ISSN: 2321-7782 (Online)
Volume 4, Issue 3, March 2016
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Authorized Deduplication Implementation with Key Management

Jyothula Vidya[1]
M.Tech Scholar
Chebrole Engineering College
Chebrole(V&M),Guntur(dt)
Andhra Pradesh-522212, India

K. Venkata Ramaiah[2]**, M.Tech**
Associate Professor
Chebrole Engineering College
Chebrole(V&M),Guntur(dt)
Andhra Pradesh-522212, India, India

Abstract: This paper makes the first attempt to formally address the problem of achieving efficient and reliable key management in secure deduplication. We first introduce a baseline approach in which each user holds an independent master key for encrypting the convergent keys and outsourcing them to the cloud. However, such a baseline key management scheme generates an enormous number of keys with the increasing number of users and requires users to dedicatedly protect the master keys. To this end, we propose Dekey, a new construction in which users do not need to manage any keys on their own but instead securely distribute the convergent key shares across multiple servers[1]. Security analysis demonstrates that Dekey is secure in terms of the definitions specified in the proposed security model. As a proof of concept, we implement Dekey using the Ramp secret sharing scheme and demonstrate that Dekey incurs limited over head in realistic environments. This paper represents that, many techniques are using for the elimination of duplicate copies of repeating data, from that techniques, one of the important data compression technique is data duplication. Many advantages with this data duplication, mainly it will reduce the amount of storage space and save the bandwidth when using in cloud storage.

Keywords: de-duplication, hybrid cloud, authorized duplicate check, confidentiality, encryption.

I. Introduction

The advent of cloud storage motivates enterprises and organizations to outsource data storage to third-party cloud providers, as evidenced by many real-life case studies [3]. One critical challenge of today's cloud storage services is the management of the ever-increasing volume of data. According to the analysis report of IDC, the volume of data in the wild is expected to reach 40 trillion gigabytes in 2020 [9]. To make data management scalable, deduplication has been a well-known technique to reduce storage space and upload bandwidth in cloud storage. 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. Unlimited "virtualized" resources to users as services across the whole internet providing by the cloud computing to hide platforms and implementation details. Highly available storage and massively parallel computing resources providing by the cloud services at low costs[2]. Cloud computing widely spread in the world, maximum amount of data stored in the clouds and shred by the users with specified rights, which define as access rights of the stored data. One of the critical challenge of cloud storage services is the management of the duplication is one of the best technique to make the data management in the cloud computing. From a user's perspective, data outsourcing raises security and privacy concerns. We must trust third-party cloud providers to properly enforce confidentiality, integrity checking, and access control mechanisms against any insider and outsider attacks. However, deduplication, while improving storage and bandwidth efficiency, is incompatible with traditional encryption. 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. Although data de-duplication brings a lot of advantages, security and privacy concerns arise as users sensitive data are susceptible to both the insider and outsider attacks. When compares the traditional encryption with data duplication[3]. It will provide data confidentiality. In the traditional encryption requires different users to encrypt data with their own keys. Thus identical copies of different users will lead to different cipher texts, making de-duplication impossible. One of the new technique has been proposed to encrypt data confidentiality while making de-duplication feasible, i,e convergent encryption. This convergent encrypt provides one convergent key to encrypt/decrypt the data, which is obtained by computing the cryptographic hash value of the content of the data copy. After completion of key generation and data encryption, users retain the keys and send the cipher text to the cloud. Since the encryption operation is deterministic and is derived from the data content, identical data copies generate the same convergent key and hence the same cipher text. Convergent encryption [8] provides a viable option to enforce data confidentiality while realizing deduplication. It encrypts/decrypts a data copy with a convergent key, which is derived by computing the cryptographic hash value of the content of the data copy itself [8]. After key generation and data encryption, users retain the keys and send the ciphertext to the cloud. Since encryption is deterministic, identical data copies will generate the same convergent key and the same cipher text[4]. This allows the cloud to perform deduplication on the ciphertexts. The cipher texts can only be decrypted by the corresponding data owners with their convergent keys. However, the baseline approach suffers two critical deployment issues. First, it is inefficient, as it will generate an enormous number of keys with the increasing number of users. Specifically, each user must associate an encrypted convergent key with each block of its outsourced encrypted data copies, so as to later restore the data copies.

II. PRELIMINARIES

In this we go through with the notations used in this paper. Analyze the secure primitives used in our secure duplication.

