ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

A Novel on Secure Distributed Deduplication Systems with Improved Reliability

G.Dileep Kumar, R.Suresh, J.Naga Muneiah

PG Student, Department of CSE, C R Engineering College, Tirupati, India

Associate Professor, Department of CSE,, C R Engineering College, Tirupati, India

HOD, Department of CSE,, C R Engineering College, Tirupati, India

ABSTRACT: Cloud data storage service providers such as Mozy, Dropbox, and others perform deduplication to save space by only storing one copy of each file uploaded. While using cloud data storage clients always concern for confidentiality of data. So clients usually encrypt their data, and put it on to the storage. this paper makes the first attempt to formalize the notion of distributed reliable deduplication system. We propose new distributed deduplication systems with higher reliability in which the data chunks are distributed across multiple cloud servers. The security requirements of data confidentiality and tag consistency are also achieved by introducing a deterministic secret sharing scheme in distributed storage systems, instead of using convergent encryption as in previous deduplication systems. Security analysis demonstrates that our deduplication systems are secure in terms of the definitions specified in the proposed security model. As a proof of concept, we implement the proposed systems and demonstrate that the incurred overhead is very limited in realistic environments.

KEYWORDS: Cloud Storage, Deduplication, distributed storage system, reliability, secret sharing

I. INTRODUCTION

As the increasing growth of the computing technology and network technology brings increasing data storage demands. Several organizations and people outsource their data to remote storage due to limited storage capacity, maintenance overhead and less budgetary. Cloud or Remote data storage satisfies these requirements. Cloud data storage provides a giant shared resource pool and therefore the users move towards it to satisfy their requirements. Deduplication is able to reduce the required storage capacity since only the unique data is stored. [18] Cloud or Remote storage offers services for information storage, information access and different process capabilities in a reliable manner. However security becomes the main concern of each cloud storage service provider and clients who use the cloud resources. Additionally, it reduces the overhead of cloud users in maintaining their crucial information in a much secured manner. However in those cases, the provider makes sure that their infrastructure is secure enough to secure client's information and its applications and the customer on the client side, must be able to attest the provider has taken correct security measures to protect their crucial information. Furthermore, the challenge for data privacy also arises as more and more sensitive data are being outsourced by users to cloud. Encryption mechanisms have usually been utilized to protect the confidentiality before outsourcing data into cloud. Most commercial storage service providers are reluctant to apply encryption over the data because it makes deduplication impossible. The reason is that the traditional encryption mechanisms, including public key encryption and symmetric key encryption, require different users to encrypt their data with their own keys. As a result, identical data copies of different users will lead to different cipher texts. To solve the problems of confidentiality and deduplication, the notion of convergent encryption [4] has been pro-posed and widely adopted to enforce data confidentiality while realizing deduplication. However, these systems achieved confidentiality of outsourced data at the cost of decreased error resilience. Therefore, how to protect both confidentiality and reliability while achieving Deduplication in a cloud storage system is still a challenge.

In this paper, we show how to design secure deduplication systems with higher reliability in cloud computing. We introduce the distributed cloud storage servers into deduplication systems to provide better fault tolerance. To further protect data confidentiality, the secret sharing technique is utilized, which is also compatible with the distributed

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

storage systems. In more details, a file is first split and encoded into fragments by using the technique of secret sharing, instead of encryption mechanisms. These shares will be distributed across multiple independent storage servers. Furthermore, to support deduplication, a short cryptographic hash value of the content will also be computed and sent to each storage server as the fingerprint of the fragment stored at each server. Only the data owner who first uploads the data is required to compute and distribute such secret shares, while all following users who own the same data copy do not need to compute and store these shares any more. To recover data copies, users must access a minimum number of storage servers through authentication and obtain the secret shares to reconstruct the data. In other words, the secret shares of data will only be accessible by the authorized users who own the corresponding data copy. Another distinguishing feature of our proposal is that data integrity, including tag consistency, can be achieved.

This paper is organized as follows. In Section II, we present the problem formulation of deduplication. Our constructions are presented in Section III. The security analysis is given in Section IV. The implementation and evaluation are shown in Sections V; finally, we draw our conclusions in Section VI.

II. PROBLEM FORMULATION

(a)System Model

This section is devoted to the definitions of the system model and security threats. Two kind's entities will be involved in this deduplication system, including the user and the storage cloud service provider (S-CSP). Both client-side deduplication and server-side deduplication are supported in our system to save the bandwidth for data uploading and storage space for data storing.

