IJIRR

International Journal of Information Research and Review
Vol. 03, Issue, 08, pp. 2709-2718, August, 2016

Research Article

EFFICIENT DYNAMIC DATA AND INDIRECT MUTUAL TRUST FOR CLOUD STORAGE SECURITY SYSTEM

[1,*]Abdulrahman Saeed A. Noman, [2]Dr. Mohamed A. M. Ibrahim, and [3]Dr. Adel Sallam M. Haider

[1]MSc, Information Technology and Engineering, Aden University, Yemen,
[2]Faculty of Engineering and IT, Taiz University, Yemen
[3]Faculty of Engineering and IT, Aden University, Yemen

ARTICLE INFO

Article History:

Received 19th May 2016
Received in revised form
26th June 2016
Accepted 18th July 2016
Published online 30th August 2016

Keywords:

Access control,
Cloud computing,
Dynamicoperations,
Data security,
Data outsourcing,
Cloud service provider,
Mutual trust.

ABSTRACT

Storage-as-a-Service offered by cloud service providers (CSPs) is a paid facility that enables organizations to outsource their sensitive data to be stored on remote servers. For the effective utilization of sensitive data from CSP, we propose cloud-based storage scheme that allows the data owner to encrypt the sensitive data before outsourcing to the cloud server, and perform full block-level dynamic operations (modification, insertion, deletion, and append) on the outsourced data with great efficiency and minimal management overhead. To protect data in cloud, data privacy is the challenging task. In order to address this problem, the proposed scheme uses efficient data security system by using strong cryptographic techniques. The proposed scheme enables indirect mutual trust between data owner and CSP to detect the dishonest parties. Also it ensures the authorized users receive the latest version of the outsourced. Finally, we justify system performance through theoretical analysis to evaluate and reduce storage, communication, and computation overheads. The result shows that the proposed scheme is more efficient than the existing systems.

INTRODUCTION

Cloud computing is a form of computing which enables ubiquitous, on-demand network access, convenient, to a shared pool of configurable computing resources (e.g., networks, applications, services storage, servers) that can be rapidly provisioned with minimal management effort or service provider interaction (NIST). In cloud computing, data is stored in remote massivelyscalable data centers where compute resources can bedynamically shared to achieve significant economies of scale.The storage capacity needs to scale with compute resources toeffectively manage and gain maximum cloud benefits. Since the data owner physically releases sensitive datato a remote CSP, there are some concerns regarding *confidentiality*, *integrity*, and *access control* of the data.The confidentiality feature can be guaranteed by theowner via encrypting the data before outsourcing toremote servers. For verifying data integrity over cloudservers, researchers have proposed provable data possessiontechnique to validate the intactness of data storedon remote sites, such as PDP protocols (AyadBarsoum, 2012). Normally, traditional access control techniques assumethe existence of the data owner and the storageservers in the same trust domain. This assumption, however, no longer holds when the data is outsourcedto a remote CSP, which takes the full charge of the outsourceddata management, and resides outside the trustdomain of the data owner. A possible resolution can be obtained to allowthe owner to implement right to use control of the data storedon a remote untrusted CSP. Through this solution, the data is encrypted under a certain key, which is shared only with the authorized users. Theunauthorized users, including the CSP, are unable toaccess the data since they do not have the decryption key. This general solution has been widely incorporated into existing schemes (di Vimercati *et al.,* 2007), which aim at providing datastorage security on untrusted remote servers. Anotherclass of solutions utilizes attribute-based encryption toachieve fine-grained access control (Yu *et al.,* 2010). In this paper, we suggest a design that deals with important concerns associated to outsourcing the storage space of data, namely *dynamic data*, *newness*, *mutual trust*, and*access control*. The remotely stored data can be updated andscaled by the owner. After updating, authorized usersshould receive the latest version of the data (newnessproperty), Mutual trust between the dataowner and the CSP isintroduced to determine the dishonest party, misbehavior. The access control is measured, which permits the owner to revoke or grant rights of access to the outsourced

Corresponding autho: Abdulrahman Saeed A. Noman,
Information Technology and Engineering, Aden University, Yemen.

data. Finally, we have analyzedthe proposed scheme in terms of storage, communication andcomputational overheads. The result shows that the proposedsecurity system is more efficient than the existing securitysystem.

Paper Organization

The rest of this paper is organized as follows. Section 2 discusses the literature review. Section 3provides a description of architecture for model. Section 4 discusses the preliminaries of proposed system. Section 5discusses theprocedural steps of the proposed schemein detail. Section 6provides performance analysis for our system.Section 7conclusion our research and future work.

LITERATURE REVIEW

To ensure the data integrity of a file consisting of a finite ordered set of data blocks F = {bj} 1≤j≤m, in cloud server, there are several solutions are defined by Qian Wang etai, in (Cong Wang, KuiRen *et al.,* 2010). The first and straight forward solution to ensure the data integrity is, the data owner pre-compute the MACs for the entire file with a set of secrete keys, before our sourcing data to cloud server. During auditing process, for each time the data owner reveals the secret key to the cloud server and ask for new MAC for verification. This method takes the huge number of communication overhead for verification of entire file, which effect on the system efficiency.Another straight forward solution to overcome the drawback of previous method is to generate the signatures for every block instead of MACs to obtain the public audit-ability. This method again results in large communication overhead and effect the system efficiency. The above solutions are supports only static data and none of them can deal with the dynamic data updates.

Remark

In this straightforward solution using authentication tags (digital signatures) to detect cheating from any side (data owner or CSP). For each file F = {bj}1≤j≤m, the owner attaches digital signature OWNσj with each block before outsourcing to the CSP. The CSP first checked digital signature of owner before storing data on cloud. In case of failed verification, the CSP rejects to store the data blocks and asks the owner to re-send the correct tags. If the tags are valid, both the blocks and the tags are stored on the cloud servers.When an authoritative user (or the owner) requests to retrieve the data file, he CSP sends file, owner's signature, and CSP's signature {bj ,OWNσj , CSPσj}1≤j≤m. The user first verifies the tags {CSPσj}1≤j≤m. In case of failed verification, the user asks the CSP to re-perform the transmission process. If {CSPσj}1≤j≤m are valid tags, the user then verifies the owner's tag OWNσj on the block bj∀ j. If any tag OWNσj is not verified, this indicates the dishonesty of data more than the cloud servers. The CSP cannot reject such dishonesty for the owner's tags {OWNσj}1≤j≤m are previously verified and stored by the CSP along with the data blocks. Since the CSP's signatures {CSPσj}1≤j≤m are attached with the received data, a dishonest owner cannot falsely accuse the CSP regarding data integrity. Although the previous straightforward solution can detect cheating from either side, it cannot guarantee the newness property of the outsourced data. The above solution increases the storage overhead on the cloud servers. Moreover, there is an increased computation overhead on different system components; for a file F containing m blocks, it requires 2m signature generations and 3m signature verifications, which may be computationally a challenging task for large data files.

For example: if a file size 1GB with 4KB block size, then the number of blocks is 2^{18} , and the file requires 2×2^{18} signature generations and 3×2^{18} signature verifications. Note. We need large storage overhead on CSP, and large communication overhead, so that is effect on the system efficiency.

