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

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st& 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

Enhancing Security of Dynamic Data for Storage Services In Cloud Computing

P. Sathyabama Gayathri, J. Angela Jennifa Sujana, T.Revathi

P.G Scholar, Dept. of Information Technology, Mepco Schlenk Engineering College, Sivakasi, TamilNadu, India.

Assistant Professor (Senior), Dept. of Information Technology, Mepco Schlenk Engineering College, Sivakasi, TamilNadu, India.

Senior Professor & Head, Dept. of Information Technology, Mepco Schlenk Engineering College, Sivaksi, TamilNadu, India.

Abstract - In Cloud Computing, Storage –as –a -Service is one of the most wanted services, but the security of the data stored in the cloud using these services is the key issue. The outsourced data in the cloud has to be guaranteed with confidentiality, integrity and access control. In this work, we device a mechanism of cloud data storage based on indirect mutual trust between the Cloud Service Provider (CSP) and the cloud users through Trusted Third Party Auditor (TTPA). This work facilitates the user to store their data as blocks and enables them to perform dynamic operations on blocks. The stored data can be accessed by a group of users authorized by the data owner. The owner has the privilege to grant or revoke access of the stored data in the cloud. The present system is providing a good security mechanism for stored data and proper sharing of keys among authorized users, and data owner for the cryptographic mechanism.

Key Terms - Mutual trust, access control, dynamic environment, outsourcing data storage

I.INTRODUCTION

Cloud computing[1] is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

In this Information age, several organizations posses huge amount of data which needs to be kept secured. These data includes personal information, health information and financial data. Local maintenance of such Storage as a Service to alleviate the burden of huge local data storage and to reduce the cost by means of outsourcing data storage to the cloud. Since the data owner outsources their sensitive data to the cloud, they want their data to be guaranteed with some security concerns like confidentiality, integrity and proper access control. In some practical applications data confidentiality is not only a security concern but also a juristic issue. For

example in e-Health applications in USA the usage and exposure of data should satisfy the policies confessed by Health Insurance Portability and Accountability Act (HIPAA) [2], thus keeping the privacy of the outsourced data on the cloud is not an option, but it is a demand. Confidentiality can be guaranteed by encrypting the data

before outsourcing it to the remote server. Also the outsourced data should not be modified by unauthorized users. Traditional access control techniques assume that the data owner and the storage servers in the same trust domain. However this assumption no longer holds when the data is outsourced to the cloud storage, which takes full maintenance of the outsourced data, and it, is untrusted by the data owner. To enforce access control data is encrypted with certain key and this key is shared only with the authorized users.

Various schemes are available which supports the data owner to outsource their sensitive data to the untrusted cloud storage by giving assurance related to the confidentiality, integrity and access control. These schemes thwart and identify malicious actions from the CSP side. Conversely the CSP needs to be protected from the dishonest owner or user, who tries to get unlawful compensations by untruly claiming data modification over

CSP. If this concern is not appropriately handled, this may lead the CSP to go out of business one day [3].

In this work, we proposed a technique which addresses some important concerns associated with outsourcing sensitive data to the untrusted remote CSP, namely dynamic data, newness, mutual trust and access control. The outsourced data can be modified and scaled by the data owner. After doing modification, the authorized users are enabled to get the most recent version (newness property) of the outsourced data. A technique is required to identify the staleness of the received data. This issue is dangerous for applications in which critical decisions are made based on the received data. Mutual trust between the data owner and CSP is enabled in the proposed scheme. A method is established to resolute dishonest party from any side. Finally, the access control is considered, which allows the data owner to grant or revoke access rights to the outsourced data.

II.Related Work

Existing work related to our proposed work can be found in the areas of integrity verification of remotely stored data and file encryption schemes in distributed systems and access control mechanisms over outsourced data. Ateniese et al. [4] designed a model based on PDP (Provable Data Possession) protocol which allows a client to verify the server's data possession. In this scheme the client preprocesses the file and generates meta-data, stores it locally, and then outsource the file to the server. The server stores the file and starts respond to challenges issued by the client. Integrity verification is done through batch verification of homomorphic hash functions.

Curtmola et al. [5] designed a model based on MR-PDP which uses replication in order to improve data availability and reliability. By storing multiple copies, if some copies are destroyed still the data can be recovered from the remaining copies. But challenges incur relatively more cost in MR-PDP.

