ISSN 2250-3153

Achieving Mutual Trust and Empowering Dynamic Data in Cloud Storage

S.Sandhiya, D.Saranya, S.Archana, M.Jayasudha

University College of Engineering Villupuram, India, Department of Information Technology

Abstract- The management of vast amount of data is quite expensive due to the requirements of high storage capacity and qualified personnel. Storage-as-a-Service (SaaS) offered by cloud service providers (CSPs) is a paid facility that enables organizations to outsource their data to be stored on remote servers. Thus, SaaS reduces the maintenance cost and mitigates the burden of large local data storage at the organization's end. A data owner pays for a desired level of security and must get some compensation in case of any misbehavior committed by the CSP. On the other hand, the CSP needs a security from any false accusations that may be claimed by the owner to get dishonest compensations. In this paper, a cloud-based storage scheme is proposed that allows the data owner to benefit from the facilities offered by the CSP and enables indirect mutual trust between them. The proposed scheme has four important features: (i) it allows the owner to outsource sensitive data to a CSP, and perform full block-level dynamic operations on the outsourced data, i.e., block modification, insertion, deletion, and append, (ii) it ensures that authorized users (i.e., those who have the right to access the owner's file) receive the latest version of the outsourced data, (iii) it enables indirect mutual trust between the owner and the CSP, and (iv) it allows the owner to grant or revoke access to the outsourced data. We discuss the security issues of the proposed scheme.

Index Terms- Storage-as-a-Service mutual trust, access control, Cloud storage, Data outsourcing

I. INTRODUCTION

Cloud computing is based on the fundamental principle of reusability of IT capabilities. It is a large-scale distributed computing paradigm in which a pool of computing resources is available to users (cloud consumers) via the Internet Computing resources, e.g., processing power, storage, software, and network bandwidth, are represented to cloud consumers as the accessible public utility services. Distributed processing, parallel processing and grid computing together emerged as cloud computing. Cloud computing is a distributed computational model over a large pool of shared-virtualized computing resources (e.g., storage, processing power, memory, applications, services, and network bandwidth). Cloud service providers (CSPs) offer different classes of services (Storage-as-a-Service (SaaS), Application-as-a-Service, and Platform-as-a-Service) that allow organizations to concentrate on their core business and leave the IT operations to experts. The users can access the stored data at any time by using Application Programming Interface (API) provided by cloud providers through any terminal equipment connected to the internet. Though cloud computing is targeted to provide better utilization of resources using virtualization techniques and to take up much of the work load from the client, it is fraught with security risks. Cloud computing is the long dreamed vision of different organizations produce a large amount of sensitive data including personal information, electronic health records, and financial data. While there is an observable drop in the cost of storagehardware, the management of storage has become more complex and represents approximately 75% of the total ownership cost SaaS offered by CSPs is an emerging solution to mitigate the burden of large local data storage and reduce the maintenance cost via the concept of outsourcing data storage.

Since the owner physically releases sensitive data to a remote CSP, there are some concerns regarding confidentiality, integrity, and access control of the data. For example, in e-Health applications inside the USA the usage and disclosure of protected health information should meet the policies admitted by Health Insurance Portability and Accountability Act (HIPAA) , and thus keeping the data private on the remote storage servers is not just an option, but a demand. The confidentiality feature can be assured by the owner via encrypting the data before outsourcing to remote servers. The proposed model provides trusted computing environment by addressing important issues related to outsourcing the storage of data, namely confidentiality, integrity, access control and mutual trust between the data owner and the CSP. This means that the remotely stored data should be accessed only by authorized users (i.e., those who have the right to access the owner's file) and should remain confidential. The CSP needs to be safeguarded from any false accusation that may be claimed by a data owner to get illegal compensations.

The access control techniques assume the existence of the data owner and the storageservers in the same trust domain. This assumption, however, no longer holds when the data is outsourced to a remote CSP, which takes the full charge of the outsourced data management, and resides outside the trust domain of the data owner. A feasible 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 a certain key, which is shared only with the authorized users. The unauthorized users, including the CSP, are not capable to access the data since they do not have the decryption key. This general solution has been widely incorporated into existing schemes], which aim at providing data storage security on untrusted remote servers. Another class of solutions utilizes attribute-based encryption (ABE) to achieve fine-grained access control. ABE is a public key cryptosystem for one-to-many communications that enables fine-grained sharing of encrypted data. The ABE associates the ciphertext with a set of attributes, and the private key with an access

structure (policy). The ciphertext is decrypted if and only if the associated attributes satisfy the access structure of the private key. Different approaches have been investigated that encourage the owner to outsource the data, and offer some sort of guarantee related to the confidentiality, integrity, and access control of the outsourced data. These approaches can prevent and detect (with high probability) malicious actions from the CSP side. On the other hand, the CSP needs to be safeguarded from a dishonest owner, who attempts to get illegal compensations by falsely claiming data corruption over cloud servers. This concern, if not properly handled, can cause the CSP to go out of business.