- *User*. The user is an entity that wants to outsourcedata 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. Furthermore, the fault tolerance is required by users in the system to provide higher reliability.
- *S-CSP*. The S-CSP is an entity that provides theoutsourcing data storage service for the users. In the deduplication system, when users own and store the same content, the S-CSP will only store a single copy of these files and retain only unique data. A deduplication technique, on the other hand, can reduce the storage cost at the server side and save the upload bandwidth at the user side. For fault tolerance and confidentiality of data storage, we consider a quorum of S-CSPs, each being an independent entity. The user data is distributed across multiple S-CSPs.

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. All data copies and tags will be stored in the S-CSP.

(b)Threat Model and Security Goals

Two types of attackers are considered in our threat model: (i) an outside attacker, who may obtain some knowledge of the data copy of interest via public channels. An outside attacker plays the role of a user that interacts with the S-CSP; (ii) an inside attacker, who may have some knowledge of partial data information such as the ciphertext. An insider attacker is assumed to be honest-but-curious and will follow our protocol, which could refer to the S-CSPs in our system. Their goal is to extract useful information from user data. The following security requirements, including confidentiality, integrity, and reliability are considered in our security model.

Confidentiality : Here, we allow collusion among the S-CSPs. However, we require that the number of colluded S-CSPs is not more than a predefined threshold. To this end, we aim to achieve data confidentiality against collusion attacks. We require that the data distributed and stored among the S-CSPs remains secure when they are unpredictable (i.e., have high min-entropy), even if the adversary controls a predefined number of S-CSPs. The goal of the adversary is to retrieve and recover the files that do not belong to them. This requirement has recently been formalized in [6] and called the privacy against chosen distribution attack. This also implies that the data is secure against the adversary who does not own the data.

Integrity: Two kinds of integrity, including tag consistency and message authentication, are involved in the security model. Tag consistency check is run by the cloud storage server during the file uploading phase, which is used to

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

prevent the duplicate/ciphertext replacement attack. If any adversary uploads a maliciously-generated ciphertext such that its tag is the same with another honestly-generated ciphertext, the cloud storage server can detect this dishonest behavior. Thus, the users do not need to worry about that their data are replaced and unable to be decrypted. Message authentication check is run by the users, which is used to detect if the downloaded and decrypted data are complete and uncorrupted or not. This security requirement is introduced to prevent the insider attack from the cloud storage service providers.

Reliability : The security requirement of reliability indeduplication means that the storage system can provide fault tolerance by using the means of redundancy. In more details, in our system, it can be tolerated even if a certain number of nodes fail. The system is required to detect and repair corrupted data and provide correct output for the users.

III. THE DISTRIBUTED DEDUPLICATION SYSTEMS

The distributed deduplication systems' proposed aim is to reliably store data in the cloud while achieving confidentiality and integrity. Its main goal is to enable deduplication and distributed storage of the data across multiple storage servers. Instead of encrypting the data to keep the confidentiality of the data, our new constructions utilize the secret splitting technique to split data into shards. These shards will then be distributed across multiple storage servers.

(a)Building Blocks

Secret Sharing Scheme. There are two algorithms in asecret sharing scheme, which are Share and Recover. The secret is divided and shared by using Share. With enough shares, the secret can be extracted and recovered with the algorithm of Recover. In our implementation, we will use the Ramp secret sharing scheme (RSSS) [7], [8] to secretly split a secret into shards. Specifically, the (n, k, r)-RSSS (where $n > k > r \geq 0$) generates n shares from a secret so that (i) the secret can be recovered from any k or more shares, and (ii) no information about the secret can be deduced from any r or less shares. Two algorithms, Share and Recover, are defined in the (n, k, r)-RSSS.

- Share divides a secret S into $(k-r)$ pieces of equal size, generates r random pieces of the same size, and encodes the k pieces using a non-systematic k-of-n erasure code into n shares of the same size;
- Recover takes any k out of n shares as inputs and then outputs the original secret S.

It is known that when $r = 0$, the $(n, k, 0)$-RSSS becomes the (n, k) Rabin's Information Dispersal Algorithm (IDA) [9]. When $r = k - 1$, the $(n, k, k-1)$-RSSS becomes the (n,k) Shamir's Secret Sharing Scheme (SSSS) [10].

Tag Generation Algorithm. In our constructionsbelow, two kinds of tag generation algorithms are defined, that is, TagGen and TagGen'.TagGen is the tag generation algorithm that maps the original data copy F and outputs a tag $T(F)$. This tag will be generatedby the user and applied to perform the duplicate check with the server. Another tag generation algorithm TagGen' takes as input a file F and an index j and outputs a tag. This tag, generated by users, is used for the proof of ownership for F.