Qian Wang etai, in (Wang *et al.,* 2009) designed an efficient solution to support the public audit-ability without retrieving the data blocks from server. The design of dynamic data operations is a challenging task for cloud storage system. (JunbeomHur, 2013) explains the cryptographic based solution for data sharing using cipher-text policy attribute-based encryption (CP ABF) to improve the security of the data. The major drawback of this method is the unauthorized users can access the key to decrypt the encrypted data. Dubey et al. in 2012 developed a system using RSA and MD5 algorithms for avoiding unauthorized users to access data from cloud server. The main drawback of this method is that the CSP has also an equal control of data as the data owner and the computation load for CSP is proportional to the degree of connectivity so that the performance of the system can degrade.

MODEL / ARCHITECTURE

Background

The block diagram of cloud computing storage system is defined in the Fig 1. It contains there are four functional blocks for data storage and accessing data from data centers in public cloud servers such as, data owner, Cloud Service Provider(CSP), authorized users, and Trusted Third Party TTP (Prakash, 2014). The functions of these functional blocks are as follows:

Data owner: The data owner that can be any organization for generating outsourcing and sensitive data to be stored in data center of public cloud model for the external use on the demand of the authorized users on the basis of pay per usage.

Cloud Service ProviderCSP: Who manage the cloud servers and data centers in the public cloud and provide the storage infrastructure to the data owner for storage of outsourced data in data center on the payment based on the requested storage capacity. It coordinates the trusted third party to verify the authorized users and to retrieve the data from cloud server to make them available for authorized user on demand (AyadBarsoum and Anwar Hasan, 2012).

Users: the set of authorized users to access the remote data stored in cloud server through trusted third party and cloud service provider, all the users are the clients of the data owner.

Trusted Third Party: an entity who is trusted by all other entities of the system such as CSP, data owner and users. The functions of TTP is to verify whether the requested user in authorized or not, updating the block status table of file and calculate the hash value for file and block status table (Kuyoro *et al.,* 2011).

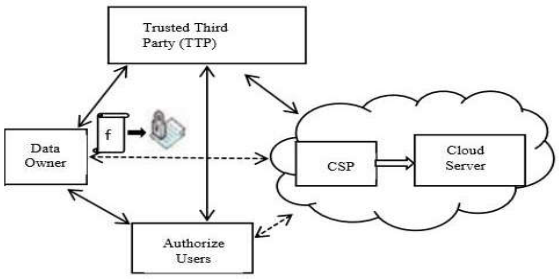

Fig. 1. Cloud computing data storage system model.

Assumptions

The following assumptions are considered to evaluate the data privacy of the proposed system:

- The data owner and users have mutual distrust relation with cloud service provider,
- Trusted third party helps to make the indirect mutual trust between authorized user and data owner with cloud service provider,
- Public cloud model is considered for storage of outsourced data in data centers

The data owner has a file (F) consisting of m blocks of equal size. Since the data is storing on remote data center, for confidentiality all the blocks are encrypted using symmetric data encryption algorithm before sending to the cloud server.

Security Requirements

Confidentiality: outsourced data must be protected from the TTP, the CSP, and users that are not granted access.

***Integrity*:** outsourced data is required to remain intact on cloud servers. The data owner and authorized users must be enabled to recognize data corruption over the CSP side.
***Newness*:** receiving the most recent version of the outsourced data file is an imperative requirement of cloud-based storage systems. There must be a detection mechanism if the CSP ignores any data-update requests issued by the owner.
***Access control*:** only authorized users are allowed to access the outsourced data. Revoked users can readunmodified data, however, they must not be able to read updated/new blocks.
***CSP's defence*:** the CSP must be safeguarded against false accusations that may be claimed by dishonest owner/users, and such a malicious behavior is required to be revealed.

Objectives

The objectives of our proposed security system are summarized as follows.

- Design an efficient data privacy algorithm using cryptographic techniques.
- Design an efficient data integrity using public trusted party auditing.
- Reduce the computational overheads of CSP, while introducing TTP. And reduce the storage overheads,
- Access the out sourced data, even if data owner is in off-line.
- Detect the dishonest party using combined hash values verification.

System Preliminaries

Overview

For enforcing privacy and access control of the outsourced data, the proposed system utilizes and combines four cryptographic techniques: lazy revocation, key rotation, and Broadcast Encryption, AES Encryption. And for efficient dynamic operation, the proposed system used a small part of the owner's work delegating to the TTP to reduce both the storage and computation overheads.

Lazy Revocation

In this work we allow the data owner to cancel the right of some users for accessing the outsourced data. In lazy revocation, it is suitable for users to read (decrypt) unchanged data blocks. However, modernized or new blocks must not be accessed by such revoked users. The idea is that allowing cancelled users to read unchanged information blocks is not a important loss in security. This is corresponding to accessing the blocks from cashed copies. Restructured or new blocks following a revocation are encrypted underneath latest keys. Lazy revocation trades re-encryption and data access charge for a degree of protection. However, it causes destruction of encryption keys, which is data blocks could have more than one key (Popa *et al.,* 2011).

Key Rotation

Key rotation is a technique in which a sequence of keys can be generated from an initial key and a master secret key (Kallahalla, *et al.,* 2003). The progression of keys has two main properties:

- only the owner of the master secret key is able to generate the next key in the sequence from the current key, and
- Any authorized user knowing a key in the sequence is able to generate all previous versions of that key.

In other words, given the *i*-th key *Ki* in the sequence, it is computationally infeasible to compute keys $\{K_l\}$ for $l > i$ without having the master secret key, but it is easy to compute keys $\{ Kj \}$ for $j < i$. The proposed scheme in this work utilizes the key rotation technique (Boneh *et al.,* 2005). Let $N = pq$denote the RSA modulus (p&$q$are prime numbers), a public key = (N, e), and a master top secret key d. The key d is known only to the data owner, and $ed \equiv 1 \bmod (p \quad 1)(q \quad 1)$. Whenever a user's access is revoked, the data owner generates a new key in the sequence (*rotating forward*). Let *ctr*indicate the index/version number of the current key in the keys sequence. The owner generates the next key as $Kctr+1 = K^{d}_{ctr} \bmod N$. Authorized users can recursively generate older versions of the current key as $Kctr \quad 1 = K^{e}_{ctr} \bmod N$ (*rotating backward*) (Ayad Barsoum and Anwar Hasan, 2012).

Broadcast Encryption

Broadcast encryption (bENC) allows a presenter to encrypt a message for an chance subset of a collection of users. The

users in the subset are only acceptable to decrypt the message. However, even if all users outside the subset scheme they cannot access the encrypted message. Such systems have the agreement struggling property, and are used in lots of practical applications as well as TV contribution services and DVD content protection. The proposed method in this work uses bENC to implement access control in outsourced data (Boneh *et al.,* 2005). The bENC is composed of three algorithms: SETUP, ENCRYPT, and DECRYPT.

1. Setup

This algorithm takes as contribution the number of system users n. It defines a bilinear group G of major order p with a generator g, a repeated multiplicative group GT, and a bilinear map ê : G×G → GT, which has the properties of bilinearity, computability, and no degeneracy (Amol B.,Shekhar, 2014). The algorithm picks a unsystematic α Є Zp, computes gi = g(αi) Є G for i = 1,2,...,n,n+2,...,2n, and sets v = gβ Є G for β Є R Zp. The outputs are a public key PK = (g, g1,...,gn, gn+2,...,g2n,v) Є G2n+1, also n private keys {di}1≤i≤n, where di = gi β Є G.