In this work, we propose a scheme that addresses some important issues related to outsourcing the storage of data, namely data dynamic, newness, mutual trust, and access control. One of the core design principles of data outsourcing is to provide dynamic scalability of data for various applications. This means that the remotely stored data can be not only accessed by authorized users, but also updated and scaled by the owner. After updating, the authorized users should receive the latest version of the data (newness property), i.e., a technique is required to detect whether the received data is stale. This issue is crucial for applications in which critical decisions are taken based on the received data. For example, in e-Health applications a physician may write a prescription based on a patient's medical history received from remote servers. If such medical data is not up-to-date, the given prescription may conflict with the patient's current circumstances causing severe health problems. Mutual trust between the data owner and the CSP is another imperative issue, which is addressed in the proposed scheme. A mechanism is introduced to determine the dishonest party, i.e., misbehavior from any side is detected and the responsible party is identified. Last but not least, the access control is considered, which allowsthe data owner to grant or revoke access rights to the outsourced data.

II. MAIN CONTRIBUTIONS

Our contributions can be summarized in two main points.

1) The design and implementation of a cloud-based storage scheme that has the following features:

- It allows a data owner to outsource the data to a remote CSP, and perform full dynamic operations at the block-level, i.e., it supports operations such as block modification, insertion, deletion, and append
- It ensures the newness property, i.e., the authorized users receive the most recent version of the data
- It establishes indirect mutual trust between the data owner and the CSP since each party resides in a different trust domain
- It enforces the access control for the outsourced data

2) We discuss the security features of the proposed scheme. Besides, we justify its performance through theoretical analysis and experimental evaluation of storage, communication, and computation overheads.

2. LITERATURE REVIEW

Existing research close to our work can be found in the areas of reliability verification of outsourced data, access control of outsourced data, and cryptographic file systems in distributed networks.

In 1999, sales- force.com was established by Parker Harris, Marc Benioff. They applied many technologies of consumer web sites like Google and Yahoo! to business applications. They also provided the concept's like ''On demand'' and ''SaaS'' with their real business and successful customers. Cloud data storage (Storage as a Service) is an important service of cloud computing referred as Infrastructure as a Service (IaaS). Amazon's Elastic Compute Cloud (EC2) and Amazon Simple Storage Service(S3) are well known examples of cloud data storage. Cloud users are mostly worried about the security and reliability of their data in the cloud. Amazon's S3 is such a good example.

Kallahalla et al designed a cryptography-based file system called Plutus for secure sharing of data on untrusted servers. Some authorized users of the data have the privilege to read and write, while others can only read the data. In Plutus, a file-group represents a set of files with similar attributes, and each file-group is associated with a symmetric key called file-lockbox key. A data file is fragmented into blocks, where each block is encrypted with a unique symmetric key called a file-block key. The file-block key is further encrypted with the file-lockbox key of the file-group to which the data file belongs. If the data owner wants to share a file-group with a set of users, the file-lockbox key is just distributed to them. Plutus supports two operations on the file blocks: read and write/modify. Delete operation can be supported by overwriting an existing block with null.

Goh et al. have presented SiRiUS, which is designed to be layered over existing file systems such as NFS (network file system) to provide end-to-end security. To enforce access control in SiRiUS, each data file (d-file) is attached with a metadata file (md-file) that contains an encrypted key block for each authorized user with some access rights (read or write). More specifically, the md-file represents the d-file's access control list (ACL). The d-file is encrypted using a file encryption key (FEK), and each entry in the ACL contains an encrypted version of the FEK under the public key of one authorized user. For large-scale sharing, the authors in presented SiRiUS-NNL that uses NNL (Naor-Naor-Lotspiech) broadcast encryption algorithm to encrypt the FEK of each file instead of encrypting using each authorized user's public key. SiRiUS supports two operations on the file blocks: read and write/modify.

Based on proxy re-encryption Ateniese et al. have introduced a secure distributed storageprotocol. In their protocol, a data owner encrypts the blocks with symmetric data keys, which are encrypted using a master public key. The data owner keeps a master private key to decrypt the symmetric data keys. Using the master private key and the authorized user's public key, the owner generates proxy re-encryption keys. A semi-trusted server then uses the proxy re-encryption keys to translate a ciphertext into a form that can be decrypted by a specific granted user, and thus enforces access control for the data.

Vimercati et al. have constructed a scheme for securing data on semi-trusted storage servers based on key derivation methods of In their scheme, a secret key is assigned to each authorized user, and data blocks are grouped based on users that can access these blocks. One key is used to encrypt all blocks in the same group. Moreover, the data owner generates public tokens to be used along with the user's secret key to derive decryption keys of

specific blocks. The blocks and the tokens are sent to remote servers, which are not able to drive the decryption key of any block using just the public tokens. The approach in allows the servers to conduct a second level of encryption (over-encryption) to enforce access control of the data. Repeated access grant and revocation may lead to a complicated hierarchy structure for key management.