Message authentication code.A message authenticationcode (MAC) is a short piece of information used to authenticate a message and to provide integrity and authenticity assurances on the message. In our construction, the message authentication code is applied to achieve the integrity of the outsourced stored files. It can be easily constructed with a keyed (cryptographic) hash function, which takes input as a secret key and an arbitrary-length file that needs to be authenticated, and outputs a MAC. Only users with the same key generating the MAC can verify the correctness of the MAC value and detect whether the file has been changed or not.

(b)The File-level Distributed Deduplication System

To support efficient duplicate check, tags for each file will be computed and are sent to S-CSPs. To prevent a collusion attack launched by the S-CSPs, the tags stored at different storage servers are computationally independent and different. We now elaborate on the details of the construction as follows.

System setup : In our construction, the number of storage servers S-CSPs is assumed to be n with identities denoted by $id_1, id_2, \cdots, id_n$, respectively. Define the security parameter as 1 and initialize a secret sharing scheme SS = (Share, Recover), and a tag generation algorithm TagGen. The file storage system for the storage server is set to be $\perp$.*File Upload.* To upload a file F, the user interacts with S-CSPs to perform the deduplication. More precisely, the user firstly computes and sends the file tag ϕ_F = TagGen(F) to S-CSPs for the file duplicate check.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

• If a duplicate is found, the user computes and sends $\phi_{F;idj}$=TagGen′(*F*, id_j)to the *j*-th server withidentity id_j via the secure channel for $1 \leq j \leq n$ (which could be implemented by a cryptographic hash function $H_j(F)$ related with index *j*). The reason for introducing an index *j* is to prevent the server from getting the shares of other S-CSPs for the same file or block, which will be explained in detail in the security analysis. If $\phi_{F;idj}$ matches the metadata stored with ϕ_F , the user will be provided a pointer for the shard stored at server id_j .

• Otherwise, if no duplicate is found, the user will proceed as follows. He runs the secret sharing algorithm SS over *F* to get $\{c_j\}$ = Share(*F*), where $c_j$is the *j*-th shard of *F* . He also computes $\phi_{F;idj}$ = TagGen′(*F*, id_j), which serves as the tag for the *j*-th S-CSP. Finally, the user uploads the set of values $\{\phi_F, c_j, \phi_{F;idj}\}$ to the S-CSP with identityid_jvia asecure channel. The S-CSP stores these values and returns a pointer back to the user for local storage.

File Download. To download a file *F*, the user firstdownloads the secret shares $\{c_j\}$ of the file from *k* out of *n* storage servers. Specifically, the user sends the pointerof *F* to *k* out of *n* S-CSPs. After gathering enough shares, the user reconstructs file *F* by using the algorithm of Recover($\{c_j\}$).

This approach provides fault tolerance and allows the user to remain accessible even if *any* limited subsets of storage servers fail.

(c) The Block-level Distributed Deduplication Sys-tem

In this section, we show how to achieve the fine-grained block-level distributed deduplication. In a block-level deduplication system, the user also needs to firstly perform the file-level deduplication before uploading his file. If no duplicate is found, the user divides this file into blocks and performs block-level deduplication. The system setup is the same as the file-level deduplication system, except the block size parameter will be defined additionally. Next, we give the details of the algorithms of File Upload and File Download.

File Upload. To upload a file*F* , the user first performsthe file-level deduplication by sending ϕ_F to the storage servers. If a duplicate is found, the user will perform the file-level deduplication, such as that in Section 3.2. Otherwise, if no duplicate is found, the user performs the block-level deduplication as follows.

He firstly divides *F* into a set of fragments $\{B_i\}$ (where $i = 1, 2, \cdots$). For each fragment B_i, the user will perform a block-level duplicate check by comput-ingϕ_{Bi} = TagGen(B_i), where the data processing and duplicate check of block-level deduplication is the same as that of file-level deduplication if the file *F* is replaced with block B_i.

Upon receiving block tags $\{\phi_{Bi}\}$, the server with identity id_j computes a block signal vector σ_{Bi} for each *i*.

• i) If σ_{Bi} =1, the user further computes and sends $\phi_{Bi;j}$=TagGen′(B_i, *j*)to the S-CSP with identityid_j. If it also matches the corresponding tag stored,S-CSP returns a block pointer of B_i to the user. Then, the user keeps the block pointer of B_i and does not need to upload B_i.

