ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.5, Pages : 260 - 266 (2014)
Special Issue of ICACSSE 2014 - Held on October 10, 2014 in St.Ann's College of Engineering & Technology, Chirala, Andhra Pradesh

Facilitating Indirect Mutual Trust for Cloud Storage Systems

Vaddepati Venkata Srikanth[1], Eswar Kodali[2]

[1]M.Tech Student, Dept. of CSE St. Ann's College of Engineering and Technology, Chirala, AP, INDIA, srikanthvaddepati@gmail.com
[2]Associate Professor, Dept. of CSE St. Ann's College of Engineering and Technology, Chirala, AP, INDIA, kodali_eswar@yahoo.co.in

Abstract : Storage as a service (SaaS) offered by cloud service providers (CSP) is a paid service that allows organizations to store their sensitive data to be on off-lying servers. A storage scheme is proposed that allows the data owner to make efficient use of services provided by the CSP and facilitate indirect mutual trust between them. The proposed scheme consists of four prominent features: (i) it allows the data owner to store sensitive data on an off-lying server, and perform complete block level dynamic operations on the stored data, i.e., block modifications, insertion, deletion, and append, (ii) it safeguards that authorized users (i.e., those who are given privilege to access the owner's file) receive the latest version of the stored data, (iii) it facilitates indirect mutual trust between the owner and the CSP, and (iv) it allows the owner to grant or revoke access to the stored data.

Key words : off-lying data storage, new ness, mutual trust, access control

1 INTRODUCTION

In this digital age, organizations produce a large amount of sensitive data including personal information such as financial data. The local management of such humongous amount of data is problematic and costly due to the requirements of high storage capacity and experienced personnel. Therefore, Storage as a Service offered by cloud service providers emerged as a solution to ease the burden of large local data storage and cut down the maintenance cost by means of outsourcing data storage.

2 LITERATURE REVIEW

Since the data owner physically releases sensitive data to off-lying servers, there are some concerns regarding integrity, privacy, and access control of the data. The privacy feature can be assured by the owner via encrypting the data before storing on off-lying servers.

2.1 Integrity verification of outsourced data

For substantiating data integrity over cloud servers, researchers have proposed unarguable data possession technique to validate the intactness of data stored on remote site. A number of PDP protocols have been presented to efficiently validate the integrity of data, e.g., [1]-[8]. Proof of retrievability [9]-[12] was introduced as a stronger technique than PDP in case that the entire data file can be reconstructed from portions of the data that are reliably stored on the servers.

2.2 Securing outsourced data on off-lying servers

Commonly, traditional access control techniques assume the existence of the data owner and the storage servers in the same trust domain. This assumption no longer holds when the data is outsourced to off- lying server, which takes the full charge of the outsourced data management, and exist outside the trust domain of the data owner. A possible solution can be presented to enable the owner to enforce access control of the data stored on a remote untrusted CSP. Through this solution, the data is encrypted under certain key, which is shared only with the authorized users. The unauthorized users, including the CSP, are unable to access the data since they do not have the decryption key. This general solution has been widely incorporated into existing schemes [13]-[16], which aim at providing data storage security on untrusted remote servers.

Eu-Jin Goh [13] have presented SiRiUS, a secure file system designed to be layered over existing file system such as NFS (Network filie System) to provide end to end security. To enforce access control in SiRiUS, each data file is attached with a metadata file that contains an encrypted key block for each authorized user with some access rights. More specifically, the metadata file represents the data file's access control list. The data file is encrypted using a file encryption key, and each entry in access control list contains an encrypted version of key under the public key of one authorized user.

Kallaha et al. [14] designed a cryptography based file system called Plutus for securing sharing of data on untrusted servers. Some authorized users of the data have privilege to read and write, while others can only read the data.