Dodis et al. [6] presented a model based on POR (*Proofs of Retrievability*) in which the client stores a file F on a server and keeps only a short private verification string locally. Later, the client can run an audit protocol to verify the server's data possession, in which the client acts as a verifier and the server proves that it possesses the data. POR is a complementary approach to PDP, and is stronger than PDP in the way that it can be reconstructed from the portions of the data which are reliably stored on the remote server.

Kallahalla et al. [7] presented a cryptographic based file system called Plutus: Scalable secure file sharing on untrusted storage, which enforces access control over outsourced data. In which a file is divided into blocks and each block is encrypted with File-block key and each File-block key is encrypted with File- lockbox key. If the data owner wants to share the file with his clients he just distributes the File- lockbox key to them.

Goh et al. [8] presented SiRiUs, which enforces access control over outsourced data. In this scheme each d-file(data file) is attached with a md-file(meta data file). The md-file contains an encrypted key block for each authorized users with some access right, more precisely the md-file contains d-file's access control list. The d-file is encrypted with FEK and FEK is further encrypted under the public key of each authorized user.

Green et al. [9] presented improved proxy re encryption scheme, in which a semi trusted proxy computes a function that converts ciphertext for Alice into ciphertext for Bob without knowing the underlying plaintext.

III.Proposed System

Our proposed work addresses some important concerns regarding outsourcing data storage to the remote untrusted storage, such as dynamic data, mutual trust, access control and newness. In our proposed work the owner is allowed to do data modifications on the outsourced data. To validate the newness property of the outsourced data, it requires some metadata which mirror the latest modifications on the outsourced data issued by the data owner. However the block indices must have the awareness that the CSP has modified the blocks at the requested position. At this end, the proposed scheme uses combined hash values and a small data structure called Block Status Table (BST). The TTPA (Trusted Third Party) establishes mutual trust between data owner, CSP and authorized users in an indirect way. To enforce access control the proposed scheme uses three cryptographic functions, namely BrdEnc (Broadcast Encryption), Key Rotation and Lazy Revocation. The BrdEnc allows the data owner to encrypt some confidential information to only authorized users allowing them to access the outsourced data. Lazy revocation enables the revoked users to access the older version of the outsourced data i.e. only the authorized users are allowed to access the most recent version of the outsourced data. Using key rotation authorized users can access both latest version of the data and older version of the data.

Block Status Table

The block status table is a small data structure used to access and restructure the received file blocks. BST will contain three columns *SN, BN,* and *KV*. *SN* is a serial number which indicates physical positioning of the file blocks. *BN* indicates the block number of the file blocks. *KV* indicates the Key Version under which the file block is encrypted. Table 1-3 shows the example BST structure for a file with 8 blocks.

Initially the ctr is initialized to 1 as in Table I. The KV is set to ctr. Table II indicates the BST entries for the deletion of block at position = 5 while there is no revocation of users. Hence, the ctr remains unchanged. But in Table III the ctr is incremented by 1 i.e., ctr=2 since, there is an revocation. Hence, insertion of new block following revocation is encrypted under KV =2.

TABLE I
Initial BST

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 II
BST after Deletion Pos=5

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

TABLE III
BST after insertion at pos=4 following Revocation

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

A. *Our System Model*

Cloud storage model considered in our proposed work has four main components as depicted in Fig.1

i. A *data owner* can be an organization, which generates sensitive data that is to be outsourced to the cloud and made available for only authorized users.

ii. A *Trusted Third Party Auditor (TTPA)* [17] who is trusted by all other components and has the capability to detect the dishonest party.

iii. A *CSP* who manages cloud services and provides paid storage service on its infrastructure to the data owner, where he outsources the file and makes them available for authorized users.

iv. *Authorized users* – a set of owner's clients who have the right to access the outsourced file.

Our cloud storage system model can be adopted by many practical applications. For example, Educational applications can be visualized by our model as in Fig.1, where the student's database that contains large and sensitive information can be stored on cloud servers. In this type of application, an institution can be considered as a data owner, the teaching staffs can be considered as the authorized users, who has given the access rights over the outsourced student's information, and an independent organization can be considered as the TTPA. Likewise more practical applications can be envisioned in similar settings. The auditing process of the data received from the CSP is done by authorized users. We used TTPA only to solve disputes that may arise due to data integrity and newness verification.