• ii) If σ_{Bi} =0, the user runs the secret sharing al-gorithm SS over B_i and gets $\{c_{ij}\}$ = Share(B_i), where c_{ij} is the *j*-th secret share of B_i. The user also computes $\phi_{Bi;j}$ for $1 \leq j \leq n$ and uploads the set of values $\{\phi_F, \phi_{F;idj}, c_{ij}, \phi_{Bi;j}\}$ to the server id_jvia a secure channel. The S-CSP returns thecorresponding pointers back to the user.

File Download. To download a file $F = \{B_i\}$, the user firstdownloads the secret shares $\{c_{ij}\}$ of all the blocks B_i in *F* from *k* out of*n*S-CSPs. Specifically, the user sends allthe pointers for $B_i$to *k* out of *n* servers. After gathering all the shares, the user reconstructs all the fragments B_i using the algorithm of Recover($\{\cdot\}$) and gets the file $F = \{B_i\}$.

IV. FURTHER ENHANCEMENT

(a)Distributed Deduplication System with Tag Consistency

In this section, we consider how to prevent a duplicate faking or maliciously-generated ciphertext replacement attack. A security notion of tag consistency has been formalized for this kind of attack [6]. In a deduplication storage system with tag consistency, it requires that no adversary is able to obtain the same tag from a pair of different messages with a non-negligible probability. This provides security guarantees against the duplicate faking attacks in which a message can be undetectably replaced by a fake one. In the previous related work on reliable deduplication over encrypted data, the tag consistency cannot be achieved as the tag is computed by the data owner from underlying data files, which cannot be verified by the storage server. As a result, if the data owner replaces and uploads another file that is different from the file corresponding to the tag, the following users who perform the duplicate check cannot detect this duplicate faking attack and extract the exact files they want. To solve this security weakness, [6] suggested computing the tag directly from the ciphertext by using a hash function. This solution obviously prevents the ciphertext

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

replacement attack because the cloud storage server is able to compute the tag by itself. However, such a method is unsuitable for the distributed storage system to realize the tag consistency. The challenge is that traditional secret sharing schemes are not deterministic. As aresult, the duplicate check for each share stored in differ-ent storage servers will not be the same for all users. In [11], though they mentioned the method of deterministic secret sharing scheme in the implementation, the tag was still computed from the whole file or ciphertext, which means the schemes in [11] cannot achieve the security against duplicate faking and replacement attacks.

Deterministic Secret Sharing Schemes

We formalize and present two new techniques for the construction of the deterministic secret sharing schemes. For simplicity, we present an example based on traditional Shamir's Secret Sharing scheme. The description of (k, n)-threshold in Shamir's secret sharing scheme is as follows. In the algorithm of Share, given a secret $\alpha \in Z_p$to be shared among n users for a prime p,choose at random a $(k-1)$-degree polynomial function $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{k-1}x^{k-1} \in Z_p[X]$suchthat $\alpha = f(0)$. The value of $f(i) \bmod p$ for $1 \leq i \leq n$ is computed as the i-th share. In the algorithm of Recover, Lagrange interpolation is used to compute α from any valid k shares.

The deterministic version of Shamir's secret sharing scheme is similar to the original one, except all the random coefficients $\{a_i\}$ are replaced with deterministic values. We describe two methods to realize the constructions of deterministic secret sharing schemes below.

The First Method

Share. To share a secret $\alpha \in Z_p$, it chooses at random a $(k-1)$-degree polynomial function $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{k-1}x^{k-1} \in Z_p[X]$such that $\alpha = f(0), a_i = H(\alpha \| i)$and p is a prime, where $H(\cdot)$is a hash function.The value of $f(i) \bmod p$ for $1 \leq i \leq n$ is computed as the i-th share and distributed to the corresponding owner.Recover. The description of algorithm Recover is the same with the traditional Shamir's secret sharing scheme by using Lagrange interpolation. The secret α can be recovered from any valid k shares.For files or blocks unknown to the adversary, these coefficients are also confidential if they are unpredictable. To show its security, these values can be also viewed as random coefficients in the random oracle model. Obviously, these methods can be also applied to the RSSS to realize deterministic sharing.

The Second Method

Obviously, the first method of deterministic secret sharing cannot prevent brute-force attack if the file is predictable. Thus, we show how to construct another deterministic secret sharing construction method to prevent the brute-force attack. Another entity, called key server, is introduced in this method, who is assumed to be honest and will not collude with the cloud storage server and other outside attackers. System Setup. Apart from the parameters for the first deterministic secret sharing scheme, the key server chooses a key pair (pk, sk) which can be initialized as RSA cryptosystem.