Different approaches have been investigated that encourage the owner to outsource the data, and offer some sort of assurance related to the privacy, integrity, and access control of the outsourced data. These approaches can prevent and detect malicious actions from the CSP side. On the other hand, the CSP needs to be safeguarded from a misleading owner, who attempts to get illegal compensations by falsely claiming data corruption over cloud servers. This matter, if not properly handled, can cause CSP to go out of business. In this work, a scheme is proposed that addresses important issues related to outsourcing the storage of data, namely dynamic data, newness, mutual trust, and access control. The remotely stored data can be not only accessed by authorized users, but also updated and scaled by the owner. After updating, authorized users should receive the fresh version of the data (newness) i.e., the technique is required to detect whether the received data is stale. Mutual trust between the data owner and CSP is another critical issue, which is addressed in the proposed scheme. A mechanism is introduced to determine the dishonest party, i.e., misbehaviour from any side is detected and the responsible party is identified. Last but not least, the access control is considered, which allows the owner to grant or revoke access rights to the outsourced data. The design and implementation of a cloud-based storage scheme that has the following features: (i) It allows a data owner to outsource the data storage to a CSP, and perform full dynamic operations at the block-level, i.e., it supports operations such as block modification, insertion, deletion, and append. (ii) It ensures the new ness property, i.e., the authorized users receive the most recent version of the outsourced data. (iii) It establishes indirect mutual trust between the data owner and the CSP since each party resides in a different trust domain. (iv) It enforces the access control for the outsourced data.

3 OUR SYSTEM AND ASSUMPTIONS

3.1 System components and relations

The cloud computing storage model considered in this work consists of four main components as illustrated in Fig. 1:

3.1.1 Data owner: A data owner can be an organization or individual generating sensitive data to be stored in the cloud and made available for controlled external use;

3.1.2 Cloud service provider (CSP): A CSP who manages cloud servers and provides paid storage space on its infrastructure to store the owner's files and make them available for authorized users.

3.1.3 Authorized user: A set of owner's clients who have the right to access the remote data

3.1.4 Trusted third party (TTP): An entity who is trusted by all other system components, and has capabilities to detect/specify dishonest parties.

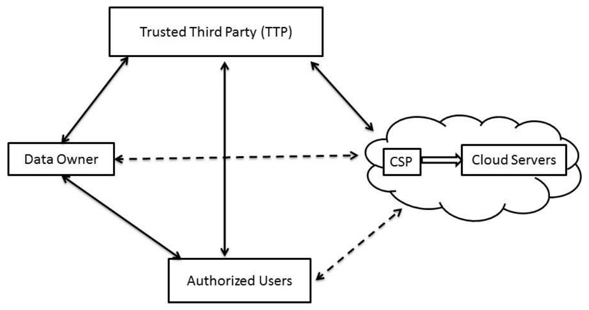


Fig1: Data storage system model.

In Fig. 1, the relations between different system components are represented by double-sided arrows, where solid and dashed arrows represent trust and distrust relations, respectively. For example, the data owner, the authorized users, and the CSP trust the TTP. On the other hand, the data owner and the authorized users have mutual distrust relation with the CSP. Thus, the TTP is used to facilitate indirect mutual trust between these three components. There is a direct trust relation between the data owner and the authorized users.

3.2 Outsourcing, updating, and accessing

For privacy, the owner encrypts the data before sending to off-lying servers. The owner can request with the CSP to perform *block-level* operations on the file. In addition, the owner enforces access control by granting or revoking access rights to the outsourced data. To access the data, the authorized user sends a data-access request to the CSP, and receives the data file in an encrypted form that can be decrypted using a secret key generated by the authorized user.

The TTP is an independent entity, and thus has no incentive to collude with any party. However, any possible leakage of data towards the TTP must be prevented to keep the outsourced data private. The TTP and the CSP are always online, while the owner is *intermittently* online. The authorized users are able to access the data file from the CSP even when the owner is offline.

3.3 Threat model

The CSP is untrusted, and thus the privacy and integrity of data in the cloud may be at risk. For economic incentives and maintaining a reputation, the CSP may hide data loss, or reclaim storage by discarding the data that has not been or is rarely accessed. To save the computational resources, the CPS may totally ignore the data-update requests, or execute just few of them. Hence, the CSP may return damaged or stale data for any access request from the authorized users. Furthermore, the CSP may not honour the access rights created by the owner, and permit unauthorized access for misuse of confidential data.