Wang et al. designed an over-encryption to enforce access control has also been used by In their scheme, the owner encrypts the data block-by-block, and constructs a binary tree of the block keys. The binary tree enables the owner to reduce the number of keys given to each user, where different keys in the tree can be generated from one common parent node. The remote storage server performs over-encryption to prevent revoked users from getting access to updated data blocks.

Popa et al. have introduced a cryptographic cloud storage system called CloudProof that provides read and write data sharing. CloudProof has been designed to offer security guarantees in the service level agreements of cloud storage systems. It divides the security properties in four categories: confidentiality, integrity, read freshness, and write-serializability. CloudProof can provide these security properties using attestations (signed messages) and chain hash. Besides, it can detect and prove to a third party that any of these properties have been violated. Read freshness and write-serializability in CloudProof are guaranteed by periodic auditing in a centralized manner. The time is divided into epochs, which are time periods at the end of each the data owner performs the auditing process. The authorized users send the attestations – they receive from the CSP during the epoch – to the owner for auditing. Like Plutus and SiRiUS, CloudProof supports two operations on the file blocks: read and write/modify.

***Discussion*.** Some aspects related to outsourcing data storage are beyond the setting of both PDP and POR, e.g., enforcing access control, and ensuring the newness of data delivered to authorized users. Even in the case of dynamic PDP, a verifier can validate the correctness of data, but the server is still able to cheat and return stale data to authorized users after the auditing process is done. The schemes have focused on access control and secure sharing of data on untrusted servers. The issues of full block-level dynamic operations (modify, insert, delete, and append), andachieving mutual trust between the data owners and the remote servers are outside their scope. Although have presented an efficient access control technique and handled full data dynamic over remote servers, data integrity, newness property, and mutual trust are not addressed. Authorized users in CloudProofare not performing immediate checking for freshness of received data; the attestations are sent at the end of each epoch to the owner for completing the auditing task. Instantaneous validation of data freshness is crucial before taking any decisions based on the received data from the cloud. CloudProof guarantees write-serializability, which is outside the scope of our current work as we are focusing on owner-write-users-read applications.

III. CURRENT DATA STORAGE CHALLENGES IN CLOUD

A cloud storage service provider should base its pricing on how much storage capacity a business has used, how much bandwidth was used to access its data, and the value-added services performed in the cloud such as security. Unfortunately, all the CSPS are not functioning in equal manners". Data storage paradigm in "Cloud" brings about many challenging design issues because of which the overall performance of the system get affected. Most of the biggest concerns with cloud data storage are:

3.1 Data integrity verification at un-trusted servers

For example, the storage service provider, which experiences Byzantine failures occasionally, may decide to hide the data errors from the clients for the benefit of their own. What is more serious is that for saving money and storage space the service provider might neglect to keep or deliberately delete rarely accessed data files which belong to an ordinary client. Consider the large size of the outsourced electronic data and the client's constrained resource capability, the core of the problem can be generalized as how can the client find an efficient way to perform periodical integrity verifications without the local copy of data files.

3.2 Data accessed by unauthorized users

The confidentiality feature can be guaranteed by the owner via encrypting the data before outsourcing to remote servers. For verifying data integrity over cloud servers, researchers have proposed provable data possession technique to validate the intactness of data stored on remote sites.

3.3 Location Independent Services

The very characteristics of the cloud computing services are the ability to provide services to their clients irrespective of the location of the provider. Services cannot be restricted to a particular location but may be requested from any dynamic location as per the choices of the customer.

3.4 Infrastructure and security

The infrastructure that is used for these services should be secured appropriately to avoid any potential security threats and should cover the life time of component.

3.5 Data recovery /Backup

For data recovery in cloud the user must concern the security as well as the bandwidth issue in consideration.

IV. SYSTEM AND ASSUMPTIONS

4.1 System Components and Relations

The cloud computing storage model considered in this work consists of four main components as illustrated in Fig. 1: (i) a data owner that can be an organization generating sensitive data to be stored in the cloud and made available for controlled external use; (ii) 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; (iii) authorized users – a set of owner's clients who have the right to access the remote data; and (iv) a trusted third party (TTP), an entity who is trusted by all other system components, and has capabilities to detect/specify dishonest parties.