Share. To share a secret α, the user first computes $H(\alpha \| i)$ for $1 \leq i \leq k-1$. Then, he interacts with the key server in an oblivious way such that the key server generates a blind signature on each $H(\alpha \| i)$ with the secret key sk without knowing $H(\alpha \| i)$. For simplicity, we denote the signature as $\sigma_i = \varphi(H(\alpha \| i), sk)$, where φ is a signing algorithm. Finally, the owner of the secret chooses at random a $(k-1)$-degree polynomial function $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{k-1}x^{k-1} \in Z_p[X]$suchthat $\alpha = f(0)$ and $a_i = \sigma_i$. The value of $f(i) \bmod p$ for $1 \leq i \leq n$ is computed as the i-th share and distributed to the corresponding owner.

V. RESULTS & DISCSSIONS

We describe the implementation details of the proposed distributed deduplication systems in this section. The main tool for our new deduplication systems is the Ramp secret sharing scheme (RSSS) [7], [8]. The shares of a file are shared across multiple cloud storage servers in a secure way.The efficiency of the proposed distributed systems are mainly determined by the following three parameters of n, k, and r in RSSS. In this experiment, we choose 4KB as the default data block size, which has been widely adopted for block-level deduplication systems. We choose the hash function SHA-256 with an output size of 32 bytes. We implement the RSSS based on the Jerasure Version 1.2 [13]. We choose the erasure code in the (n, k, r)-RSSS whose generator matrix is a Cauchy matrix [14] for the data encoding and decoding. The storage blowup is determined by the parameters n, k, r. In more details, this value is $\frac{n}{k-r}$in theory.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

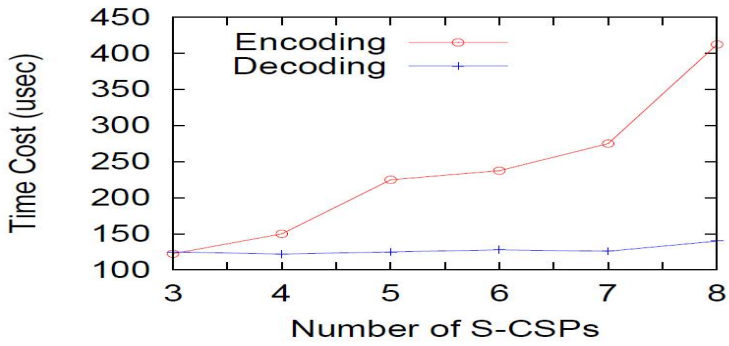
All our experiments were performed on an IntelrXeonr E5530 (2.40GHz) server with Linux 3.2.0-23-generic OS. In the deduplication systems, the (n, k, r)-RSSS has been used. For practice consideration, we test four cases:

- case 1: $r = 1$, $k = 2$, and $3 \leq n \leq 8$ (Figure 3(a));
- case 2: $r = 1$, $k = 3$ and $4 \leq n \leq 8$ (Figure 3(b));
- case 3: $r = 2$, $k = 3$, and $4 \leq n \leq 8$ (Figure 3(c));
- case 4: $r = 2$, $k = 4$, and $5 \leq n \leq 8$ (Figure 3(d)).

As shown in Figure 1, the encoding and decoding times of our deduplication systems for each block (per4KB data block) are always in the order of microseconds and hence are negligible compared to the data transfer performance in the Internet setting. We can also observe that the encoding time is higher than the decoding time. The reason for this result is that the encoding operation always involves all $n$shares, while the decoding operation only involves a subset of $k < n$ shares.

The performance of several basic modules in our constructions is tested in our experiment. First, The average time for generating a hash function with 32-byte output from a 4KB data block is 25.196 usec. The average time is 30 ms for generating a hash function with the same output length from a 4MB file, which only needs to be computed by the user for each file. Next, we focus on the evaluation with respect to some critical factors in the (n, k, r)-RSSS. First, we evaluate the efficiency between the computation and the number of S-CSPs. The results are given in Figure 2, which shows the encoding/decoding times versus the number of S-CSPs n. In this experiment, r is set to be2and the reliabilitylevel $n-k = 2$ are also fixed. From Figure 2, the encoding time increases with the number of n since more shares are involved in the encoding algorithm.