On the other hand, a data owner and authorized users may collude and falsely accuse the CSP to get a certain amount of reimbursement. They may dishonestly claim that data integrity over cloud servers has been violated, or the CSP has returned a stale file that does not match the most recent modifications issued by the owner.

3.4 Security requirements

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

3.4.2 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.

3.4.3 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.

3.4.4 Access control: Only authorized users are allowed to access the outsourced data. Revoked users can read unmodified data; however, they must not be able to read updated, new blocks.

3.4.5 CSP's defence: The CSP must be safeguarded against false accusations that may be claimed by dishonest owner / user, and such a malicious behaviour is required to be revealed.

4 SYSTEM PRELIMINARIES

4.1 Lazy Revocation

The proposed scheme in this work allows the data owner to revoke the right of some users for accessing the outsourced data. In lazy revocation, it is acceptable for revoked user to read *unmodified* data blocks. However, updated or new blocks must not be accessed by such revoked users. The notation of lazy revocation was introduced in [20]. The idea is that allowing revoked loss in security. This is equivalent to accessing the blocks from cashed copies. Updated or new blocks following a revocation are encrypted under new keys. Lazy revocation trades re-encryption and data access cost for a degree of security. However, it causes fragmentation of encryption keys. i.e., data blocks could have more than one key. Lazy revocation has been incorporated into many cryptographic systems [19], [21], [22].

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.5, Pages : 260 - 266 (2014)
Special Issue of ICACSSE 2014 - Held on October 10, 2014 in St.Ann's College of Engineering & Technology, Chirala, Andhra Pradesh

5 PROPOSED SCHEME

5.1 Warm up Discussion

A straight forward solution to detect cheating from any side is through digital signatures. For a file the owner attaches a signature with each block before outsourcing. The owner sends blocks along with signatures to the CSP, where the signatures are verified. In case signatures failed in verification, the CSP rejects to store the data blocks and asks the owner to re-send the correct signatures. If the signatures are valid, both the blocks and the signatures are stored on the cloud servers. The signatures achieve non repudiation from the owner side. When an authorized user or owner requests to retrieve the file, the CSP sends data file and owner's signatures and adds a CSP signature on file and owner signatures. The user first verifies the tags CSP signature. In case signatures failed in verification, the users ask the CPS to re-perform the transmission process. If CSP signatures are valid the user then verifies the owner's signatures on the blocks. If any signatures of owner are failed in verification, this indicates the corruption of data over cloud servers. The CSP cannot repudiate previously verified and stored by the CSP along with the data blocks. Since the CSP's signatures 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 CSP can replace the new blocks and tags with old versions without being detected (*replay attack*).

If the CSP receives the data block from a trusted entity, the block tags and the signature operations are not needed since the trusted entity has no incentive for repudiation or collusion. Therefore, delegating a small part of the owner's work to the TTP reduces both the storage and computation overheads.

5.2 NOTATIONS

- F is a data file to be outsourced
- h is a cryptographic hash function
- K is a data encryption/ secret key
- Ḟ is an encrypted version of file F
- FH_{TTP} is a hash value for Ḟ, 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

5.3 BLOCK STATUS TABLE

The block status table (BST) is a small dynamic data structure used to reconstruct and access file blocks outsourced to the CSP. The BST consists of three columns.

5.3.1 Serial number (SN)

Indexing to the file blocks. It indicates the physical position of each block in the data file.

5.3.2 Block number (BN)

Counter used to make a logical numbering to the file blocks. The relation between SN and BN can be viewed as a mapping between the logical number BN and physical position SN.

5.3.3 Key version (KV)

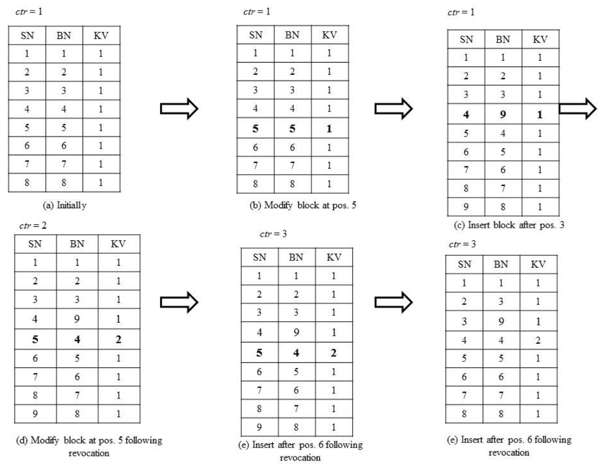
Version of the key that is used to encrypt each block in the data file.