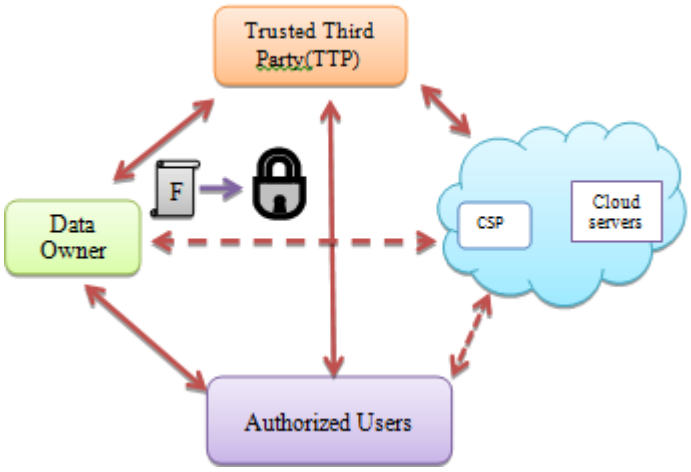

Fig. 1, Cloud data storage system model

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 relations with the CSP. Thus, the TTP is used to enable indirect mutual trust between these three components. There is a direct trust relation between the data owner and the authorized users.

Statement 1: The idea of using a third party auditor has been used before in outsourcing data storage systems, especially for customers with constrained computing resources and capabilities .The main focus of a third party auditor is to verify the data stored on remote servers, and give incentives to providers for improving their services. The proposed scheme in this work uses the TTP in a slightly different fashion. The auditing process of the data received from the CSP is done by the authorized users, and we resort to the TTP only to resolve disputes that may arise regarding data integrity or newness. Reducing the storage overhead on the CSP side is economically a key feature to lower the fees paid by the customers. Moreover, decreasing the overall computation cost in the system is another crucial aspect. To achieve these goals, a small part of the owner's work is delegated to the TTP.

4.2 Block-Level operations

The data owner has a file F consisting of m blocks to be outsourced to a CSP, where storage fees are pre-specified according to the used storage space. For confidentiality, the owner encrypts the data before sending to cloud servers. After data outsourcing, the owner can interact with the CSP to perform block-level operations on the file. These operations includes modify, insert, append, and delete specific blocks. In addition, the owner enforces access control by granting or revoking access rights to the outsourced data. An 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 in the system. 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.

4.3 Risk model

The CSP is untrusted, and thus the confidentiality and integrity of data in the cloud may be at risk. For economic reasons and maintaining a reputation, the CSP may hide data loss (due to hardware failure, management errors, various attacks), or reclaim storage by discarding data that has not been or is rarely accessed. To save the computational resources, the CSP may totally ignore the data update requests issued by the owner, or execute just a 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 honor 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 allege 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.

4.4 Requirements for Secure Storage

4.4.1 Confidentiality:

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

4.4.2 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.

4.4.3 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.

4.4.4 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.

4.4.5 CSP's defense:

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.

Combining the confidentiality, integrity, newness, access control, and CSP's defense properties in the proposed scheme enables the mutual trust between the data owner and the CSP. Thus, the owner can benefit from the wide range of facilities offered by the CSP, and at the same time, the CSP can mitigate the concern of cheating customers.

4.5 System Preliminaries

4.5.1 Lazy Revocation

The data owner can revoke the rights of some authorized users for accessing the outsourced data, the user can access the unmodified data block he cannot access the updated or new block.

4.5.2 Key Rotation

Key rotation is the technique user can generate a sequence of key by using initial key and a master secret key it has to property (i) the owner of the master secret key can generate next key in sequence (ii) Authorized users knowing the sequence of key can generate the previous keys.

4.5.3 Broadcast Encryption (bENC)

bENC is to enforce the access control over the outsourced data. This allows the broadcaster to encrypt the data for a set of arbitrary user, the set of user only can decrypt the message.

V. PROPOSED STORAGE SCHEMA

5.1 Existing scheme

Once the data has been outsourced to a remote CSP, which may not be trustworthy, the owner loses the direct control over the sensitive data. This lack of control raises the data owner's concerns about the integrity of data stored in the cloud. Conversely, a dishonest owner may falsely claim that the data stored in the cloud is corrupted to get some compensation. This mutual distrust between the data owner and the CSP, if not properly handled, may hinder the successful deployment of cloud architecture.