We also test the relation between the computational time and the parameter r. More specifically, in Figure 3, it shows the encoding/decoding times versus the confidentiality level r. To realize this test, the number of S-CSPs $n = 6$ and the reliability level $n-k = 2$ are fixed. From the figure, it can be easily found that the encoding/decoding time increases with r. Actually, this observation could also be derived from the theoretical result. If we recall that a secret is divided into $k-r$ equal-size pieces in the Share function of the RSSS. As a result, the size of each piece will increase with the size of r, which increases the encoding/decoding computational overhead. From this experiment, we can also conclude it will require much higher computational overhead in order to achieve higher confidentiality. In Figure 4, the relation of the factor of $n-k$ and the computational time is given, where the number of S-CSPs and the confidentiality level are fixed as $n = 6$ and $r = 2$. From the figure, we can see that with the increase of $n-k$, the encoding/decoding time decreases. The reason for this result is based on the RSSS, where fewer pieces (i.e., k) will be required with the increase of $n-k$.



(a) Impact on Encoding/Decoding time: case 1.

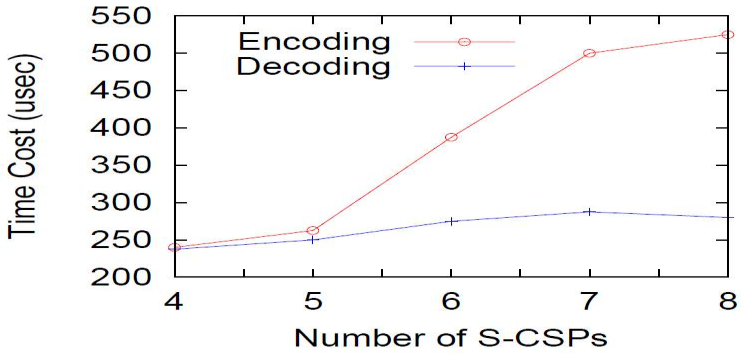
ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

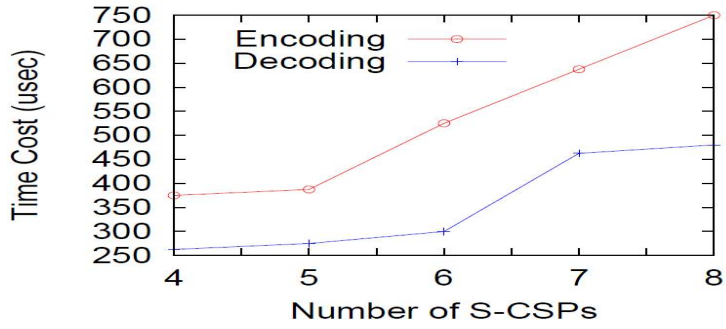
(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

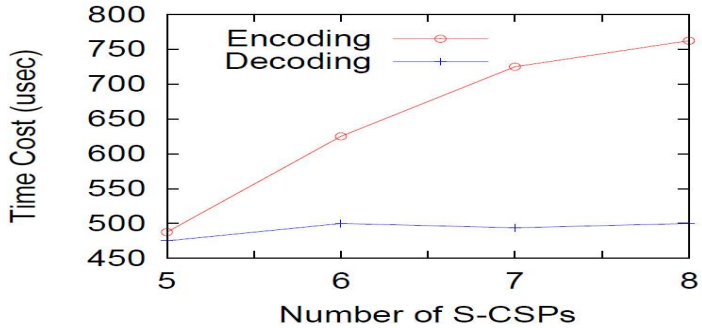
Vol. 6, Issue 9, September 2017



(b) Impact on Encoding/Decoding time: case 2.



(c) Impact on Encoding/Decoding time: case 3.



(d) Impact on Encoding/Decoding time: case 4.

Fig. 1. The Encoding and Decoding time for different RSSS parameters.

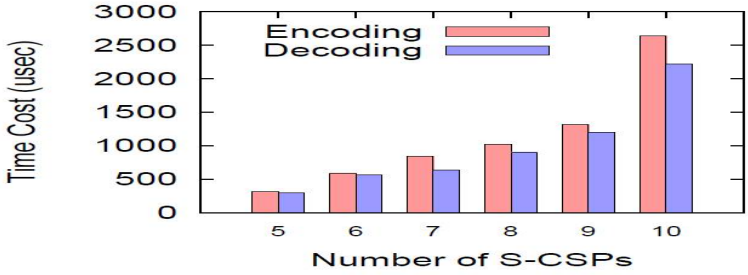


Fig. 2. Impact of number of S-CSPs n on encoding/decoding times, where $r = 2$ and $n-k = 2$.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