The BST is implemented as a 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 index of the list. Thus each entry contains just two integers BN and KV.

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 or new block is set to be equal to *ctr*.

Fig. 2 shows some examples demonstrating the changes in the VST due to dynamic operations on a data file F. When a file blocks are initially created (Fig. 2a), *ctr* is initialized to 1, serial number and block numbers to the respective index value of the block, and key version to 1. Fig. 2b shows no change for updating the block at position 5 since no revocation is performed. To insert a new block after position 3 in the file F, Fig. 2c shows that a new entry <4,9,1> is inserted in the BST after entry with SN=3, where 4 is physical position of the newly inserted block, 9 is the new logical block number computed by incrementing the maximum of all previous logical block numbers, and 1 is the version of key used to encryption.

A first revocation in the system increments *ctr* by 1 (*ctr* = 2). Modifying the block at position 5 following a revocation (Fig. 2d) results in setting KV to *ctr* at position 5. Thus, the table entry at position 5 becomes <5,4,2>. Fig. 2e shows that a new block is to be inserted after position 6 following a second revocation, which increments *ctr* to be 3. In Fig. 2e, a new table entry <7,10,3> is inserted after entry with SN = 6, where KV 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 corresponding value of SN and shifting all subsequent entries one position up (Fig. 2f). Note that during all dynamic operations, SN indicates the actual physical positions of the data blocks in F.



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

(a) Initially

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

(b) Modify block at pos. 5

ctr = 1

SN	BN	KV
1	1	1
2	2	1
3	3	1
4	**9**	**1**
5	4	1
6	5	1
7	6	1
8	7	1
9	8	1

(c) Insert block after pos. 3

ctr = 2

SN	BN	KV
1	1	1
2	2	1
3	3	1
4	9	1
5	**4**	**2**
6	5	1
7	6	1
8	7	1
9	8	1

(d) Modify block at pos. 5 following revocation

ctr = 3

SN	BN	KV
1	1	1
2	2	1
3	3	1
4	9	1
5	**4**	**2**
6	5	1
7	6	1
8	7	1
9	8	1

(e) Insert after pos. 6 following revocation

ctr = 3

SN	BN	KV
1	1	1
2	3	1
3	9	1
4	4	2
5	5	1
6	6	1
7	7	1
8	8	1

(e) Insert after pos. 6 following revocation

Fig. 2 Changes in the BST due to different dynamic operations on a file F.

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.5, Pages : 260 - 266 (2014)
Special Issue of ICACSSE 2014 - Held on October 10, 2014 in St.Ann's College of Engineering & Technology, Chirala, Andhra Pradesh

5.3 Procedural Steps of the Proposed Scheme

5.3.1 Setup and file preparation

The system setup has two parts: one is done on the owner side, and the other is done on the TTP side.

Owner Role: The data owner starts *ctr* with 1, and generates an initial secret key K_{ctr}. For a file F containing *m* blocks the owner generates a BST with entries SN=BN=*j* (where *j* is block number) and K_j = *ctr*. To achieve privacy-preserving, the owner creates an encrypted file version Ḟ, and sends {Ḟ,BST} to the TTP and deletes the data file from local storage.

TTP Role: To resolve disputes that may arise regarding data integrity and newness, the TTP computes combined hash values for encrypted file Ḟ and the BST. It computes FH_{TTP} and TH_{TTP}, then sends {Ḟ,BST} to the CSP. The TTP keeps only FH_{TTP} and TH_{TTP} on its local storage.

CSP Role: CSP 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 BST and encrypted file Ḟ.