B. *Outsourcing, updating and accessing*

The data owner has a file F, which is divided into m blocks and is to be outsourced to CSP, who will provide paid storage space to the data owner. Before outsourcing the file to the cloud server, to achieve confidentiality the owner encrypts the file blocks. After doing so, the owner can interact with the CSP to do full block-level dynamic operations on the file. These block-level operations include insert, delete, append, and modify certain blocks of the outsourced file. For time being, we have considered only insert and delete operations in our work. An authorized user receives the encrypted file, by sending the data access request to the CSP. The encrypted file can be decrypted using a secret key which can be generated by the authorized user.

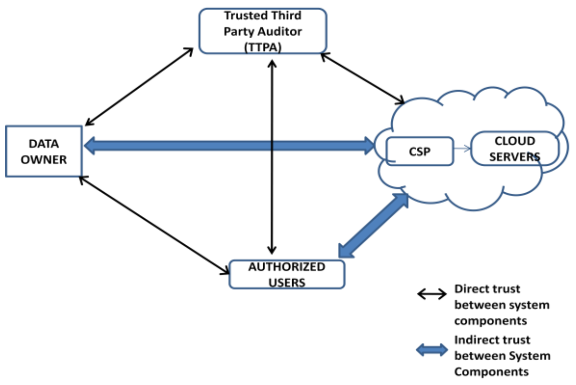
Trusted Third Party Auditor (TTPA)
DATA OWNER
CSP
CLOUD SERVERS
AUTHORIZED USERS
Direct trust between system components
Indirect trust between System Components

Fig.1 Cloud Storage System Model

We imagine that, the verification of the authorized users' identity has already been done with the data owner; hence we haven't considered this in our work. And also all authorized users have the same access privilege over the outsourced data.

The TTPA is an autonomous entity, and thus has no motivation to collude with any party in the system. The TTPA and the CSP are always online, while the data owner can be online or offline. Even though the owner is in offline, the authorized users can access the outsourced data from the CSP.

C. *Access control mechanism*

The three cryptographic techniques Lazy Revocation, Key Rotation and Broadcast Encryption which are discussed below are combined to enforce access control over outsourced data.

i. Lazy Revocation

The data owner in our proposed work is allowed to revoke access right of some users from accessing the outsourced data at any time. The revoked users are allowed to access unmodified blocks in Lazy Revocation. However, modified or new blocks must not be accessed by such revoked users. This is equivalent to accessing the file blocks from caches. The idea behind this scheme is, modified or new blocks following revocation are encrypted under new key. Thus each data block may have more than one key. Lazy Revocation trades re-encryption cost. Lazy Revocation has been used in many cryptographic schemes [10], [11], [12]

ii. Key Rotation

In this technique [7], a sequence of keys can be
be
generated from an initial key and a master secret key. The sequence of keys has two main characteristics

i. The next key in the sequence can only be generated by the owner of the master secret key.

ii. The authorized users knowing the key in the sequence can able to generate previous keys in the sequence. i.e. given the ith key key$_i$ in the sequence, the authorized users can compute the previous keys in the sequence

{ Key$_j$ } where j < i, but it is infeasible to compute { Key$_j$ }, where j > i without having the master secret key.

Property i. allows the data owner to revoke the access right over outsourced data

Property ii. Allows the authorized users to maintain access to the file blocks

Let $N = pq$ denote a RSA modulus (p & q are prime numbers), a public key $= (N, e)$and a master secret key d. The key d is known only to the data owner, and $ed \equiv 1\, mod(p-1)(q-1)$ [14].Whenever a user's access is revoked the key is rotated forward to generate new key in the sequence as

$$Key_{ctr+1} = Key_{ctr}^{d}\ mod\ N \qquad (1)$$

The authorized users can recursively generate older versions of the key (backward rotation) as

$$Key_{ctr-1} = Key_{ctr}^{e}\ mod\ N \qquad (2)$$

iii. Broadcast Encryption

Broadcast Encryption (BrdEnc) scheme [13], [14] allows a broadcaster to encrypt a message for a group of users. The users in the group can only able to decrypt the message. However, the users outside the group collude they could not decrypt the message. In our work, we use BrdEnc to enforce access control over outsourced data. This scheme is a combination of three algorithms

a) Setup b) Encrypt c) Decrypt

These algorithms are explained in [15].

