hence the same cipher text. To prevent or protect from an

unauthorized access, a secure proof of ownership protocol is also needed to provide the proof that the user owns the same file when a duplicate is found. After the proof, subsequent users with the same file will be provided with a pointer from the server without needing to upload the same file. A user will be able to download encrypted file with the pointer from the server, which can only be decrypted by the corresponding data owners with their salt keys. Thus, salt encryption allows the cloud to perform de-duplication on the cipher texts and the proof of ownership prevents the unauthorized user to access

Each file uploaded to the cloud is also bounded by a set of privileges to specify which type of users is allowed to perform the duplicate check and access the files. Before submitting the duplicate check request of some file, user needs to take this file and his own privileges as inputs. The user, thus able to identify a duplicate for this file only if there is a copy of this file and a matched privilege stored in cloud. For example, in an organisation, many different privileges will be assigned to employees. In order to save cost and efficient management, the data will be moved to the storage server provider (S- CSP) in the public cloud with specified privileges and the de-duplication method will be applied to store only one copy of the similar or same file. Because of privacy consideration, few files will be encrypted and allowed the duplicate check by employees with specified privileges to realize the access control. Traditional de-duplication systems that are based on salt encryption, although they provide confidentiality to some extent, does not support the duplicate check with differential privileges. In other words, no differential privileges have been considered in the salt encryption technique of de-duplication. It seems contradictory if we want to realize de-duplications and differential authorization duplicate check at the same time.

In this paper, we enhance our system in security. Specifically, we present a forward-looking scheme to support stronger and higher security by encrypting the file with differential privilege keys. Using this way, the users without corresponding privileges can't perform the duplicate check. Moreover, such un-authorized users cannot decrypt the ciphertext even collude with the S-CSP. Security analysis indicates that our system is highly secure in terms of the definitions specified in the proposed security model.

A. SYSTEM MODEL

At a high level, our setting of interest is an enterprise network, which consists of a group of affiliated clients (example, employees of an organization) who will use the S-CSP and store data with de-duplication technique. De-duplication can be used frequently in these settings for data backup and disaster recovery applications while greatly reducing the storage space. Such systems are broad and are more suitable to user file backup and synchronization applications than richer storage abstractions. Three entities are defined in our system, i.e. users, private cloud and S-CSP in public cloud which is shown in Fig. 1. The S-CSP performs the de-duplication by checking whether the contents of two files are same and stores only one of them.

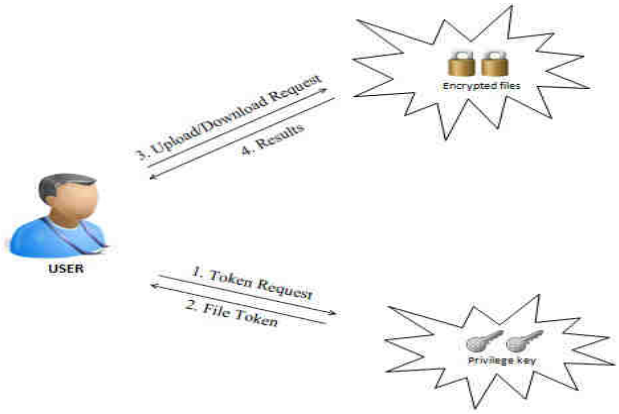

Fig. 1: Architecture of Authorized De-duplication

The access right to a file is defined based on a set of privileges. The true definition of a privilege varies across applications. Each privilege is represented in the form of a short message called token. Every file is associated with some file tokens, which denote the tag with specified privileges. The user finds out and sends duplicate check tokens to the public cloud for authorized duplicate check.

Users has access to the private cloud server, a semi- trusted third party which will aid in performing deduplicable encryption by creating file tokens for the requesting users. We will further explain the role of the private cloud server below. Users are also provisioned with per-user encryption keys and credentials (example. user certificates). In this paper, we will consider the file- level de-duplication only for the simplicity. In another word, we refer a data copy to be a whole file and file-level de-duplication which removes the storage of any redundant files. Actually, block-level de-duplication can be easily figure out from file-level de-duplication. Specifically, to upload a file, a user generally performs the file-level duplicate check first. If the file is a duplicate, then all its blocks must be duplicates as well, else ways, the user further start performing the block-level duplicate check and finally identifies the unique blocks to be uploaded. Each data copy (a block or a file) is associated with a token for the duplicate check.