A straightforward solution to detect cheating from any side(data owner or CSP) is through using authentication tags (digital signatures). For a file F = $\{b_j\}_{1\leq j\leq m}$, the owner attaches a tag $OWN\sigma j$ with each block before outsourcing. The tags are generated per block not per file to enable dynamic operations at the block level without retrieving the whole outsourced file. The owner sends $\{bj, OWN\sigma j\}1\leq j\leq m$ to the CSP, where the tags $\{OWN\sigma j\}1\leq j\leq m$ are first verified. 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. The tags $\{OWN\sigma j\}1\leq j\leq m$ achieve non-repudiation from the owner side. When an authorized user (or the owner) requests to retrieve the data file, the CSP sends $\{b_j, OWN\sigma_j, CSP\sigma_j\}_{1\leq j\leq m}$, where $CSP\sigma_j$ is the CSP's signature/tag on $b_j||OWN\sigma_j$. The authorized user first verifies the tags $\{CSP\sigma_j\}_{1\leq j\leq m}$. In case of failed verification, the user asks the CSP to re-perform the transmission process. If $\{CSP\sigma_j\}_{1\leq j\leq m}$ are valid tags, the user then verifies the owner's tag $OWN\sigma_j$ on the block $bj \forall j$. If any tag $OWN\sigma_j$ is not verified, this indicates the corruption of data over the cloud servers. The CSP cannot repudiate such corruption for the owner's tags $\{OWN\sigma j\}1\leq j\leq m$ are previously verified and stored by the CSP along with the data blocks. Since the CSP's signatures $\{CSP\sigma_j\}_{1\leq j\leq m}$ 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 CSP can replace the new blocks and tags with old versions without being detected (*replay attack*). The above solution increases the storage overhead – especially for large files in order of gigabytes – on the cloud servers as each outsourced block is attached with a tag. Moreover, there is an increased computation overhead on different system components; the data owner generates a signature for each block, the CSP performs a signature verification for each outsourced block, and the authorized user (or the owner) verifies two signatures for each received block from the cloud servers. Thus, for a file F containing *m* blocks, the straightforward solution requires *2m* signature generations and *3m*signature verifications, which may be computationally a challenging task for large data files. For example, if the outsourced file is of size 1GB with 4KB block size, the straightforward solution requires 2^{19} signature generations and 3×2^{18} signature verifications. If the CSP receives the data blocks from a trusted entity (other than the owner), 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. However, the outsourced data must be kept private and any possible leakage of data towards the TTP must be prevented.

5.2 Overview Logic

The proposed scheme in this work addresses important issues related to outsourcing data storage: *data dynamic, newness, mutual trust,* and *access control.* The owner is allowed to update and scale the outsourced data file. Validating such dynamic data and its newness property requires the knowledge of some metadata that reflects the most recent modifications issued by the owner. Moreover, it requires the awareness of block indices to guarantee that the CSP has inserted, added, or deleted the blocks at the requested positions. To this end, the proposed scheme is based on using combined hash values and a small data structure, which we call block status table (BST). The TTP establishes the mutual trust among different system components in an indirect way. For enforcing access control of the outsourced data, the proposed scheme utilizes and combines three cryptographic techniques: *bENC, lazy revocation,* and *key rotation*. The bENC enables a data owner to encrypt some secret information to only authorized users allowing them to access the outsourced data file. Through lazy revocation, revoked users can read unmodified data blocks, while updated/new blocks are encrypted under new keys generated from the secret information broadcast to the authorized users. Using key rotation, the authorized users are able to access both updated/new blocks and unmodified ones that are encrypted under older versions of the current key.

5.3 Representations

- **F** is a data file to be outsourced composed of a sequence of m blocks, i.e., $F = \{b_1, b_2, \ldots, b_m\}$
- **h** is a cryptographic hash function
- **DEK** is a data encryption key
- $\mathbf{E_{DEK}}$is a symmetric encryption algorithm under DEK, e.g., AES (advanced encryption standard)
- $\mathbf{E^{-1}_{DEK}}$ is a symmetric decryption algorithm under DEK
- $\mathbf{F^{-1}}$ is an encrypted version of the file blocks
- $\mathbf{FH_{TTP}}$is a combined hash value for F-1, 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
- **Rot=<ctr,bENC(K_{ctr})>**is a rotator, where bENC(K_{ctr}) is a broadcast encryption of the key K.
- ⊕ is anXOR Operator

5.4 Block Status table (BST)

To reduce the computational task in the owner and CSP side, the trusted third party (TTP) is introduced to reduce generation of block tags and signature verification. Now validation of outsourced dynamic data and newness property are addressed in proposed schema, this is based on combined hash value and a small data structure called block status table (BST). The TTP established mutual trust among different system component. To enforce access control of outsourced data, this schema use cryptographic techniques: bENC, Key rotation and Lazy revocation. Block status table (BST) is a small dynamic data structure used to reconstruct and access the file blocks outsourced to the CSP. This table consist of the columns *Serial Number(SN)* is an index to the file blocks. It shows the physical locations of each blocks in the data file. *Block Number(BN)* is a counter to make logical numbering to each file blocks. *Key Version(KV)* is to indicate the version of the key used to encrypt the blocks in the data file. The BST is implemented as a linked list to simplify the insertion and deletion of table entity. *SN* is a simple index to the table so it is not needed. Where *BN* and *KV* is an integer of *8m* bytes, where *m* is the number of blocks in the file. When a data file is created initially counter (ctr) and *KV*is set to 1. When a block level modifications is performed then ctr is increased by 1 and *KV* of the modified/new block is set to ctr.