2. Encrypt

This algorithm takes as input a subset $S \subseteq \{1, 2, \dots, n\}$, and a public key *PK*. It outputs a pair (Hdr, *K*), where Hdr is called the header (broadcast ciphertext), and *K* is a message encryption key. Hdr = (*C*0, *C*1) ∈G2, where for $t \in_R Zp$, $C0 = gt$and $C1 = (v \cdot \pi j \in S\ gn+1\ \ j)^t$. The key $K = \hat{e}(gn+1, g)^t$is used to encrypt a message *M* (symmetric encryption) to be broadcast to the subset *S*.

3. Decrypt

This algorithm takes as contribution a subset S Є {1,2,...,n}, a user-ID i Є {1,2,...,n}, the private key di for user i, the header Hdr = (C0,C1), and the public key PK. If i Є S, the algorithm outputs the key K = ê(gi,C1)/ê(di .π j Є S ,j≠I gn+1 j+i,C0), which can be used to decrypt the encrypted description of M. In the above structure of the bENC, a private key contains only one factor of G, and the broadcast cipher text (Hdr) consists of two factors of G. On the further hand, the public key PK is comprised of 2n + 1 factors of G. A second structure, which is a simplification of the first one, was accessible in to trade the PK size for the Hdr size. The main idea is to run several parallel instances of the first structure, where each instance can broadcast to at most B users. Setting B= $[\sqrt{n}]$ results in a system with (O$\sqrt{n}$) factors of G for each of PK and Hdr. The private key is at a standstill just one factor.

Block Status Table

The proposed system used a small part of the owner's work delegating to the TTP to reduce both the storage and computation overheads. The Block Status Table (BST) is a small dynamic data structure used to access the outsourced encrypted file from the cloud service provider. It consists of three columns: serial number (*SN*), block number (*BN*), and key version (*KV*). *SN* is an indexing to the file blocks. It indicates the physical position of each block in the data file. *BN* is a counter used to make a logical numbering/indexing to the file blocks. *KV* indicates the version of the key that is used to encrypt each block in the data file. The BST is implemented as a linked list to simplify the insertion and deletion of table entries. During implementation, SN is not needed to be stored in the table; SN is considered to be the entry/table index. Thus, each table entry contains just two integers BN and KV (8 bytes), i.e., the total table size is 8m bytes, where m is the number of file blocks.When a data file is initially created, the owner initializes both ctr and KV of each block to 1. If block modification or insertion operations are to be performed following a revocation, ctr is incremented by 1 and KV of that modified/new block is set to be equal to ctr (Ayad Barsoum and Anwar Hasan, 2012).

Ctr = 1		
SN	***BN***	***KV***
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1

Table 1. Structure of Block Status Table

Procedural steps of the proposed scheme

Setup and File Preparation

The proposed system consists of a three functional roles to setup file F; data owner, trusted third party and cloud serviceprovider.Fig. 2 describes the role of each system component (owner, CSP, and TTP) during setupand File Preparation,

Owner Role

The data ownerinitializes *ctr*to 1, and generates an initial secret key K_{ctr} is *K*1. K_{ctr}can be rotated forward following user revocations, to generate the new key for next block data encryption, and rotated backward for previous block encryption to enable authorized users to access blocks that are encrypted under older versions of K_{ctr}. For a file $F = \{bj\}1 \le j \le m$, the owner generates a BST with $SNj = BNj = j$ and $KVj = ctr$. To achieve privacy-preserving, the owner creates an encrypted file version $F' = \{b'_j\}1 \le j \le m$, where $b'_j = E_{Kctr}(BN_j \,||b_j)$ Moreover, the owner creates a rotator *Rot* = (*ctr,* bENC(*Kctr*)), where bENC enables only authorized users to decrypt $K_{ctr}$and access the outsourced file. The owner sends *{F',* BST*, Rot}* to the TTP, and deletes the data file from its local storage. Embedding *BNj*with the block *bj*during the encryption process supports in reconstructing the file blocks in the correct order (more details will be explained later). The owner keeps and stores a local copy of the BST, to validate each time the data owner wants to issue a dynamic request. This help to avoid such communication and computation overheads,

Notations

- $F = \{b1, b2, \dots, bm\}$ is a data file.
- *h*is a cryptographic hash function.
- $E_{Kctr}$is a symmetric encryption algorithm under K_{ctr}, e.g., AES encryption.

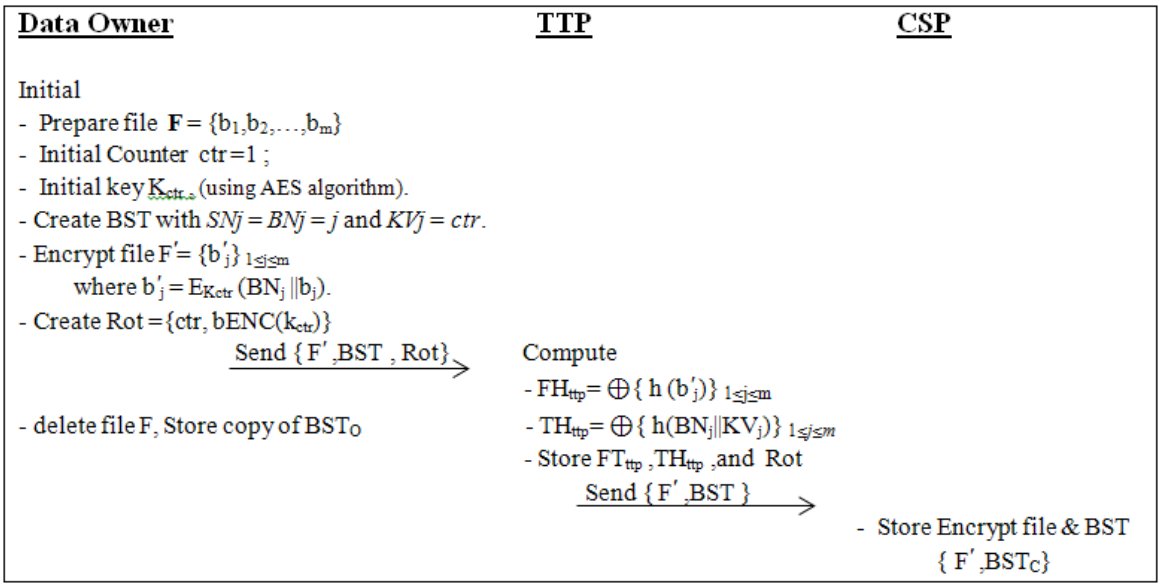

Fig. 2. Setup and File Preparationprocedure in the proposed scheme

- FH_{ttp} is a combined hash value for F', and is computed and stored by the TTP.
- TH_{ttp} is a combined hash value for the BST, and is computed and stored by the TTP
- *ctr*is a counter kept by the data owner to indicate the version of the most recent key
- *BST* is a Block Status Table.
- BST_O is block status table at owner, *and* BST_C is it at CSP.
- $\oplus$ is an XOR operator .
- Rot = (ctr, bENC(K_{ctr})) is a rotator, where bENC(K_{ctr}) is a broadcast encryption of K_{ctr}

TTP Role: The TTP receives the {F',BST, Rot} from data owner, then resolve disputes that may arise regarding data integrity/newness, the TTP computes combined hash values for the encrypted file F and BST using the following formula.