IV.Implementation Procedure

The implementation procedure of the proposed system is discussed in this section. This section explains about algorithms used for dynamic block level operations over outsourced data. Also this section explains the algorithms used for data access and cheating detection procedure.

Procedural Steps

a) File Preparation

File preparation may contain two parts. one is owner's part and another one is TTPA's part.

Owner's part

In owner's part the file F, which is to be outsourced is divided into q blocks i.e. $F = \{b_i\}_{1\le i\le q}$. The owner initializes the *ctr* as 1 and generates the initial secret key Key_{ctr}/Key_1. Key_{ctr} can be rotated forward following user revocation and also it can be rotated backward to enable the authorized users to gain access right on the outsourced blocks, which are encrypted under older versions of Key_{ctr}.

For a file $= \{b_i\}_{1\le i\le q}$, the owner generates the BST with the values $SN_i = BN_i = i$ and $KV_i = ctr$. To achieve privacy, the owner encrypts the file F and generates an encrypted file $\hat{F} = \{\hat{b}_i\}_{1\le i\le q}$. Where $\hat{b}_i = E_{DEK}(BN_i \parallel b_i)$ and DEK = h(Key_{ctr}) . Furthermore, the owner creates a rotator R = (*ctr*, BrdEnc(Key_{ctr})) . Where, BrdEnc allows only the authorized users to decrypt Key_{ctr} and access the file blocks. The owner sends the $\{\hat{F}, BST, R\}$ to the TTPA and deletes the file from its local storage. Embedding BN_i , with b_i helps in reconstructing the blocks in correct sequence.

TTPA's part

A small part of owners' work is delegated to TTPA in order to reduce the storage and computation overhead on the owner side. The TTPA will compute combined hash values of the encrypted file $\hat{F}$ and BST and keep them locally to resolve disputes that may arise due to integrity/newness violation. The TTPA computes

$$F_{HTTPA} = \oplus_{i=1}^{q} h(\hat{b}_i) \qquad (3)$$

$$T_{HTTPA} = \oplus_{i=1}^{q} h(BN_i \parallel KV_i) \qquad (4)$$

The TTPA then sends the $\{\hat{F}, BST\}$ to the CSP and keeps only F_{HTTPA} and T_{HTTPA} on its local storage.

b) *Dynamic Operations on the Outsourced Data*

The request for dynamic block level operations is in the general form as follows $(BlockOp, TEntry_{BlockOp}, i, KV_i, h(\hat{b}_i), RevFlag, b^{\star})$ where $BlockOp$ corresponds to block deletion (denoted as BD) and block insertion (denoted as BI). And $TEntry_{BlockOp}$ denotes an entry in BST_{own} corresponds to dynamic request issued by the owner. The parameter i is the block index at which dynamic operation is to be performed, KV_i is the value of the key version in BST_{own} at index i before doing requested dynamic operation. $h(\hat{b}_i)$ is the hash of the block at index i before insertion/deletion. $RevFlag$ is a one bit flag initialized to false. And it is set to true following a user revocation. $b^{\star}$ is the new block value.

Insertion

In the block insertion request the owner wants to insert a new block $\bar{b}$ after index i in the file F = $\{b_1, b_2, \dots \dots b_q\}$ thus the newly con constructed file $F = F' = \{b_1, b_2, \dots b_i, \bar{b}, \dots, b_{q+1}\}$.

The below algorithm [16] describes the steps performed by data owner, CSP and TTPA for the block insertion operation.

Deletion

Block deletion [16] is just opposite to the block insertion operation. The blocks are one level moved forward following deletion of a block at index i. The step $F_{HTTPA}\ F_{HTTPA} \oplus h(\hat{b}_j)$ is used to delete $\hat{b}_j$ from F_{HTTPA} (properties of XOR).