Symmetric Encryption:

It uses a common secret key k to encrypt and decrypt information. A symmetric encryption scheme consists of three primitive functions:

a. KeyGenSE() $\rightarrow \kappa$ is the key generation algorithm that generates κ using security parameter;

b. EncSE(κ,M) $\rightarrow C$ is the symmetric encryption algorithm that takes the secret κ and message M and then outputs the cipher text C; anD[5]

c. DecSE(κ,C) $\rightarrow M$ is the symmetric decryption algorithm that takes the secret κ and cipher text C and then outputs the original message M.

Convergent Encryption:

With this convergent encryption [4], [8] we get secure confidentiality of de-duplication. Data owner gets convergent key from each original data copy and encrypts data copy with the convergent key. A tag is also provide to the user with the data copy, tag will be used to detect duplicates[6]. If two data copies are same, then their tags are same. To identify and check the duplicates, the user first sends a tag to the server side to check. Server will replies, if the identical copy has been already stored or not. Both (confidentiality check and tag) are independently derived. Tag cannot used to reduce the convergent key and compromise data confidentiality. Tag and it's encrypted data copy will be stored in the server side.

2.1 Proof of Ownership

The notion of proof of ownership (PoW) is to solve the problem of using a small hash value as a proxy for the entire file in client-side deduplication [11], where the adversary could use the storage service as a content distribution network. This proof mechanism in PoW provides a solution to protect the security in client-side deduplication. In this way, a client can prove to the server that it indeed has the file. Dekey supports client-side deduplication with PoW to enable 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 _ðMÞ from a data copy M. To prove the ownership of the data copy M, the prover needs to send _0 and run a proof algorithm with the verifier. It is passed if and only if

_0 ¼ _ðMÞ and the proof is correct. In our paper, we use the notations of PoWF and PoWB to denote PoW for a file F and block B, respectively. Specifically, the notation of PoWF;j will be used to denote a PoW protocol with respect to TjðFÞ ¼ TagGenCEðF; jÞ.

III. Problem Formulation

3.1 System Model

We first formulate a data outsourcing model used byDekey. There are three entities, namely: the user, the storage cloud service provider (S-CSP), and the key management cloud service provider (KM-CSP), as elaborated below. User. A user is an entity that wants to outsource data storage to the S-CSP and access the data later. To save the upload bandwidth, the user only uploads unique data but does not upload any duplicate data, which may be owned by the same user or different users.. S-CSP. The S-CSP provides the data 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. KM-CSP. A KM-CSP maintains convergent keys for users, and provides users with minimal storage and computation services to facilitate key management. For fault tolerance of key management, we consider a quorum of KM-CSPs, each being an independent entity. Each convergent key is distributed across multiple KM-CSPs using RSSS (see Section 2)[7]. In this work, we refer a data copy to be either a whole file or a smaller-size block, and this leads to two types of deduplication: 1) file-level deduplication, which eliminates the storage of any redundant files, and 2) block-level deduplication, which divides a file into smaller fixed-size or variable-size blocks and eliminates the storage of any redundant blocks. Using fixed-size blocks simplifies the computations of block boundaries, while using variable size blocks (e.g., based on Rabin fingerprinting [12]) provides better deduplication efficiency. We deploy our deduplication mechanism in both file and block levels. 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 tag for the duplicate check (see Section).

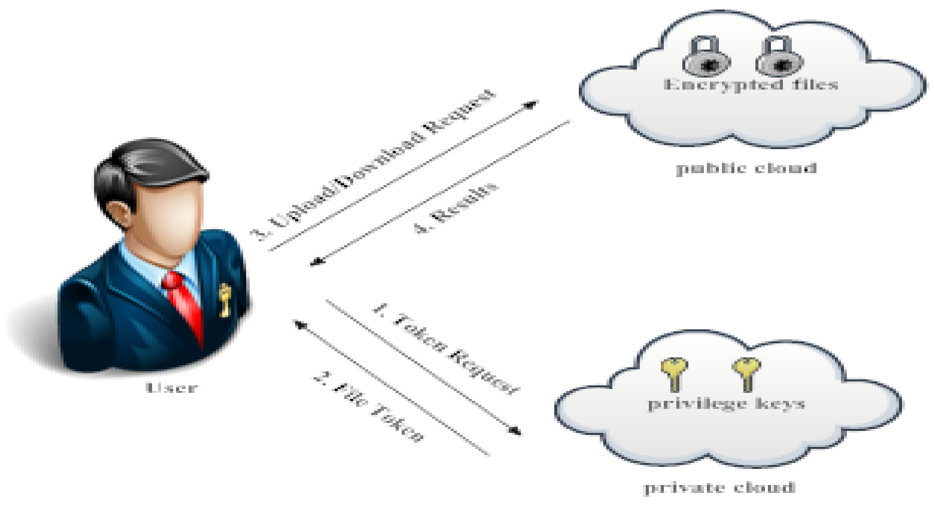

Fig-1: Working of authorized de-duplication

There are three entities define in our system as shown in figure 1, those are,

- Users
- Private cloud
- S-CSP in public cloud

De-duplication performed by S-CSP by checking if the contents of two files are the same and stores only one of them. Based on the set of privileges, the access right of a file is defined. The exact definition of a privilege varies across applications. For example, we may define a *role-based* privilege [9], [16] 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.