Fig 2 shows some examples on changes made in BST due to different dynamic operation on file $F = \{b_j\}_{1 \le j \le 8}$. (Fig2.a) ctr is set to 1, $SN_j = BN_j = j$, and$KV_j = 1$: $1 \le j \le 8$.(Fig 2.b) shows no change for updating the block at position 5 since no revocation is performed. (Fig 2.c) inserting new block after position 3 for a file F with new entry (4, 9, 1) here 4 is the physical position of newly created block and 9 is the logical number and the version number of key used to encrypt the file block is 1. The first revocation in the system increment the ctr by 1, now ctr =2, (Fig 2.d) now modify the block at the position five, new key is used for encryption of block number 4 now the table entity in the position five is (5, 4, 2). (Fig 2.e) shows new block is inserted in position 7 and the second revocation is done, which increments ctr by 1 and the table entity is (7, 10, 3) (Fig 2.f) shows the deleting the block at position 2 from the data file all the sub sequent entity are pop up by 1 step upward.

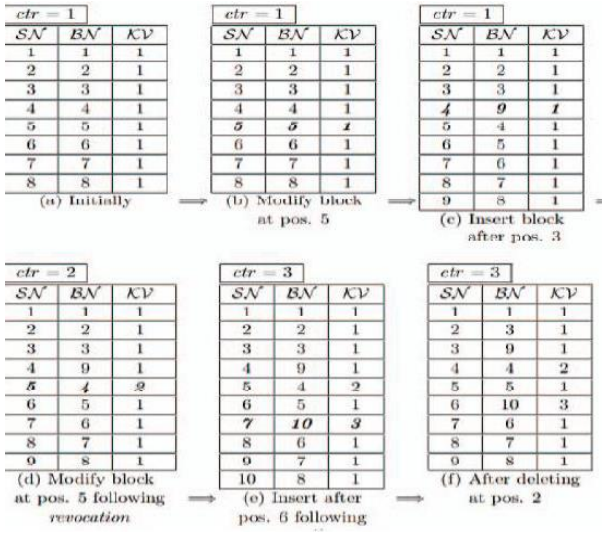

Fig.2,Changes in the BST due to different dynamic operations on a file F = $\{b_j\}_{1 \le j \le 8}$.

5.5 System Setup and File Preparation

The setup is done only once during the life time of the data storage system, which may be for tens of years. The system setup has two parts: one is done on the owner side, and the other is done on the TTP side.

5.5.1 Owner Role

The data owner initializes ctr to 1, and generates an initial secret key K_{ctr}/K_1. K_{ctr} can be rotated forward following user revocations, and rotated backward to enable authorizedusers to access blocks that are encrypted under older versions of K_{ctr}.

For a fileF = $\{b_j\}_{1 \le j \le m}$, the owner generates a BST with $SN_j = BN_j = j$, and KV_j = ctr. To achieve privacy-preserving, the owner creates an encrypted file version $\tilde{F} = \{\tilde{b}_j\}_{1 \le j \le m}$Moreover, the owner creates a rotator *Rot=<ctr,bENC(K_{ctr})>*, where bENC enables only authorized users to decrypt K_{ctr} and access the outsourced file. The owner sends { $\tilde{F}$, BST,Rot} to the TTP, and deletes the data file from itslocal storage.

Embedding *BN j* with the block b_j during the encryption process helps in reconstructing the file blocks in the correct order. If the encrypted blocks are not corrupted over cloud servers, but randomly delivered to an authorized user, the latter can utilize the embedded *BNj*and the BST to orderly reconstruct the data file F.