FH_{TTP} = h (b1 ⊕ b2 ⊕ . . . ⊕ bm) = ⊕{h(bj)}1≤j≤m …... (1)

TH_{TTP} = h ((BN1||KV1)⊕(BN2||KV2)⊕. . . ⊕ (BNm||KVm))= ⊕{h(BNj||KVj)} 1≤j≤m …... (2)

Then sends *{F′,* BST*}* to the CSP. The TTP keeps only $FH_{TTP}$and TH_{TTP}onits local storage.

CSP Role

The CSP receives the {F′, BST} from TTP and stores a copy of the BST along with the outsourced data file. When a user requests to access the data, the CSP responds by sending both the BST_C and the encrypted file F′.

Table 2. The data stored by each component in the system

Owner	TTP	CSP
Ctr, K_{ctr}, BST_O	Rot, FH_{TTP}, TH_{TTP}	F', BSP_C

Dynamic Operations on the Outsourced Data

The BST is implemented as a connected list to make things easier the insertion, deletion and modification of table entries. If block alteration or addition operations are to perform following a revocation, ctr is incremented by 1 and KV of that customized/new block is set to be equal to ctr. Fig. 3: change in BST Due to Different active Operation on a File F={bj}1≤j≤8 When a data file is initially created, the data owner initializes both ctr and KV of each block to 1. If block alteration or placing operations are to be performed following a revocation, ctr is incremented by 1 and KV of that customized/new block is set to be equal to ctr. Figure shows some examples representing the changes in the BST due to dynamic operations on a data file F = {bj}1≤j≤8. When the file blocks are initially formed (Fig.(a)) ctr is initialized to 1, SNj =BNj = j, and KVj = 1: 1 ≤ j ≤ 8. Fig. (b) Shows no modify for update the block at location 5 since no revocation is performed. To add a new block after location 3 in the file F. Fig (c) shows that a new entry {4,9,1} is added in the BST after SN3, where 4 is the physical location of the newly added block, 9 is the new logical block number compute by incrementing the maximum of all previous logical block numbers, and 1 is the version of the key used for encryption.

A first revocation in the scheme increments ctr by 1 (ctr = 2). Modifying the block at position 5 following a revocation (Fig.(d)) answers in setting KV5 = ctr. Thus, the table entries at location 5 become {5, 4, 2}. (Fig. (e)) shows that a new block is to be added after position 6 following a second revocation, which Increments ctr to be 3. In Fig. (e), a new table entry {7, 10, 3} is insert after SN6, where KV7 is set to be equal to ctr (the Most recent key version). Deleting a block at position 2 from the Data file requires deleting the table entry at SN2 and shifting all Ensuing entries one position up. Note that during all Dynamic operations, SN indicates the real physical positions of the information blocks in F.

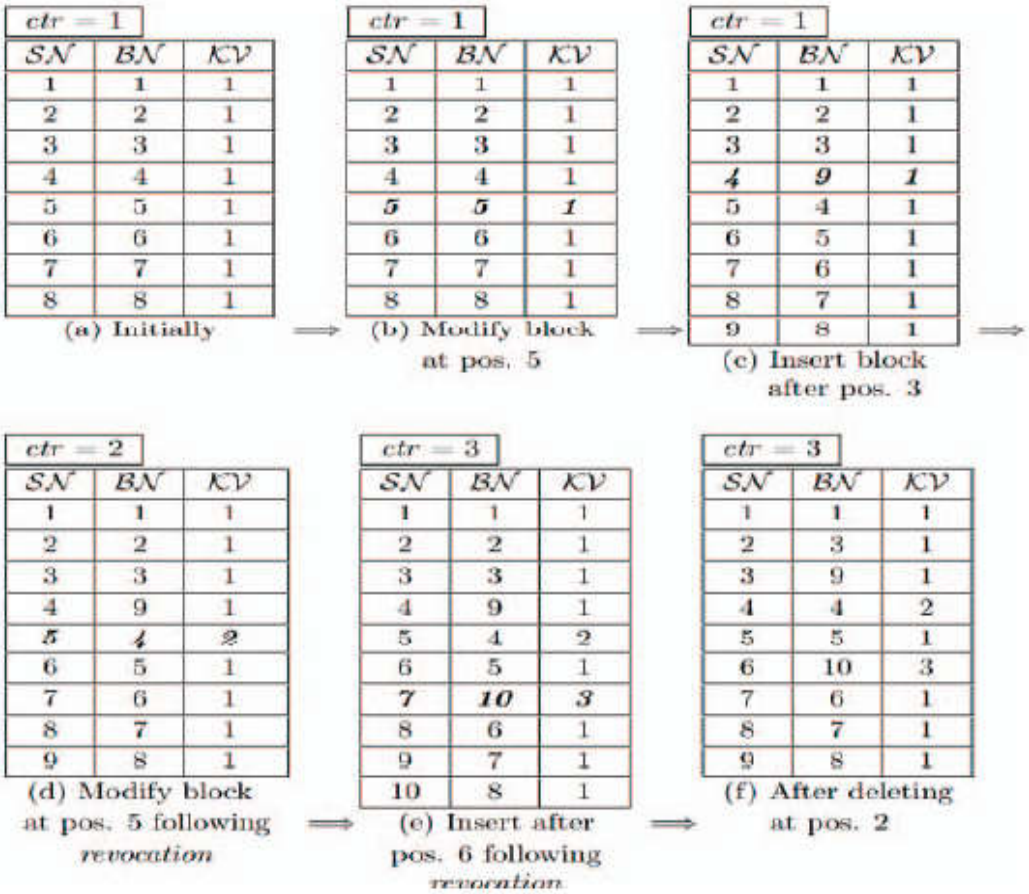

Fig. 3. Changes in the BST due to different dynamic operations on a file F = {bj} 1≤j≤8