Moreover, the BST is used during each dynamic operation on the outsourced data file, where one table entry is modified or inserted or deleted with each dynamic change on the block level. If the BST is stored only on the CSP side, it needs to be retrieved and validated each time the data owner wants to issue a dynamic request. To avoid such communication and computation overheads, the owner keeps a local copy of the BST, and thus there are two copies of the BST: one is stored on the owner side referred to as BST_o, and the other is stored on the CSP side referred to as BST_c.

TABLE 1: Data stored by each component in the System.

Owner	TTP	CSP
ctr, Kctr,BST_o	FH_{TTP}, TH_{TTP}	Ḟ,BST_o

5.3.2 Dynamic Operations on the Outsourced Data

The dynamic operations in the proposed scheme are performed at the block level via a request. The request contains parameters such as the operation to be performed, and the Entry in the Block Status Table on which the operation need to be performed, key version and hash value of the block need to be modified.

Modification: For a file F, owner wants to modify a block Fig. 3 describes the steps performed by each system component during block modification. The owner uses the technique of one-sender-multiple-receiver to send the modify request to both the CSP and the TTP. TTP updates the combined hash value TH_{TTP} and hash value of encrypted file after modification FH_{TTP}.

Insertion: In a block insertion, the owner wants to insert a new block after index *j* in a file F. The newly constructed file will be F'. Fig. 4 describes the steps performed by each system component during block insertion.

Append: It means adding a new block at the end of the outsourced data. It can be simply done through insert operation after the last block of the data file.

Deletion: When one block is deleted all subsequent blocks are moved one step forward. Fig. 5 describes the steps performed by each system component during block deletion.

5.3.3 Data Access and Cheating Detection

To access outsourced file by the data owner, data user sends a request to both the TTP and the CSP. Fig. 6 shows the verification process performed for the data received from the CSP, and presents how authorized users get access to the outsourced file.

For achieving non-repudiation, the CSP generates two signatures σF and σT for Ḟ and BST_c, respectively. The user receives the encrypted file Ḟ, BST_c, σF and σT from CSP, and FH_{TTP}, TH_{TTP} from the TTP. The authorized user verifies the signatures, and proceeds with the data access procedure only if both signatures are valid.

The authorized user verifies the content of BST_c entries by computing a hash value TH_U and comparing it with the authentic value TH_{TTP} received from TTP. If the user claims that $TH_U \neq TH_{TTP}$, owner is informed and the TTP is invoked to determine the dishonest party.

In case of $TH_U = TH_{TTP}$, the user continues to verify the contents of the encrypted file Ḟ by computing the hash FH_U and compares it with FH_{TTP}. If there is a dispute then the owner is informed and TTP is invoked to determine the dishonest party and resolve such a conflict.

The authorized user to access outsourced data Ḟ, BST_c, are used to determine the keys version of each block. And the user decrypts each block with the key obtained with the help of key version. With the help of BN and BST_c the physical block position SN is determined. And thus the file F is reconstructed in a useable form.

```
/* Modification of a block for the outsourced file */

/* RevFlag is initialized to false */

Data Owner

    1) If the access of one or more users has been revoked then
        a) Increment $ctr$ by 1, and sets $RevFlag$ = true
        b) Copies $KV_j$ from $BST_O$ to $\overline{KV}$ (i.e., $\overline{KV} = KV$)
        c) Sets $KV_j = ctr$ in $BST_O$.
    2) Creates an encrypted block $\tilde{b}_j'$, By encrypting it using key $K_{ctr}$.
    3) Forms a block- modify request $TEntry_{BM} = \{BN_j, KV_j\}$
    4) Sends a modify request
       $\langle BM, TEntry_{BM}, j, \overline{KV_j}, h(\tilde{b}_j), RevFlag, \tilde{b}_j' \rangle$ to both the
       CSP and the TTP, where $h(\tilde{b}_j)$ is the hash of the
       outsourced block to be modified. The $\overline{KV_j}$ is not sent in
       the modify request if $RevFlag$ = false.
    5) The CSP accepts the modify request only if $\{BN_j, \overline{KV_j}\}$
       sent from the owner matches $\{BN_j, KV_j\}$ in $BST_C$, and
       $h(\tilde{b}_j)$ is equal to hash of $\tilde{b}_j$ on the cloud server to
       guarantee that correct values are sent to the TTP.

CSP /* upon accepting the modify request from the owner */

    1) Replaces the block $b_j$ with $b_j'$ in the outsourced file $\dot{F}$
    2) If $RevFlag$ = true then

       Updates the $BST_C$ entry at index $j$ using $TEntry_{BM}$

TTP

    1) Updates $FH_{TTP} = FH_{TTP} FH_{TTP} \oplus h(\tilde{b}_j) \oplus h(\tilde{b}_j')$
    2) If $RevFlag$ = true then
        a) Updates the $TH_{TTP} = TH_{TTP} \oplus h(BN_j || \overline{KV_j}) \oplus$
           $h(BN_j || KV_j)$
```