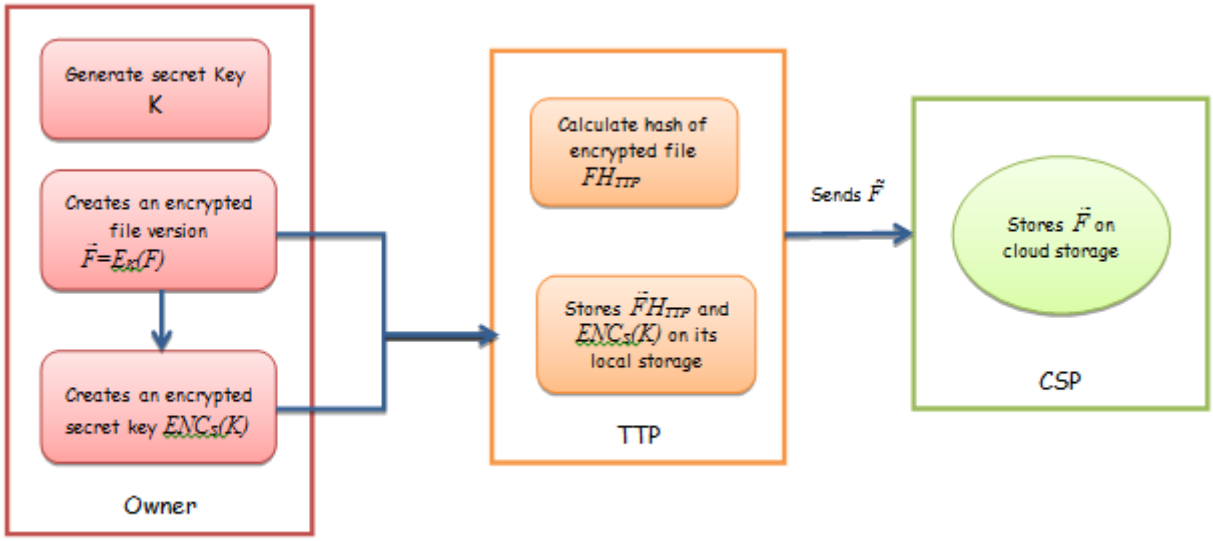

Fig.2, Setup and file preparation for data outsourcing

5.5.2 TTP Role

A small part of the owner's work is delegated to the TTP toreduce the storage overhead and lower the overall system computation. For the TTP to resolvedisputes that may arise regarding data integrity/newness, it computes and locally stores combinedhash values for the encrypted file $\tilde{F}$ and the BST. The TTP computes $FHTTP = \bigoplus^{m}_{j=1} h(\tilde{b}_j)$ and $TH_{TTP} = \bigoplus^{m}_{j=1} h(BN_j \| KV_j)$,then sends $\{\tilde{F}$, BST$\}$ to the CSP. The TTP keeps only $FH_{TTP}$and TH_{TTP} on its local storage.

Statement 2: The BST is used by the authorized users to reconstruct and access the outsourced data file. The proposed scheme in this work assumes that the data owner is intermittently online and the authorized users are enabled to access the data file even when the owner is offline. To this end, the CSP stores a copy of the BST along with the outsourced data file. When an authorized user requests to access the data, the CSP responds by sending both the BST and the encrypted file e F. Moreover, the BST is used during each dynamic operation on the outsourced data file, where one table entry is modified/inserted/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 on the outsourced file. 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. Recall that the BST is a small dynamic data structure with a table entry size = 8 bytes. For 1GB file with 4KB block size, the BST size is only 2MB (0.2% of the file size). Table 1 summarizes the data stored by each component in the proposed scheme.

Owner	TTP	CSP
ctr, *Kctr*, BST_O	*Rot*, *FHTTP* , *THTTP*	*F*, BST_C

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

5.6 Dynamic Tasks on the Outsourced Data

The dynamic operations in the proposed scheme are performed at the block level via a request in the general form (BlockOp,$TEntry_{BlockOp}$, j ,KV_j , h($\tilde{b}_j$), RevFlag , $\tilde{b}^*$) where BlockOp corresponds to block modification (denoted by BM), block insertion(denoted by BI), or block deletion (denoted by BD). $TEntry_{BlockOp}$ indicates an entry in BST_O correspondingto the issued dynamic request. The parameter j indicates the block index on which thedynamic operation is to be performed, *KVj* is the value of the key version at index j of BST_Obefore running a modification operation, and h($\tilde{b}$j) is the hash value of the block at index j beforemodification/deletion. *RevFlag* is a 1-bit flag (true/false and is initialized to false) to indicatewhether a revocation has been performed, and $b^*$is the new block value.

5.6.1 Modification

Data modification is one of the most frequently used dynamic operations in theoutsourced data. For a file*F = {b*1, *b*2, . . . ,*bm}*, suppose the owner wants to modify a block *bj*with a block b'j .The owner uses the technique of one-sender-multiple-receiver(OSMR) transmission to send the modify request to both the CSP and the TTP.The TTP updates the combined hash value $FH_{TTP}$for $\tilde{F}$ through the step $FH_{TTP} = FH_{TTP} \oplus h(\tilde{b}_j) \oplus h(\tilde{b}'j)$, which simultaneously replaces the hash of the old block h($\tilde{b}$j) with the new one h($\tilde{b}$'j). This is possible due to the basic properties of the $\oplus$operator. The same idea is used whenRevFlag = true to update the combined hash value TH_{TTP} on the TTP side by replacing thehash of the old table entry at index j with the hash of the new value.

5.6.2 Insertion

In a block insertion operation, the owner wants to insert a new block $\bar{b}$ after indexj in a file*F = {b*1, *b*2, . . . , *bm}* i.e., the newly constructed file F' = *{b*1, *b*2, . . . *bj,$\bar{b}$,……,bm+1}*,where $b_{j+1} = \bar{b}$. The block insertion operation changes the logical structure of the file, whileblock modification does not.

5.6.3 Append

Block append operation means adding a new block at the end of the outsourced data.It can simply be implemented via insert operation after the last block of the data file.