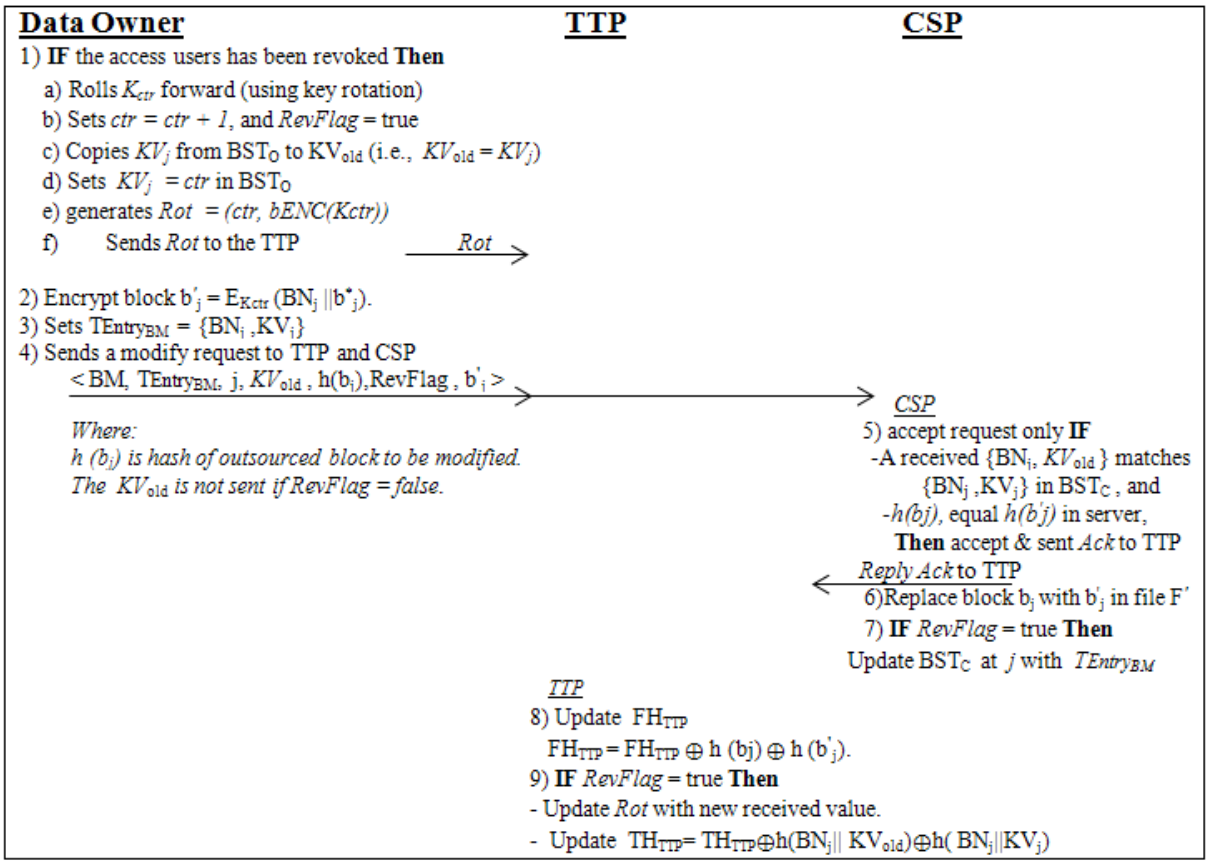

Fig. 4. The Steps ofBlock Modification in the proposed scheme

Block Modification Operation

The dynamic operations in the proposed scheme are performed at the block level via a request in the general form*(BlockOp, $TEntry_{BlockOp}$, j, KVj, h(b'$_j$),RevFlag, b*)*

Where:

BlockOp corresponds to blockmodification (denoted by BM), block insertion (denotedby BI), or block deletion (denoted by BD).$TEntry_{BlockOp}$indicates an entry in BST_Ocorresponding to the issued dynamic request. The parameter *j* indicates the block index,on which the dynamic operation is to be performed. *KVj*is the value of the key version at index *j* of BST_Obefore running a modification operation. *h*(b′$_j$) isthe hash value of the block at index *j* before modification/deletion. The *RevFlag* is a 1-bit flag (true/false andis initialized to false) to indicate whether a revocationhas been performed, and b*is the new block value. For a file *F = {b1, b2, . . . ,bm}*, supposethe owner wants to modify a block *bj*with a block b*$_j$. Fig. 4 describes the steps performed by each systemcomponent (owner, CSP, and TTP) during blockmodification. The owner send the modify request to both the CSP and the TTP. The TTP updates the combined hash value $FH_{TTP}$for F'which simultaneously replaces the hash of the old block*h*(bj)with the new one *h*(b'*j*), through the step $FH_{TTP}= FH_{TTP}\oplus h(bj)\oplus h(b'j)$. This is possible due to thebasic properties of the $\oplus$operator. The same idea is usedwhen RevFlag = true to update the value TH_{TTP}.

Block Insertion Operation

In a block insertion operation, the owner wants to insert a new block b'after index *j* in a file F = {b1, b2, … ,bm},the newly constructed file. F'= *{b1, b2,... ,bj, b', ... , bm*+1*}*, where *bj*+1 = b'. Fig. 5 describes the steps performed by each system component(owner, CSP, and TTP) during block insertion.

Block Deletion Operation

When one block is deleted all subsequentblocks are moved one step forward. Fig. 6 describes the steps performed by each system component (owner, CSP, and TTP) during block deletion. The step $FH_{TTP}= FH_{TTP}\oplus h(b'j)$ is used to delete the hash value of$b'j$from the combined hash FH_{TTP} . The same idea is used with the TH_{TTP}value.

Data Access and Cheating Detection

The authorized user sends a request to both the CSP and TTP to access the outsourced file from the cloud server for retrieve the data file. For achieving non-repudiation, the CSP generatestwo signatures $\sigma_F$and $\sigma_T$for$F'$and BST_C, respectively.After receiving {F′, BST_C ,σ_F,σ_T} from the CSP, and {FHttp,THttp , Rot} from the TTP. The authorized user verifies the signatures, and proceeds with the data access procedure only if both signatures are valid.the user verifies the contents of BST_C entries by computing the combined hash values of BST_C using the following equation.

$$TH_{User} = h((BN1||KV1)\oplus(BN2||KV2)\oplus \ldots \oplus(BNm||KVm)) = \oplus\{h(BNj||KVj)\}\ 1\le j\le m \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The user compares the TH_{ttp} received from TTP and TH_{User}. If(TH_{User}!=TH_{ttp}), then issue a dishonest party report to the TTP and data owner. In case of (TH_{User}==TH_{ttp}) the user verifies the contents of the encrypted file F by calculating the FH_{User} using the following equation and comparing with FH_{ttp}.

$$FH_{User} = h(b'_1\oplus b'_2\oplus \ldots \oplus b'_m) = \oplus\{h(b'_j)\}\ 1\le j\le m \ \ldots.. (4)$$

If (FH_{User}! = FH_{ttp}), then informs to the TTP to resolve such a conflict. For the authorized user to access the encrypted file F' ={b′$_j$}1≤j≤m ,$BST_C$and *Rot*are usedto generate the key K_{ctr} that decrypts the block b'_j, The component bENC(K_{ctr}) of *Rot* is decrypted to get the most recent key K_{ctr}. Using the key rotation technique, the user rotates K_{ctr}backward with each block until it reaches the version that is used to decrypt the blocks b'_jusing the AES decryption algorithm, Both *ctr*andthe key version *KVj*can determine how many rotationsteps for *Kctr*with each block b'_j. Decrypting the blockb'_jreturns (BN_j|| b_j). The BN_j and BST_C are utilized to reconstruct the original source file (F).

Performance analysis

Overview

We evaluate the performance of the proposed cloud security scheme by analyzing the storage, communication and computation overhead, the data file F used in our performance analysis is ofsize 1GB with different block size (4KB, 8KB,and 16KB) to reduce overheads, and optimize efficient of the system.We assume that the desired security level is 128-bit, and we utilize a cryptographic hash *h* of size 256 bits (32 bytes), an elliptic curve used to implement bENCcan be represented by 256 bits (≈ 32 bytes), and BLS(Boneh-Lynn-Shacham) signature of size 256 bits(32 bytes) used to compute $\sigma_F$and σ_T.