Fig. 3: Block modification procedure in the proposed scheme.

```
/* inserting of a block $\bar{b}$ after index j in the outsourced file */

/* RevFlag is initialized to false */

Data Owner

    1) If the access of one or more users has been revoked
       then
        a) Increment $ctr$ by 1, and sets $RevFlag$ = true
        b) Copies $KV_j$ from $BST_O$ to $\overline{KV}$ (i.e., $\overline{KV} = KV$)
    2) Construct a new block insert table entry $TEntry_{BI}$ =
       $\{BN_{j+1}, KV_{j+1}\} = \{1 + Max\{BN_j\}_{i \le j \le m}, ctr\}$, and
       insert this entry in $BST_O$ after $j$.
    3) Creates an encrypted block $\tilde{\bar{b}}$ With the encryption key
       $K_{ctr}$.
    4) Sends a request $\langle BI, TEntry_{BI}, j, RevFlag, \tilde{\bar{b}} \rangle$ to both
       the CSP and The TTP.

CSP /* upon receiving the insert request from the owner */

    1) Inserts the block $\tilde{\bar{b}}$ after index $j$ in the outsourced file $\dot{F}$
    2) Inserts the table entry $TEntry_{BI}$ after the index $j$ in the
       $BST_C$

TTP

    1) Updates $FH_{TTP} = FH_{TTP} \oplus h(\tilde{\bar{b}})$
    2) Updates $TH_{TTP} = TH_{TTP} \oplus h(BN_{j+1} || KV_{j+1})$
```

Fig. 4: Block Insertion procedure in the proposed scheme.

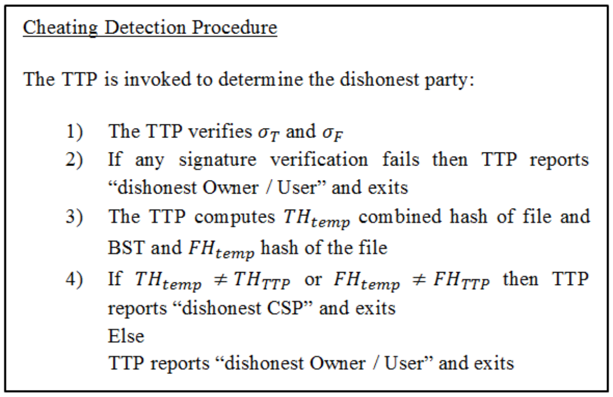
```
/* Deleting of a block $b_j$ from the outsourced file */

Data Owner

    1) Copies entry at index $j$ from $BST_O$ to a block-delete
       table entry $TEntry_{BD} = \{BN_j, KV_j\}$
    2) Delete the entry at index $j$ from $BST_O$
    3) Sends a request $\langle BD, TEntry_{BD}, j, h(\tilde{b}_j) \rangle$ to both the
       CSP and the TTP, where $h(\tilde{b}_j)$ is the hash of the
       outsourced block to be deleted.
    4) The CSP accepts the delete request only if $TEntry_{BD}$
       sent from the owner matches $\{BN_j, KV_j\}$ in $BST_C$ and
       $h(\tilde{b}_j)$ is equal to the hash of the block $b_j$ on the cloud
       server.

CSP /* upon receiving the delete request from the owner */

    1) Deletes the block at index $j$ from the outsourced file $\dot{F}$
    2) Deletes the entry at index $j$ from the $BST_C$

TTP

    1) Updates $FH_{TTP} = FH_{TTP} \oplus h(\tilde{b}_j)$
    2) Updates $TH_{TTP} = TH_{TTP} \oplus h(BN_j || KV_j)$
```