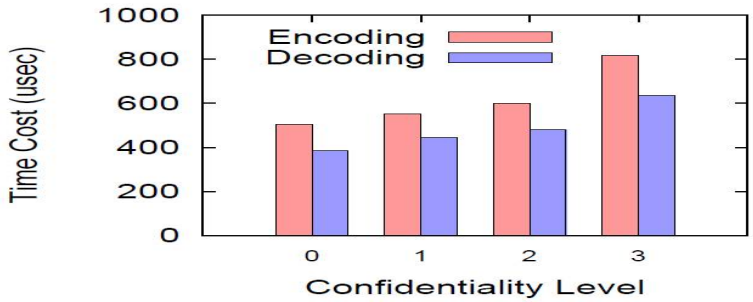


Fig 3 Impact of confidentiality level r on the encoding/decoding times where $n = 6$ and $n-k = 2$.

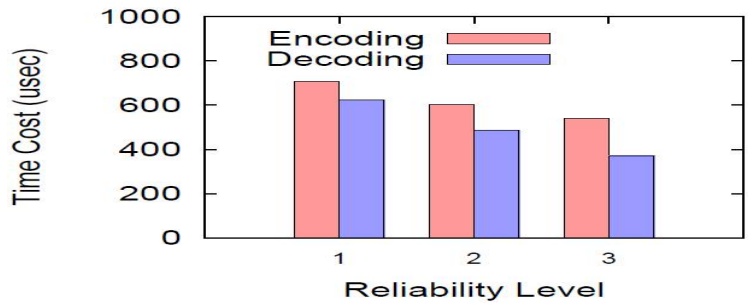


Fig. 4. Impact of reliability level $n-k$ on encoding/decoding times,where $n = 6$ and $r = 2$.

management in deduplication. However, they did not mention about the application of reliable deduplication for encrypted files. Later, in [16], they showed how to extend the method in [11] for the construction of reliable deduplication for user files. However, all of these works have not considered and achieved the tag consistency and integrity in the construction.

*Convergent encryption.*Convergent encryption [4]ensures data privacy in deduplication. Bellare et al. [6] formalized this primitive as message-locked encryption, and explored its application in space-efficient secure outsourced storage. There are also several implementations of convergent implementations of different convergent encryption variants for secure deduplication (e.g., [17], [18], [19], [20]). It is known that some commercial cloud storage providers, such as Bitcasa, also deploy convergent encryption [6]. Li et al. [11] addressed the key-management issue in block-level deduplication by distributing these keys across multiple servers after encrypting the files. Bellare et al. [5] showed how to protect data confidentiality by transforming the predicatable message into unpredictable message. In their system, another third party called the key server was introduced to generate the file tag for the duplicate check. Stanek et al. [21] presented a novel encryption scheme that provided differential security for popular and 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 deduplication was proposed for unpopular data. In this way, they achieved better tradeoff between the efficiency and security of the outsourced data.

Proof of ownership. Harnik et al. [22] presented a numberof attacks that can lead to data leakage in a cloud storage system supporting client-side deduplication. To prevent these attacks, Halevi et al. [12] proposed the notion of “proofs of ownership” (PoW) for deduplication systems, so that a client can efficiently prove to the cloud storage server that he/she owns a file without uploading the file itself. Several PoW constructions based on the Merkle Hash Tree are proposed [12] to enable client-side deduplication, which includes the bounded leakage setting. Pietro and Sorniotti [23] proposed another efficient PoW scheme by choosing the projection of a file onto some randomly selected bit-positions as the file proof. Note that all of the above schemes do not consider data privacy. Recently, Xu et al. [24] presented a PoW scheme that allows client-side deduplication in a bounded leakage setting with security in the random oracle model. Ng et al. [25] extended PoW for encrypted file, but they did not address how to minimize the key management overhead.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

*PoR/PDP.*Ateniese et al. [26] introduced the conceptof proof of data possession (PDP). This notion was introduced to allow a cloud client to verify the integrity of its data outsourced to the cloud in a very efficient way. Juels et al. [27] proposed the concept of proof of retrievability (PoR). Compared with PDP, PoR allows the cloud client to recover his outsourced data through the interactive proof with the server. This scheme was later improved by Shacham and Waters [28]. The main difference between the two notions is that PoR uses Error Correction/Erasure Codes to tolerate the damage to portions of the outsourced data.