Storage Overhead

It is the additional storage space required to store thenecessary information for data security, other than the stored file F. The overheadon the data owner is due to store BST_O in its local storage, where the size of the BST_O is calculated based on the number of data blocks m. The BST_O contains two columns *BN,KV* both are integer it occupies total 8 bytes of memory space for single data block, therefore the total storage space of BST_O is 8xm bytes, where m is the number of data blocks. Like the owner, the storage overhead on the CSP sidecomes from the storage of BST_C is 8xm bytes. To resolve disputes thatmay arise regarding data integrity or newness property,the TTP stores $FH_{TTP}$and TH_{TTP}*(hash values of the file and BST is 64 bytes)* each of size 32 bytes. Besides, the TTP stores *Rot* ={*ctr,* bENC(K_{ctr})}that enables the data owner to enforce access controlfor the outsourced data. The *ctr*is 4 bytes, and bENC is(32 bytes). ThebENChas storage complexity $O(\sqrt{n})$, where *n* denote the total number of system users.The total storage overhead of the proposed security system is the sum of the storage overhead at data owner, TTP and CSP side. The overall storage overhead is calculated using the following equation.

$$\begin{aligned} Overhead_{storage} &= overhead_{(Owner)} + overhead_{(CSP)} + overhead_{(TTP)} \\ &= BST_O + BST_C + (FH_{TTP} + TH_{TTP} + Rot) \\ &= \{(8 \times m) + (8 \times m) + (68+32(\sqrt{\boldsymbol{n}}))\} \text{ bytes} \end{aligned}$$

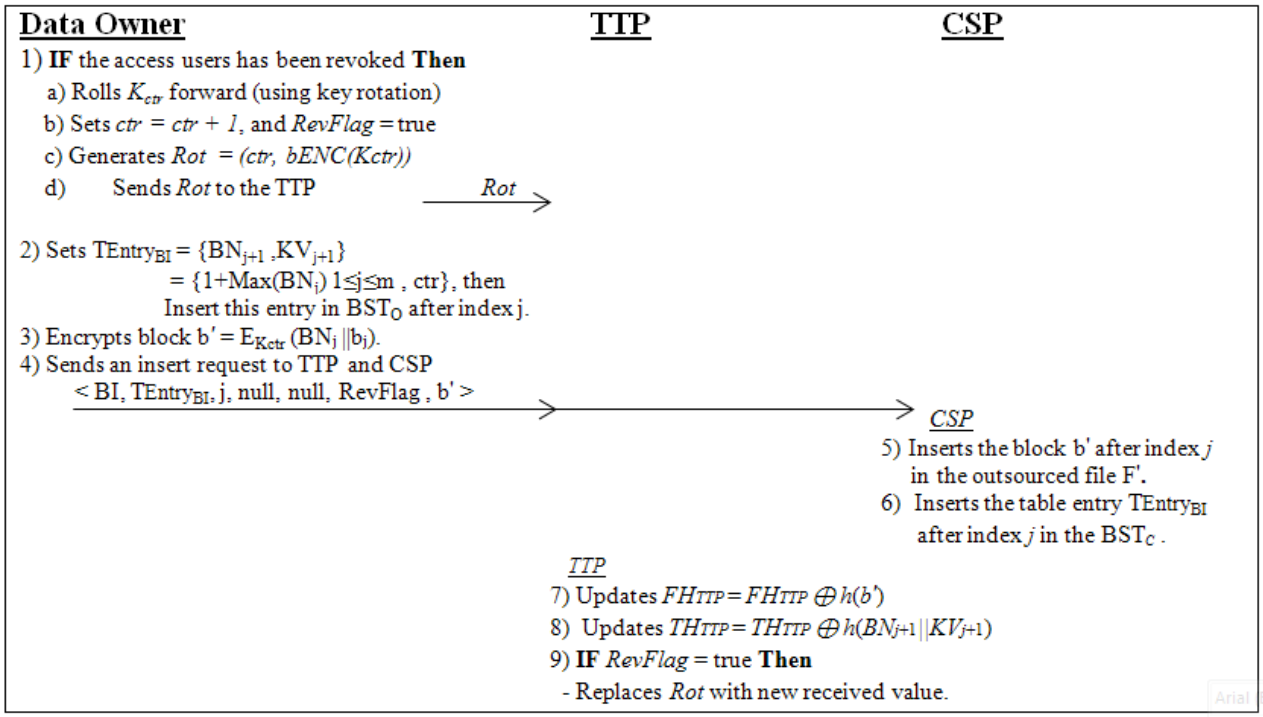

Fig. 5. The Steps ofBlock Insertion Operation. (Insert block b' after index j in the outsourced file)

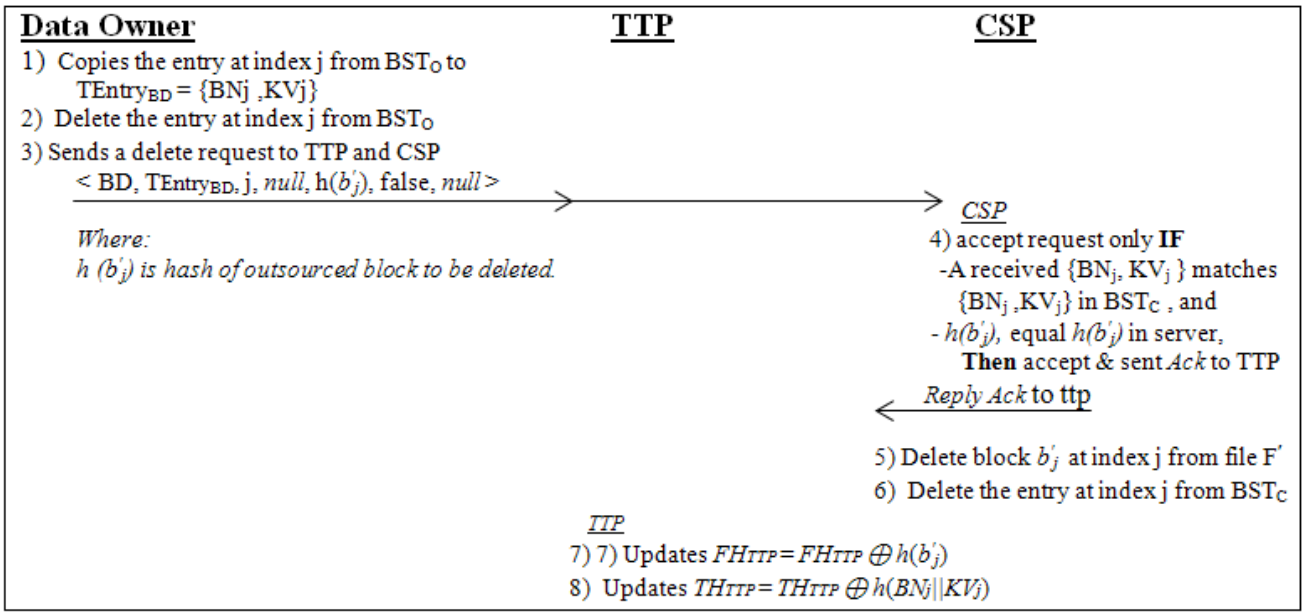

Fig. 6. The Steps ofBlock Deletion in the proposed scheme

For our data file F (1GB with 4KB block size) and n = 100,000 authorized users. The BST_O for the file *F* is only2MB (0.2% of *F*), also the BST_C is only2MB (0.2% of *F*). Therefore, the storage overhead onthe TTP side is close to 10KB. Overall, the storage overheadfor the file *F* is less than 4.01MB (≈0.4% of *F*). While increasing the data block size still we can reduce the BSTstorage overhead, to reduce the total storage overhead, and optimize the efficient of the system. So, if our data file F (1GB with 8KB block size). Then the BST_O is 1MB (0.1% of *F*), and the BST_C is 1MB (0.1% of *F*). The total storage overheadfor the file *F* is less than 2.01MB (≈0.2% of *F*). If data file F (1GB with 16KB block size). The BST_O is 0.5MB, and the BST_C is 0.5MB, the total storage overheadis less than 1MB (≈0.1% of *F*) and thus.