Fig. 5: Block deletion procedure in the proposed scheme.

```
1) An authorized user sends a access request to both the
   CSP and the TTP
2) The CSP responds by sending the outsourced file $\dot{F}$
   associated with a signature $\sigma_F$(CSP's signature on the
   entire file), and sending $BST_C$ associated with a signature
   $\sigma_T$ (CSP's Signature on the entire table) to the authorized
   user.
3) The authorized user verifies $\sigma_F$ and $\sigma_T$, and accepts the
   data only if they are valid signatures.
4) The TTP sends $FH_{TTP}$, $TH_{TTP}$ to the authorized user
5) Verification of the $BST_C$ entries
    a. The user computes $TH_U$ combined hash value
       for received file and BST.
    b. If the user claims that $TH_U \neq TH_{TTP}$ then
       report "integrity violation" to the owner and
       invoke cheating detection procedure (Fig. 7)
6) Verification of the data file $\dot{F}$
    a. The authorized user computes $FH_U$ hash value
       of file received.
    b. If the user claims that $FH_U \neq FH_{TTP}$ then
       report "integrity violation" to the owner and
       invoke cheating detection procedure (Fig. 7)
7) Using $BST_C$ user reconstruct the file F by getting the
   physical position SN of block using BN in $BST_C$ entries.
```

Fig. 6 Data access procedure in the proposed scheme.

ISSN 2278-3091



Cheating Detection Procedure

The TTP is invoked to determine the dishonest party:

1) The TTP verifies σ_T and σ_F
2) If any signature verification fails then TTP reports "dishonest Owner / User" and exits
3) The TTP computes TH_{temp} combined hash of file and BST and FH_{temp} hash of the file
4) If $TH_{temp} \neq TH_{TTP}$ or $FH_{temp} \neq FH_{TTP}$ then TTP reports "dishonest CSP" and exits
   Else
   TTP reports "dishonest Owner / User" and exits

Fig. 7: Cheating detection procedure in the proposed scheme.

6 SECURITY ANALYSES

6.1 Data confidentiality

The outsourced data are kept confidential. The data owner creates an encrypted version of the data file. The encryption of a file is done using a secret key K generated by owner, where it can be accessed by only owner and authorized users.

6.2 Integrity and newness

Integrity and newness properties are preserved using techniques like hashing and lazy revocation. Where hashing techniques preserve the integrity of the data and enables the parties to detect corruption. Lazy revocation will ensure the data newness. It allows the owner to revoke the right of some users for accessing the outsourced data and also allows the revoked users to read unmodified data blocks. However the updated blocks must not be accessed by such users.

6.3 Detection of data corruption

Due to hashing techniques that are followed in this scheme at all the parties the data cannot be corrupted on cloud servers without being detected. During the data access phase of the proposed scheme, the authorized user receives the encrypted file from the CSP and FH_{TTP} from the TTP. The authorized user computes a hash for the received file and compares it with one received form the TTP. If both the computed hash and hash received from the TTP are not matched then file has been corrupted on the server. For violating data integrity without being detected the CSP hast to send a file F' which is not the original file uploaded by the owner but their hashes must match. Due the the one way nature of the hashing technique the CSP cannot generate such a file.

6.4 Enforcement of access control

The owner creates a secret key K only authorized users will know. With which they decrypt the file to read data, and thus the access control is achieved in the proposed scheme.

6.5 Cheating detection of dishonest party

If the owner falsely accuses the CSP regarding data integrity, the TTP performs cheating detection. In this procedure TTP retrieves the encrypted file from the CSP and computes hash and compares with stored hash, if they match then owner/ user is the dishonest party, else CSP is the dishonest party.