c) *Data Access and Cheating Detection*

An authorized user sends a data access request to both CSP and TTPA. The CSP sends *{* $\hat{F}$, BST_C ,Sig_F, Sig_T*}* to the authorized user. The TTPA sends {FH_{TTPA} , TH_{TTPA} , Rot} to the authorized user. The users' verification on these entries and data access procedure is explained in algorithm3 [16]. And the cheating detection procedure is discussed in algorithm4 [16].

```
**Algorithm1:** Block Insertion

**Data Owner**

1)  **If** one or more users' access is revoked **then**
    a)  Using forward key rotation Rolls $Key_{ctr}$ forward
    b)  Increments $ctr$ by 1 and changes $RevFlag$ to true
    c)  Generates rotator $R = (ctr, BrdEnc(Key_{ctr}))$
    d)  Sends rotator $R$ to the TTPA
2)  Creates a block-modify table entry $TEntry_{BI} = \{BN_{i+1}, KV_{i+1}\} = \{1 + Max\{BN_i\}_{1 \le i \le q}, ctr\}$ and inserts this entry in $BST_{Own}$ at index $i$
3)  Generates an encrypted block $\bar{\bar{b}} = E_{DEK}(BN_i \parallel \bar{b}_i)$ where DEK = $h(Key_{ctr})$
4)  Sends a block-modify request $(BI, TEntry_{BI}, i, null, null, RevFlag, \bar{\bar{b}})$ to both the CSP and the TTPA(OSMR transmission)

**CSP** /* upon receiving the insert request from the owner */
    1)  Inserts the block $\bar{\bar{b}}$ after index $i$ in the outsourced file $\hat{F}$
    **2)**  Inserts the table entry at index $i$ of $BST_{CSP}$ using $TEntry_{BI}$ components

**TTPA**
    1)  Updates the combined hash value of the file blocks as $F_{HTTPA} = F_{HTTPA} \oplus h(\bar{\bar{b}})$
    2)  Updates the combined hash value of the BST as $T_{HTTPA} = T_{HTTPA} \oplus h(BN_i \parallel \overline{KV_i}) \oplus h(BN_i \parallel KV_i)$
    3)  **If** RevFlag = true **then**
 Updates the previously stored rotator R with the newly received value
```

```
**Algorithm2:** Block Deletion
    **Data Owner**
    1)  Copies the entry at index $i$ from $BST_{Own}$ to a block delete table entry $TEntry_{BD} = \{BN_i, KV_i\}$
    2)  Deletes this entry in $BST_{Own}$ at index $i$
    **3)**  Sends a request $(BD, TEntry_{BD}, i, null, h(\hat{b}_i), false, null)$ to both
            a.  the CSP and the TTPA(OSMR) where $h(\hat{b}_i)$ is the hash
            b.  of the outsourced block to be deleted
    4)  The CSP accepts the delete request only if $TEntry_{BD}$ sent from the owner matches $\{BN_i, KV_i\}$ in $BST_{CSP}$ and $h(\hat{b}_i)$ is equal to the hash of the block block $\hat{b}_i$ on the cloud server
```

```
    **CSP** /* upon receiving the delete request from the owner */
            1)  Deletes the block $\hat{b}_i$ after index $i$ in the outsourced file $\hat{F}$
            **2)**  Deletes the table entry at index $i$ of $BST_{CSP}$ using $TEntry_{BI}$ components

    **TTPA**
            1)  Updates $F_{HTTPA} = F_{HTTPA} \oplus h(\hat{b}_i)$
            2)  Updates $T_{HTTPA} = T_{HTTPA} \oplus h(BN_i \parallel KV_i)$
```

```
**Algorithm3:** Data Access Procedure

    1)  An authorized user sends a data-access request to both the CSP and the TTPA
    2)  The CSP responds by sending the outsourced file $\hat{F} = \{\hat{b}_i\}_{1 \le i \le q}$ associated with a signature $Sig_F$ (CSP's signature on the entire file), and sending $BST_{CSP}$ associated with $Sig_T$ (CSP's signature on the entire table) to the authorized user
    3)  The authorized user verifies $Sig_F$ and $Sig_T$, and accepts the data only if $\sigma_F$ and $\sigma_T$ are valid signatures
    4)  The TTPA sends $F_{HTTPA}$, $T_{HTTPA}$, and $R = (ctr, bENC(Key_{ctr}))$ to the authorized user
    5)  The TTPA sends $F_{HTTPA}$, $T_{HTTPA}$, and $R = (ctr, bENC(Key_{ctr}))$ to the authorized user
    6)  Verification of the $BST_{CSP}$ entries
        a.  The user computes $TH_U = \oplus_{i=1}^{q} h(BN_i \parallel KV_i)$
        b.  If the user claims that $TH_U \ne T_{HTTPA}$ then report "integrity violation" to the owner and invoke cheating detection procedure
    7)  Verification of the data file $\hat{F}$
            a)  The authorized user computes $FH_U = \oplus_{i=1}^{q} h(\hat{b}_i)$
            b)  If the user claims that the computed value $FH_U \ne FH_{TTP}$ then report "integrity violation" to the owner and invoke cheating detection procedure
    8)  Data access
            a)  The authorized user gets $Key_{ctr}$ by decrypting BrdEnc($Key_{ctr}$) part in $R$
            b)  for i=1 to m do
    /* rotate backward the current $Key_{ctr}$ to the version that is used to decrypt the block $\hat{b}_j$ */
    -   Set $K_i = Key_{ctr}$
    -   For i=1 to $ctr - KV_i$ do
$Key_i = Key_i^e \bmod N$
    end for
    -   $(BN_i \parallel b_i) = E_{DEK}^{-1}(\hat{b}_i)$ where $DEK = h(K_i)$
    -   Get the physical position $SN_i$ of $b_i$ using $BN_i$ and $BST_{CSP}$
    -   The authorized user places $b_i$ in the correct order of the decrypted file F
```