Communication Overhead

It is the additional information sent along with the outsourced data blocks whether during dynamic operations or during data access.*During dynamic operations*, the communication overhead on the owner side comes from the transmission general formulaof a block operations {BlockOp, $TEntry_{BlockOp}$, j, KVj, $h(b'_j)$}={1 + 8 + 4 + 4 + 32}bytes, sent to TTP, and the owner also sends *Rot* (4 + $32\sqrt{n}$bytes) to the TTP. The *Rot* represents the major factor in the communication overhead, and thus the overhead is 49 bytes if block modification/deletion with revocations, and is only 45 bytes without revocations (only 13 bytes for insertion operations).

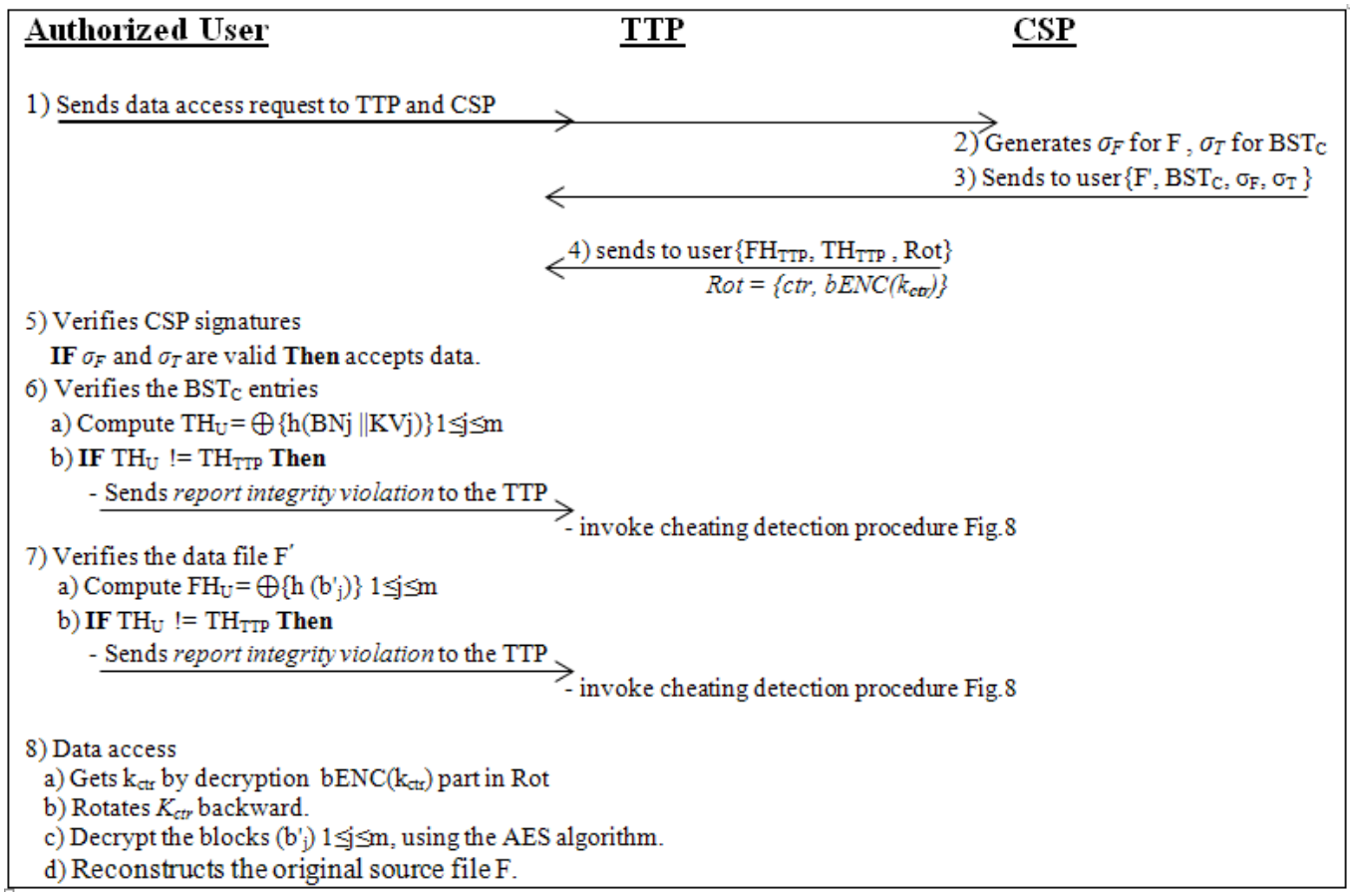

Fig. 7. Data access procedure in the proposed scheme

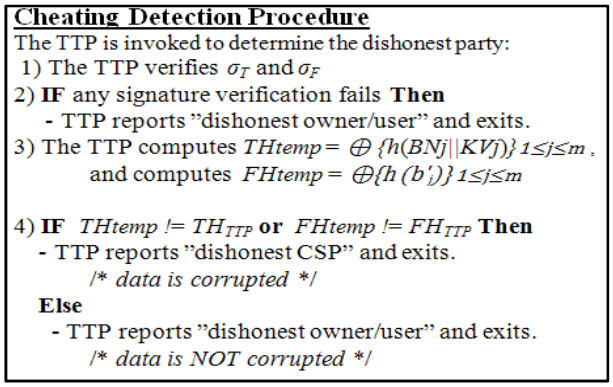

Fig. 8. Cheating detection procedure in the proposed scheme

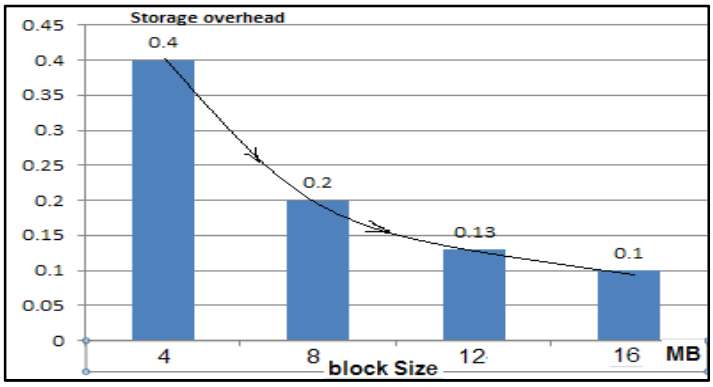

Fig. 9. Reduce storage overhead due to dynamic operations

Therefore, in the worst case scenario (i.e., block modifications following revocations), the owner's overhead is less than 10KB.Hence, the communication overhead due to dynamic changes on the data is about 1% of the block size (the block is 4KB in our analysis). As a response to *access the outsourced data*, the CSP sends the file along with σ_F (32 bytes), σ_T (32 bytes), and BST_C (8m bytes). Moreover, the TTP sends FH_{TTP} (32 bytes), TH_{TTP} (32 bytes), and Rot. Thus, the communication overhead due to data access is 64 + 8m bytes on the CSP side, and 68 + 32√n bytes on the TTP side. Overall, to access the file F(1GB with 4KB block size), the proposed scheme has communication overhead close to 2.01MB (≈ 0.2% of F). For getting the more efficient system, we can reduce the communication overhead, by increasing the data block size.