7 CONCLUSION

The cloud based storage scheme is proposed that allows owners to benefit from facilities offered by the CSP and facilitates indirect mutual trust between them. The proposed scheme enables data owners to release their concerns regarding confidentiality, integrity, access control of the outsourced data. It enables the authorized users to ensure that they are receiving the most recent version of the outsourced data. To resolve disputes regarding integrity a trusted third party is able to determine the dishonest party. The owner enforces access control by using secret keys and grant or revoke permissions by using lazy revocation techniques.

The security features of the proposed scheme are studied, and showed that the scheme satisfies: (i) Confidentiality on the security underlying encryption, (ii) Detection of data integrity violation based on hashing techniques, (iii) Enforcing access control based on lazy revocation techniques and key sharing to only authorized users so that they can decrypt the outsourced data, and most important (iv) detection of dishonest party through a trusted third party.

REFERENCES

[1] G. Ateniese, R. D. Pietro, L. V. Manchini, and G. Tsudik, "Scalable and efficient provable data possession," in proceedings of the 4th International Conference on Security and Privacy in Comminication Networks, 2008, pp. 1-10

[2] F.Sabe, J.Domingo-Ferrer, A. Martinez-Balleste, Y. Desware, and J.J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," IEEE Trans. On Knowledge and Data Engineering vol. 20, no. 8, 2008

[3] G. Atenises, R.Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D.Song, "Provable data possession at untrusted stores," in proceedings of the 14th ACM conference on computer and Communications security ser. CCS '07, 2007, pp. 598-609.

[4] A. F. Barsoum and M. A. Hasan, "On verifying dynamic multiple data copies over cloud servers," Cryptology ePrint Archive, Report 2011/447, 2011, 2011, http://eprint.iacr.org/.

[5] A. F. Barsoum and M. A. Hasan, "Provable possession and replication of data over cloud servers," Centre For Applied Cryptographic Research, Report 2010/32, 2010, http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-32.pdf.

[6] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: multiple-replica provable data possession," in 28th IEEE ICDCS, 2008, pp. 411–420.

[7] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in Proceedings of the 14th European Conference on Research in Computer Security, 2009, pp. 355–370.

[8] C. Erway, A. K¨upc¸ ¨ u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proceedings of the 16th ACM Conference on Computer and Communications Security, 2009, pp. 213–222.

[9] A. Juels and B. S. Kaliski, "PORs: Proofs of Retrievability for large files," in CCS'07: Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2007, pp. 584–597.

[10] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: a high-availability and integrity layer for cloud storage," in CCS '09: Proceedings of the 16th ACM conference on Computer and communications security. New York, NY, USA: ACM, 2009, pp. 187–198.

[11] H. Shacham and B. Waters, "Compact proofs of retrievability," in ASIACRYPT '08, 2008, pp. 90–107.

[12] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography, 2009.

[13] Eu-Jin Goh, H. Shacham, N. Modadugu, and D. Boneh, "Sirius: Securing remote untrusted storage," in Proceedings of the Network and Distributed System Security Symposium, NDSS, 2003.

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

[15] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," in NDSS, 2005.

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

[19] R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, and L. Zhuang, "Enabling security in cloud storage SLAs with cloudproof," in Proceedings of the 2011 USENIX conference, 2011.

[20] K. E. Fu, "Group sharing and random access in cryptographic storage file systems," Master's thesis, MIT, Tech. Rep., 1999.

[21] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in Proceedings of the 2009 ACM workshop on Cloud computing security, 2009, pp. 55–66.

[22] M. Backes, C. Cachin, and A. Oprea, "Secure key-updating for lazy revocation," in 11th European Symposium on Research in Computer Security, 2006, pp. 327–346.

Vaddepati Venkat Srikanth is a M.Tech Student in the Department of Compute Science and Engineering at St. Ann's College of Engineering and Technology, Chirala. He received B.Tech degree in Computer Science and Engineering from Jawaharlal Nehru Technological University Kakinada, Kakinda in 2012.

Eswar Kodali is presently working as an Associate Professor in Department of Computer Science and Engineering, in St. Ann's College of Engineering and Technology, Chirala. He guided many UG an PG students. He has more than 10 years of excellence in teaching. He published various international journals and presented various papers in several conferences.