 S-CSP. This is an entity that provides a data storage service in public cloud. The S-CSP provides the data 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 de-duplication 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 de-duplication, 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[14]. In the authorized de-duplication 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 de-duplication with differential privileges.

 Private Cloud. Compared with the traditional de-duplication 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.

3.2 Design Description:

The detailed architecture of the design is showed in figure2. We can get the processing details from this architecture. Four different types of modules are present in the architecture. Data Owner Module, Encryption and Decryption Module, Remote User Module, Cloud Server Module[10]. User login details are required to upload or download a file and the details of modules mentioned below.

- DATA OWNER MODULE:

a. Data Owner login validations.

b. Upload Files.

c. Manipulates Encrypted files.

d. Differential Authorization.

- ENCRYPTION AND DECRYPTION MODULE:

a. Generate signs.

b. Encrypts and uploads files.

c. Decrypts and downloads files.

d. Data confidentiality.

- REMOTE USER MODULE:

a. Accessing Files.

b. Remote User login validations.

- CLOUD SERVER MODULE:

a. Authorized Duplicate Check.

b. Accessing files.

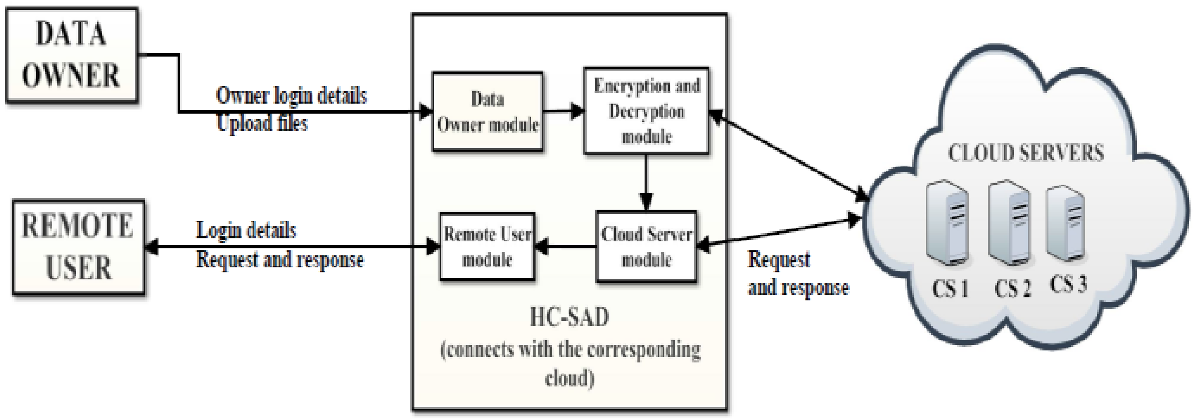

Fig-2. System Architecture design

IV. Algorithms Used

In this section, we use two types of algorithms,

1). For file uploading.

2). For file downloading.

FOR UPLOADING A FILE

```
BEGIN
Step –1 Read file
Step –2 Cloud server checks for duplication
Step –3 Sends duplication response whether the file already exists or not
Step – 4 If the file does not exist
    4.1 Display “file does not exist”
Step – 5 Then it uploads the file
Step – 6 If the file already exist
    6.1 Display “file already exist”
END
```

FOR DOWNLOADING A FILE

```
BEGIN
Step –1 Read file
Step –2 Cloud server checks for duplication
Step –3 Sends duplication response whether the file already exists or not
Step –4 If the file exist -4.1 Display “file exist”
Step –5 then it downloads the file
Step –6 If the file does not exist –
    6.1 Display "file does not exist"
END
```

V. Implementation

We implement a prototype of the proposed authorized De-duplication system, in which we model three entities as separate C++ programs. A *Client* program is used to model the data users to carry out the file upload process[11]. A *Private Server* program is used to model the private cloud which manages the private key and handles the file token computation. A *Storage Server* program is used to model the S-CSP which stores and de-duplicates files. We implement cryptographic operations of hashing and encryption with the OpenSSL library [12].We also implement the communication between the entities based on HTTP, using GNU Libmicrohttpd [10] and libcurl [13]. Thus, users can issue HTTP Post requests to the servers. Our implementation of the **Client** provides the following function calls to support token generation and de-duplication along the file upload process.