Notice that this is a novel architecture for data de-duplication in cloud computing, which contain a twin clouds (the public cloud and the private cloud). Actually, this hybrid cloud setting recently has attracted more and more attention. For example, an enterprise might use a public cloud scheme, 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 crypto- graphic co-processors, which are offered as a service by a third party and provides the hardware based security features to implement a remote execution environment trusted by the users.

B. PROPOSED SYSTEM

Cloud computing is the use of computing resources (i.e. hardware and the software) which are delivered as a service over the network (usually the Internet). The name coming from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing give authority to remote services with a user's data, software and computation. Cloud computing consisting of a hardware and a software resources made accessible on the Internet as managed third-party services. These services typically provide access to advanced software applications and high-end networks of server computers.

We propose advanced de-duplication system which supports authorized duplicate check. In this new system of de-duplication, hybrid cloud architecture has been introduced to solve the issue. The private keys for privileges will not be issued to the users directly, which will be kept and managed by the private cloud server instead. In this way, the users cannot share these private keys of privileges in this proposed construction, which means that it can prevent the privilege key sharing among the users in the above straightforward construction. To get a token of the file, the user needs to send a request to the private cloud server. The intuition of this construction can be described as follows. To perform the duplicate check for some file, the user needs to acquire the file token from the private cloud server. The private cloud server will also check the identity of the user before providing the corresponding file token to the specified user. The authorized duplicate check for this file can be carried out by the user with the public cloud before uploading this file. Now on the basis of the results obtained from duplicate check, the user either upload this file or runs PoW. How successful an adversary is at breaking a system is measured by its advantage. An adversary's advantage is the difference between the adversary's probability of breaking the system and the probability of the system can be broken by simply guessing. The advantage is specified as the function of the security parameters.

For everyone all the files are sensitive and it needs to be protected fully against both public cloud and private cloud. Under the assumption, two types of adversaries are considered, i.e. 1) external adversaries which aims to extract secret or private information as much as possible from both public cloud and private cloud. and 2) internal adversaries which aims to obtain more information on the file from the public cloud and duplicate-check token information from the private cloud outside their scopes. Such adversaries may contain S-CSP, private cloud server and authorized users. The detailed and full security definitions against these adversaries is discussed in the below and in Section 5, where the attacks are launched by external adversaries can be viewed as special attacks from internal adversaries.

II. LITERATURE SURVEY

In archival storage systems, there is a enormous amount of duplicate data or redundant data, which occupy prominent extra equipments and power consumptions. The goal of data de-duplication is to minimize the duplicate data in the inter level, has been receiving broad attention both in academic and industry in recent years. With the advent of cloud computing, secure data de-duplication has attracted more attention from research community.

Yuan et al.[1] introduced a de-duplication system in cloud storage to reduce the storage size of tags for integrity check. To increase the security of de-duplication and protect the confidentiality,

Bellare et al.[2] demonstrate how to protect the data confidentiality by transforming the predictable message into unpredictable message. In their system, another third party called key server is proposed to generate the file tag for duplicate check.

Stanek et al. presented a novel encryption scheme that endows the essential security for popular data as well as unpopular data. For popular data that are not particularly sensitive, the traditional conventional encryption is performed. Another two-layered encryption scheme with stronger security while supporting de-duplication is introduced for unpopular data. In this way, they achieved preferable trade between the efficiency and security of the outsourced data. Liet al. addressed the key management issue in block level de-duplication by distributing these keys across multiple servers after encrypting the files.

III. SECURE SYSTEM

A.Symmetric encryption

Symmetric encryption algorithm uses a most common secret key K to encrypt and decrypt the information. A symmetric encryption scheme consists of mainly three primitive functions:

• KeyGenSE (1λ) → K is the key generation algorithm which generates K using security parameter 1λ;

• EncSE (K, M) → C is the symmetric encryption algorithm that takes the secret K and message M and then gives the output ciphertext C; and

• DecSE (K, C) → M is the symmetric decryption algorithm that takes the secret K and cipher-text C and thenprovide the output as the original message M.

B.Salt Encryption Technique

A new salt is arbitrarily generated for each data. In a typical setting, the salt and the data are united and processed along with a cryptographic function, and the resulting i.e. final

output (not the original data) is stored with the salt in a database. Hashing will be allowed for later authentication while protecting the plain text data in the event that the authentication data store is compromised.