Algorithm4: Cheating Detection

```
The TTPA is invoked to determine the dishonest party:
    1) The TTPA verifies $Sig_T$ and $Sig_F$
    2) If any signature verification fails then
    3) The TTPA computes $TH_{temp} = \oplus_{i=1}^{q} h(BN_i \parallel KV_i)$ and $FH_{temp} = \oplus_{i=1}^{m} h(\hat{b}_i)$
    4) If $TH_{temp} \neq TH_{TTPA}$ or $FH_{temp} \neq FH_{TTPA}$ then
        TTPA reports "dishonest CSP" and exits
        /* data is corrupted */
    else
        TTPA reports "dishonest owner/user" and exits
```

cloudsim. Our implementation consists of four modules: owner module, CSP module, TTPA module and user module. For BrdEnc algorithm we have implemented using jpair library. To implement this algorithm we use an elliptic curve with a 256 bit group order. And we have used MD-5, SHA-256 for hashing, and digital signature algorithms.

V. Performance Analysis

We evaluate the performance of the proposed scheme by analyzing storage and computation overhead. The data file we have used for our experiments is of size 10GB with block size of 100MB.

Storage overhead. This is the additional storage space required to store necessary information other than the outsourced file F. An entry of BST at the owner side is of 8bytes, and the no of entries will be equal to number of blocks q of the file F. Likewise, at the CSP side the additional storage of BST requires 8q bytes, where q is the number of blocks. Each may require 800 MB storage.

TABLE IV Storage Overhead

Data Owner	Authorized Users	CSP	TTP
8q	-	8q	-

Computation overhead The computation cost for encrypting the data before outsourcing, and the dynamic operations require hash function, encryption, BrdEnc and it may require *FR* forward rotation if there is a revocation. To reflect latest version of the outsourced data, the TTPA updates the combined hash values for the file F and BST. Therefore the computation overhead on the TTPA side is *4h*. On accessing data from CSP the authorized user has to verify the two signatures generated by CSP on F and BST, and verifies the data file and BST_{CSP} entries. Hence, the computation overhead on the authorized user side is $2V_\sigma + 3m\,h + BrdEnc^{-1} + [BR]$, and $2S_\sigma$ on the CSP side. In case of disputes regarding integrity violation, The TTPA verifies the two signatures generated by CSP, if signatures are valid then TTPA computes temporary combined hash values on the File F and BST, hence the computation overhead on TTPA is

Operations	Owner	CSP	TTP	USERS
Dynamic Operations	$h + E_{DEK} + [FR + BrdEnc]$	-	2h+[2h]	-
Data Access	-	$2S_\sigma$	-	$2V_\sigma + 3mh + BrdEnc^{-1} + [BR]$
Cheating Detection	-	-	-	$2V_\sigma + [2mh]$

$2V_\sigma + 2m\,h$.