VI. CONCLUSIONS

Cloud data storage service providers such as Mozy, Dropbox, and others perform deduplication to save space by only storing one copy of each file uploaded. While using cloud data storage clients always concern for confidentiality of data. So clients usually encrypt their data, and put it on to the storage. We proposed the distributed deduplication systems to improve the reliability of data while achieving the confidentiality of the users' outsourced data without an encryption mechanism. Four constructions were pro-posed to support file-level and fine-grained block-level data deduplication. The security of tag consistency and integrity were achieved. We implemented our deduplication systems using the Ramp secret sharing scheme and demonstrated that it incurs small encoding/decoding overhead compared to the network transmission over-head in regular upload/download operations.

REFERENCES

[1] Amazon, "Case Studies," https://aws.amazon.com/solutions/case-studies/#backup.
[2] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf, Dec 2012.
[3] M. O. Rabin, "Fingerprinting by random polynomials," Center for Research in Computing Technology, Harvard University, Tech. Rep. Tech. Report TR-CSE-03-01, 1981.
[4] 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 ICDCS, 2002, pp. 617–624.
[5] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: Server-aided encryption for deduplicated storage," in USENIX Security Symposium, 2013.
[6] ——, "Message-locked encryption and secure deduplication," in EUROCRYPT, 2013, pp. 296–312.
[7] G. R. Blakley and C. Meadows, "Security of ramp schemes," in Advances in Cryptology: Proceedings of CRYPTO '84, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds. Springer-Verlag Berlin/Heidelberg, 1985, vol. 196, pp. 242–268.
[8] A. D. Santis and B. Masucci, "Multiple ramp schemes," IEEE Transactions on Information Theory, vol. 45, no. 5, pp. 1720–1728, Jul. 1999.
[9] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," Journal of the ACM, vol. 36, no. 2, pp. 335–348, Apr. 1989.
[10] A. Shamir, "How to share a secret," Commun.ACM, vol. 22, no. 11, pp. 612–613, 1979.
[11] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplica-tion with efficient and reliable convergent key management," in IEEE Transactions on Parallel and Distributed Systems, 2014, pp. vol. 25(6), pp. 1615–1625.
[12] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems." in ACM Conference on Computer and Communications Security, Y. Chen, G. Danezis, and V. Shmatikov, Eds. ACM, 2011, pp. 491–500.
[13] J. S. Plank, S. Simmerman, and C. D. Schuman, "Jerasure: A library in C/C++ facilitating erasure coding for storage applica-tions - Version 1.2," University of Tennessee, Tech. Rep. CS-08-627, August 2008.
[14] J. S. Plank and L. Xu, "Optimizing Cauchy Reed-solomon Codes for fault-tolerant network storage applications," in NCA-06: 5th IEEE International Symposium on Network Computing Applications, Cambridge, MA, July 2006.
[15] C. Liu, Y. Gu, L. Sun, B. Yan, and D. Wang, "R-admad: High reliability provision for large-scale de-duplication archival storage systems," in Proceedings of the 23rd international conference on Supercomputing, pp. 370–379.
[16] M. Li, C. Qin, P. P. C. Lee, and J. Li, "Convergent dispersal: Toward storage-efficient security in a cloud-of-clouds," in The 6th USENIX Workshop on Hot Topics in Storage and File Systems, 2014.
[17] P. Anderson and L. Zhang, "Fast and secure laptop backups with encrypted de-duplication," in Proc. of USENIX LISA, 2010.
[18] Z. Wilcox-O'Hearn and B. Warner, "Tahoe: the least-authority filesystem," in Proc. of ACM StorageSS, 2008.
[19] A. Rahumed, H. C. H. Chen, Y. Tang, P. P. C. Lee, and J. C. S. Lui, "A secure cloud backup system with assured deletion and version control," in 3rd International Workshop on Security in Cloud Computing, 2011.
[20] M. W. Storer, K. Greenan, D. D. E. Long, and E. L. Miller, "Secure data deduplication," in Proc. of StorageSS, 2008.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 9, September 2017

BIOGRAPHY

Mr. G. Dileep Kumar has pursuing his Master of Technology in Computer Science & Engineering (CSE), CSE Department, Chadalawada Ramanamma Engineering College (CHDL) Tirupati, and Andhra Pradesh, India.

Prof.R.Suresh, working as a Professor in the department of Computer Science and Engineering, in CREC, Affiliated to JNTUA, Anantapuramu, possessed BE, M.Tech, Ph.D. He taught several subjects to UG and PG students during his 22 years teaching experience. His Research interests are Software Engineering, Object Oriented Systems Data Mining and Image Processing.