To support authorized de-duplication, the tag of a file F will be determined by the file F and the privilege. To display the difference with traditional notation of tag, we call it file token instead. To support the authorized i.e legal access, a secret key kp will be bounded with a privilege p in order to generate a file token. Let $\varphi'_{F,p}$ = TagGen(F, kp) denotes the token of F which is allowed to access by user with privilege p. In other word, the token $\varphi'_{F,p}$ could only be computed by the users with privilege p. As a result, if a file has been uploaded by a user having a duplicate token $\varphi'_{F,p}$ then a duplicate check which has been sent from other user will be successful only if he also having the file F and a privilege p. Such a token generation function can be easily implemented as H(F,kp), where H(·) indicates a cryptographic hash function.

Our system is designed to solve the differential privilege problem in the secure de-duplication process. The security will be examined in terms of two aspects, i.e. first one is the authorization of duplicate check and second one is the confidentiality of data. Some basic tools have been used to construct the secure de-duplication, which are assumed to be highly secure. These basic tools consist of the salt encryption scheme, symmetric encryption scheme, and the PoW scheme. Based upon this assumption, we show you that our systems are secure with respect to the following security analysis.

Before giving our construction of the de-duplication system, we define a binary relation R = {((p, p′)} as follows. We have two privileges let's say them p and p′, we can say that p matches p′ only if R (p, p′) = 1. This type of a generic binary relational definition could be instantiated based on the background of the applications, such as the common hierarchical relation. More accurately, In a hierarchical relation, p matches p′ if and only if p is a higher-level privilege.

A symmetric key kpi for each and every pi □ P will be selected and the set of keys {kpi }pi □P will be sent to the private cloud. An identification protocol Π = (Proof, Verify) is also going to defined, where Proof and Verify are the proof and verification algorithm respectively. Moreover, each user i.e. U has a secret key sk_U to perform the process of identification with servers. Let us assume that user U has the privilege set P_U. It also initializes a PoW protocol POW for the file ownership proof. The private cloud server will maintain a table which can store each and every user's public information pk_U and its corresponding privilege set P_U.

Suppose that a data owner wanted to upload and share a file, let's say, F with users whose privilege belongs to the set P_F = {pj }. The data owner needs to interact with the private cloud before performing the process of duplicate check with the S-CSP. More accurately, the data owner performs an identification process in order to prove its identity with private key sk_U. If it is passed, the private cloud server will find the corresponding privileges PU of the user from its stored table list. The user then computes and immediately sends the file tag φ_F = TagGen(F) to the private cloud server, who will return $\{\varphi'_{F,p\tau}$ = TagGen(φ_F , kpτ)} back to the user for all pτ satisfying R(p,pτ) = 1 and p □ P_U. Then, the user will interact and sends the file token $\{\varphi'_{F,p\tau}\}$ to the S-CSP. That is, the user can finally recover the original file with the convergent key k_F after receiving the encrypted data from the S-CSP.

The encryption key can be viewed as the form of

$k_{F,p}$ = H'(H(F),kp)H2(F)

Where H', H and H2 are all the cryptographic hash functions. The file F is encrypted with another key k, while k will be encrypted with $k_{F,p}$. In this way, the private cloud server and S-CSP cannot decrypt the ciphertext. Furthermore, it is semantically secure to the S-CSP based on the security of symmetric encryption.

IV. EXPERIMENTAL RESULT

There are some experimental results given below. Fig. 2. shows the user dashboard which shows the efficiency report graph of different files uploaded by the user. Here we just took a random set of files and uploaded it and just saw the results for each step i.e. token generation, duplicate check, encryption, transfer.

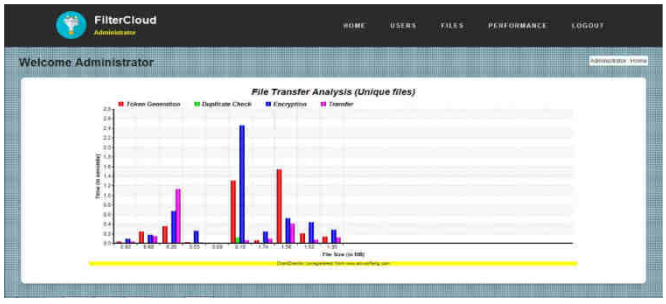

Fig. 2: File Transfer Analysis

Fig. 3: Assigning Rights to the User

Fig. 4. shows the files uploaded by the user and file upload performance analysis. User first need to upload a file with privilege set and then user computes and sends S-CSP file token. If a duplicate is found, it will shows a message "

dada duplication is not allowed! File already exist" this is shown in Fig. 5.