TABLE V Computation Overhead

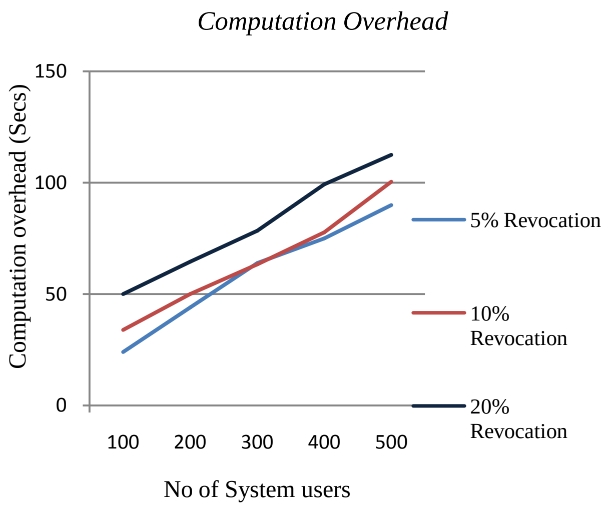

Fig 2. Owners' avg computation overhead

VI. Conclusion

In this project, we have envisaged a cloud-based storage scheme which 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. The proposed scheme enables the authorized users to ensure that they are receiving the most recent version of the outsourced data. Moreover, in case of dispute regarding data integrity/newness, a TTPA is able to determine the dishonest party. The data owner enforces access control for the outsourced data by combining three cryptographic techniques: broadcast encryption, lazy revocation, and key rotation. The experimental results show that the proposed scheme is a robust model in terms of security.

References

[1] 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

[2] 104th United States Congress, "Health Insurance Portability and Accountability Act of 1996 (HIPAA)," Online at http://aspe.hhs.gov/admnsimp/pl104191.htm, 1996.

[3] 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 on USENIX annual technical conference, ser. USENIXATC’11. USENIX Association, 2011.

[4] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable data possession at untrusted stores,” in CCS ’07: Proceedings of the 14th ACM Conference on Computer and Communications Security, New York, NY, USA, 2007, pp. 598–609.

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

[6] Y. Dodis, S. Vadhan, and D. Wichs, “Proofs of retrievability via hardness amplification,” in TCC ’09: Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 109–127.

[7] 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 Conference on File and Storage Technologies. USENIX, 2003.

[8] E.-J. Goh, H. Shacham, N. Modadugu, and D. Boneh, “Sirius: Securing remote untrusted storage,” in Proceedings of the Network and Distributed System Security Symposium, NDSS. The Internet Society, 2003.

[9] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, “Improved proxy re-encryption schemes with applications to secure distributed storage,” in Proceedings of the Network and Distributed System Security Symposium, NDSS. The Internet Society, 2005.

[10] K. E. Fu, “Group sharing and random access in cryptographic storage file systems,” Master’s thesis, MIT, Tech. Rep., 1999.

[11] M. Backes, C. Cachin, and A. Oprea, “Secure key-updating for lazy revocation,” in Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, ser. Lecture Notes in Computer Science. Springer, 2006, pp. 327–346.

[12] Riedel, M. Kallahalla, and R. Swaminathan, “A framework for evaluating storage system security,” in Proceedings of the 1st USENIX Conference on File and Storage Technologies, ser. FAST ’02. USENIX Association, 2002.

[13] Boneh, C. Gentry, and B. Waters, “Collusion resistant broadcast encryption with short ciphertexts and private keys,” in Advances in Cryptology - CRYPTO, 2005, pp. 258–275.

[14] A.Fiat and M. Naor, “Broadcast encryption,” in Proceedings of the 13th annual international cryptology conference on Advances in cryptology. Springer-Verlag New York, Inc., 1994, pp. 480–491.

[15] B.Menezes, “An introduction to pairing-based cryptography,” Lecture Notes 2005, Online at http://www.math.uwaterloo.ca/_ajmeneze/publications/pairings.pdf

[16] A. F. Barsoum and M. A. Hasan,”Enabling Dynamic Data and Indirect Mutual Trust for Cloud Computing Storage Systems” IEEE Transactions on Parallel and Distributed Systems, 2012.

[17] Angela Jennifa Sujana J and Revathi T, “Ensuring Privacy in Data Storage as a Service for Educational Institution in Cloud Computing” International Symposium on CLOUD and SERVICES COMPUTING - ISCOS December-17th-19th, 2012.