- *FileTag(File)* – It computes SHA-1 hash of the File as File Tag;

Our implementation of the **Private Server** includes corresponding request handlers for the token generation and maintains a key storage with Hash Map.

- *TokenGen(Tag, UserID)* **-** It loads the associated privilege keys of the user and generate the token with HMAC-SHA-1 algorithm; and

- *ShareTokenGen(Tag, {Priv.})* **-** It generates the share token with the corresponding privilege keys of the sharing privilege set with HMAC-SHA-1 algorithm.

Our implementation of the **Storage Server** provides de-duplication and data storage with following handlers and maintains a map between existing files and associated token with Hash Map.

- *DupCheck(Token)* **-** It searches the File to Token Map for Duplicate; and

- *FileStore(FileID, File, Token)* **-** It stores the File on Disk and updates the Mapping.

VI. Conclusion

Notion of authorized data de-duplication was proposed to protect the data security by including differential privileges of users in the duplicate check. We also presented several new de-duplication 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, we implemented a prototype of our proposed authorized duplicate check scheme and conduct test-bed experiments on our prototype[16]. We showed that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer. We propose Dekey, an efficient and reliable convergent key management scheme for secure deduplication. Dekey applies deduplication among convergent keys and distributes convergent key shares across multiple key servers, while preserving semantic security of convergent keys and confidentiality of outsourced data. We implement Dekey using the Ramp secret sharing scheme and demonstrate that it incurs small encoding/decoding overhead compared to the network transmission overhead in the regular upload/download operations.

References

1. OpenSSL Project. [Online]. Available: http://www.openssl.org/.
2. NIST's Policy on Hash Functions, Sept. 2012. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/policy.html.
3. AmazonCase Studies. [Online]. Available: https://aws.amazon.com/solutions/case studies/#backup.
4. P. Anderson and L. Zhang, ''Fast and Secure Laptop Backups with Encrypted De Duplication,'' in Proc. USENIX LISA, 2010, pp. 1-8.
5. M. Bellare, S. Keelveedhi, and T. Ristenpart, ''Message-Locked Encryption and Secure Deduplication,'' in Proc. IACR Cryptology ePrint Archive, 2012, pp. 296-3122012:631.
6. G.R. Blakley and C. Meadows, ''Security of Ramp Schemes,'' in Proc. Adv. CRYPTO, vol. 196, Lecture Notes in Computer Science, G.R. Blakley and D. Chaum, Eds., 1985, pp. 242-268.
7. A.T. Clements, I. Ahmad, M. Vilayannur, and J. Li, ''Decentralized Deduplication in San Cluster File Systems,'' in Proc. USENIX ATC, 2009, p. 8.
8. J.R. Douceur, A. Adya, W.J. Bolosky, D. Simon, and M. Theimer, ''Reclaiming Space from Duplicate Files in a Serverless Distributed File System,'' in Proc. ICDCS, 2002, pp. 617-624.
9. J. Gantz and D. Reinsel, The Digital Universe in 2020: Big Data, Bigger Digital Shadows, Biggest Growth in the Far East,Dec.2012.[Online].Available:http://www.emc.com/collateral/analystreports/idc-the-digital-universe-in-2020.pdf.
10. R. Geambasu, T. Kohno, A. Levy, and H.M. Levy, ''Vanish: Increasing Data Privacy with Self-Destructing Data,'' in Proc. USENIX Security Symp., Aug. 2009, pp. 316-299.
11. D. Ferraiolo and R. Kuhn. Role-based access controls. In 15th NIST-NCSC National Computer Security Conf., 1992.
12. GNU Libmicrohttpd. Http://www.gnu.org/software/libmicrohttpd/.
13. S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg. Proofs of ownership in remote storage systems. In Y. Chen, G. Danezis, and V. Shmatikov, editors, ACM Conference on Computer and Communications Security, pages 491–500. ACM, 2011.
14. J. Li, X. Chen, M. Li, J. Li, P. Lee, andW. Lou. Secure deduplication with efficient and reliable convergent key management. In IEEE Transactions on Parallel and Distributed Systems, 2013.
15. libcurl. http://curl.haxx.se/libcurl/
16. C. Ng and P. Lee. Revdedup: A reverse deduplication storage system optimized for reads to latest backups. In Proc. of APSYS, Apr 2013.