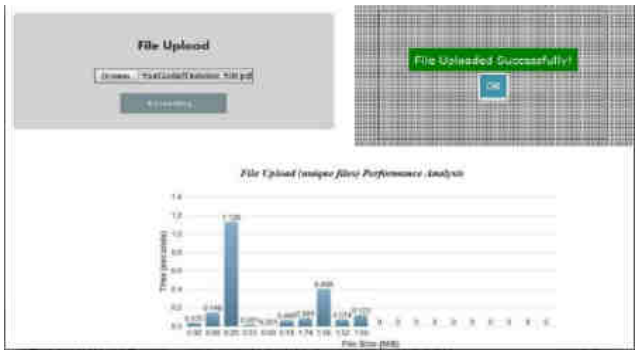

Fig 4: File uploading process and File upload analysis

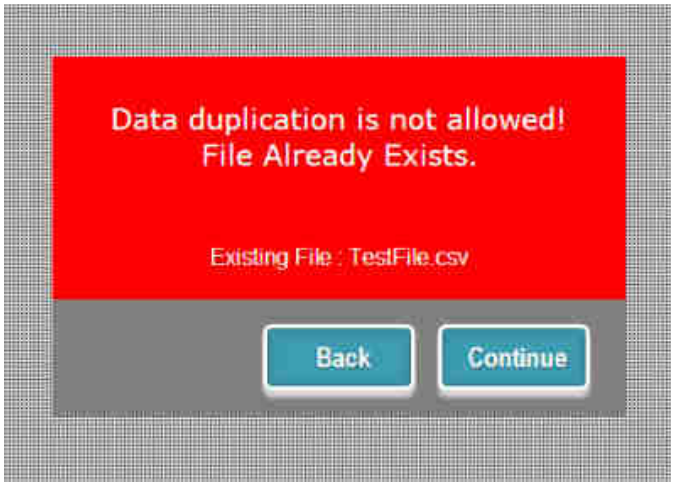

Fig. 5: Data de-duplication Result

And if no duplicate is found then the file will be stored in the cloud server.

V. CONCLUSION

In this paper, the impression of authorized data de-duplication was proposed to protect the data security by including differential privileges of users in the duplicate check. Safety analysis demonstrates that our technique is more secure in terms of both the attacks i.e. insider as well as outsider attacks specified in the proposed security model. We also presented several new de-duplication constructions supporting accredited duplicate check in hybrid cloud architecture, in which the duplicate-check tokens of files are generated by the private cloud server with private keys. As a proof of concept, we enforced a prototype of our proposed authorized duplicate check scheme and conduct testbed experiments on our prototype.

VI. ACKNOWLEDGEMENT

I am thankful to people those who gave me spirit and knowledge about my research and provided me support at all the stages. I am also thankful to the entire CSE department of Government College of Engineering, Aurangabad for giving me all facilities and work environment. I am also thankful to the researchers.

REFERENCES

[1] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with de-duplication," IACR Cryptology ePrint Archive, 2013:149, 2013

[2] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure de-duplication," In Eurocrypt, pp. 296– 312, 2013.

[3] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: Server- aided encryption for deduplicated storage," In Usenix Security Symposium, 2013

[4] P. Anderson and L. Zhang, "Fast and secure laptop backups with encrypted de-duplication," In Proc. of Usenix Lisa 2010.

[5] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of owenership in remote storage system," In Y. Chen, G. Danezis and V. Shmatikov, editors, ACM Conference on Computer and Communications Security, pp. 491-500, 2011

[6] K. Zhang, X. Zhou, Y. Chen, X. Wang and Y. Raum, "Sedic privacy aware data intensive computing on hybrid clouds," In Proceedings of the 18th ACM Conference on Computer and Communications security, CCS'11, New York, NY, USA, pp. 515-526, 2011

[7] Z. Wilcox-O'Hearn and B. Warner, "Tahoe: the least-authority file system," In Proc. of ACM StorageSS, 2008.