The total communication overhead for the system is the sum of the communication overheads between the *owner, user, CSP and TTP.*The overallcommunication overhead is calculated using the followingequation.

$Overhead_{Commn} = overhead_{(Owner,TTP)} + overhead_{(User,CSP)} + overhead_{(User,TTP)}$

$= \{(45 + [8 + 32\sqrt{\boldsymbol{n}}]) + (64 + 8m) + (68 + 32\sqrt{\boldsymbol{n}})\}$ bytes

Computation Overhead

For confidentiality requirement the static data in the cloud storage system has the computational cost for data access from the CSP. The computation cost is the cost of the time required

to perform the data encryption, data decryption. For the proposed scheme, thecomputation overhead on the owner side due to dynamicoperations (modification/insertion),the overhead is one encryption operations for block, forward rotation,and bENC to generate the *Rot*.Hence, the computation overhead on the owner side forthe dynamic operations is E_{Kctr} + *FR* + *bENC*(worst case scenario). And the computation overhead on theTTP side for updating values of both $FH_{TTP}$and $TH_{TTP}$is *4h*maximum at revocation or (2h at no revocation).Thetotal computation overhead for *data access* is the sum of thecost of the four verifications (2 signatures, CSP_C ,and file F), key rotation backward, and broadcast decryption$\{2V\sigma + 2mh + BR + bENC^{-1}\}$ on the owner side, and 2 generate signature $\{2S\sigma\}$on the CSPside. The maximum computation overhead required fordetecting the unauthorized user on the TTP side is $\{2V\sigma + 2mh\}$. The overallcomputation overhead is calculated using the followingequation.

$Cost_{Computation} = Cost_{Dynamic\_operation} + Cost_{Data\_access} + Cost_{Detection\_cheating}$

$= (E_{Kctr} + FR + bENC + 4h) + (2V\sigma + 2mh + BR + bENC^{-1} + 2S\sigma) + (2V\sigma + 2mh)$

$= E_{Kctr} + FR + BR + bENC + bENC^{-1} + 2S\sigma + 4h + 4mh + 4V\sigma$

Conclusion and future work

In this paper, we discuss the problem of data security in cloud storage system. To control the outsourced data and provide the quality of the cloud storage service for the users, we proposed a cloud-based storage scheme that provides a higher security and privacy to our data by maintaining encryption and decryption standards. Our data is provided with better security, data integrity, privacy, and access control of the outsourced data. Our proposed scheme supports outsourcing of dynamic data, where the owner is capable of not only archiving and accessing the data stored by the CSP, but also updating and scaling this data on the remote servers. And it enables the authorized users to ensure that they are receiving the most recent version of the outsourced data. To detect the dishonest party (owner/users or Cloud Service Provider), we implemented the verification techniques using hash function at TTP. The data owner enforces access control for the outsourced data by combining three cryptographic techniques: broadcast encryption, lazy revocation, and key rotation.

We have investigated the computation overhead, communication overhead and storage overhead for the outsourced data. We try to optimize the system overheads and enhanced efficient the system by using different blocks size. Such as the storage overhead is 0.4% of the outsourced data file, reduced to 0.1%, and thus. In the future work, we are focusing on the efficient authorized auditing to control of dynamic fine-grained data updates, when the verification is done by trusted third party.

REFERENCES

Amol, B., Shekhar, J., Chirag, C., Kavita, K. 2014. 'Indirect Mutual Trust and Allowing DynamicData for Cloud Storage System'in *International Journal of Engineering Research & Technology (IJERT),* Vol. 3 Issue 5, May.

AyadBarsoum and Anwar Hasan, Enabling Dynamic Data and IndirectMutual Trust for Cloud Computing Storage Systems, In IEEETransac-tions on Parallel and Distributed Systems, 2012.

Boneh, D., C. Gentry, and B. Waters, 2005. "Collusion resistant broadcast encryption with short ciphertexts and private keys," in Advancesin Cryptology – CRYPTO, pp. 258–275.

Cong Wang, KuiRen, W Lou, Jin Li, 2010. Toward Publicly Auditable Secure Cloud Data Storage Services, In IEEE Computer Networks, pages 19-24.

di Vimercati, S. D. C., S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, 2007. "Over-encryption: Management of access control evolution on outsourced data," in Proceedings of the 33rd International Conference on Very Large Data Bases. *ACM*, pp. 123–134.

Dubey, A. K., Dubey, A. K., Namdev, M., Shrivastava, S. S. 2012. Cloud user Security based on RSA and MD5 Algorithm for Resource Attestation and Sharing in Java Environment, Software Engineering(CONSEG),CSI Sixth International Conference on, pages 18,September.

Goyal, V., O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," inCCS '06, 2006, pp. 89–98.

JunbeomHur, 2013. Improving Security and Efficiency in Attribute Based Data Sharing, In IEEE Transactions on Knowledge and Data Engineering, Volume: 25,Issue: 10, pages 2271-2282.

Kallahalla, M., E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, 2003. "Plutus: Scalable secure file sharing on untrusted storage," inProceedings of the FAST 03: File and Storage Technologies.

Kuyoro, S. O., Ibikunle, F., Awodele, O. 2011. Cloud Computing Security Issues and Challenges, In International Journal of Computer Networks, vo!.3, issue 5.

Kuyoro, S. O., Ibikunle, F., Awodele, O. 2011. Cloud Computing Security Issuesand Challenges, In *International Journal of ComputerNetworks*, vo!.3, issue 5.

NIST SP 800-145, "A NIST definition of cloud computing",http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf

Popa, R. A., J. R. Lorch, D. Molnar, H. J. Wang, and L. Zhuang, 2011. "Enabling security in cloud storage SLAs with cloud proof," inProceedings of the 2011 USENIX conference.

Prakash, G. L., Manish, P., Inder Singh, 2014. Efficient Data Security Method to Control Data in Cloud Storage System using Cryptographic Techniques, in IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE),May 09-11, Jaipur, India

Wang, Q., C. Wang, J. Li, K. Ren, and W. Lou, 2009. Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing,InProceeding of 14th European Symposium, Research in Computer Security (ESORICS 09), pages 355-370.

Yu, S., C. Wang, K. Ren, and W. Lou, 2010. "Achieving secure, scalable, and fine-grained data access control in cloud computing" in INFOCOM'10, pp.534–542 .

****