5.6.4 Deletion

Block deletion operation is the opposite of the insertion operation. When one block isdeleted all subsequent blocks are moved one step forward. The step $FH_{TTP} = FH_{TTP} \oplus h(\tilde{b}_j)$ is used to delete the hash value of the $\tilde{b}_j$ from the combined hash FH_{TTP}(properties of $\oplus$operator). The same idea is used with the TH_{TTP} value.

5.7 Data Access and Cheating Detection

An authorized user sends a data-access request to both the CSP and the TTP to access the outsourced file. The user receives $\{\tilde{F}, BST_C, \sigma_F, \sigma_T\}$ from the CSP, and $\{FH_{TTP}, TH_{TTP}, Rot\}$ from the TTP. For achieving non-repudiation, the CSP generates two signatures σ_F and σ_T for $\tilde{F}$ and BST_C,respectively.

5.7.1 Verification of encrypted data file

The authorized user verifies the signatures, and proceeds with the data access procedure only if both signatures are valid. The authorized user verifies the contents of BST_C entries by computing $TH_U = \bigoplus^{m}_{j=1} h(BN_j||KV_j)$, and comparing it with the authentic value TH_{TTP} received from the TTP. If the user claims that $TH_U \neq TH_{TTP}$, a report is issued to the owner 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 file $\tilde{F}$ by computing $FH_U = \bigoplus^{m}_{j=1} h(\tilde{b}_j)$ and comparing with FH_{TTP} . If there is a dispute that $FH_U \neq FH_{TTP}$, the owner is informed and we resort to the TTP to resolve such a conflict.

For the authorized user to access the encrypted file $\tilde{F} = \{\tilde{b}_j\}_{1 \leq j \leq m}$, BST_C and *Rot* are used to generate the key *DEK* that decrypts the block $\tilde{b}j$. 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 block $\tilde{b}_j$. Both *ctr*and the key version KV_j can determine how many rotation steps for $K_{ctr}$with each block $\tilde{b}_j$. Decrypting the block $\tilde{b}_j$ returns $(BN_j||b_j)$. Both BN_j and BST_C are utilized to get the physical block position SN_j into which the block b_j is inserted, and thus the file F is reconstructed in plain form.

Optimization The backward key rotation done can be highly optimized by computing a set of keys $Q = \{K_i\}$ from K_{ctr}. Each $K_i$in Q is the result of rotating K_{ctr} backward $ctr-I$ times. For example, if ctr= 20, a set $Q = \{K_1, K_5, K_{10}, K_{15}\}$can be computed from K_{ctr}. To decrypt a block $\tilde{b}_j$, the authorized user chooses one key K_i from Q, which has the minimum positive *distance* $i - KV_j$. Then, K_i is rotated backward to get the actual key that is used to decrypt the block $\tilde{b}_j$. A relatively large portion of the outsourced data is kept unchanged on the CSP, and thus K_1 from Q can be used to decrypt many blocks without any further key rotation. The size of Q is negligible compared with the size of the received data file.

5.7.2 Cheating Detection Procedure

Fig. 4 shows how the TTP determines the dishonest party in the system. The TTP verifies the signatures σ_T and σ_F, which are previously verified and accepted by theauthorized user. If any signature is invalid, this indicates that the owner/user is dishonest for corrupting either the data or the signatures. In case of valid signatures, the TTP computes temporary combined hash values $TH_{temp} = \bigoplus^{m}_{j=1} h(BN_j||KV_j)$ and $FH_{temp} = \bigoplus^{m}_{j=1} h(\tilde{b}_j)$. If $TH_{temp} \neq TH_{TTP}$ or $FH_{temp} \neq FH_{TTP}$, this indicates that the CSP is dishonest for sending corrupted data, otherwise the owner/user is dishonest for falsely claiming integrity violation of received data.

VI. PERFORMANCE ANALYSIS

6.1 Settings and Overheads

The data file F used in our performance analysis is of size 1GB with 4KB block size. Without loss of generality, we assume that the desired security level is 128-bit. Thus, we utilize a cryptographic hash *h* of size 256 bits (e.g., SHA-256), an elliptic curve defined over Galois field *GF(p)* with |p| = 256 bits (used for bENC), and BLS (Boneh-Lynn-Shacham) signature of size 256 bits (used to compute $\sigma_F$andσ_T). Here we evaluate the performance of the proposed scheme by analyzing the storage, communication, and computation overheads. We investigate overheads that the proposed scheme brings to a cloud storage system for static data with only confidentiality requirement. This investigation demonstrates whether the features of our scheme come at a reasonable cost. The computation overhead is estimated in terms of the used cryptographic functions, which are notated . Let m and n denote the number of file blocks and the total number of system users, respectively. It presents a theoretical analysis for the storage, communication, and computation overheads of the proposed scheme. It summarizes the storage and communication overheads for our data file F (1GB with 4KB block size) and 100,000 authorized users.

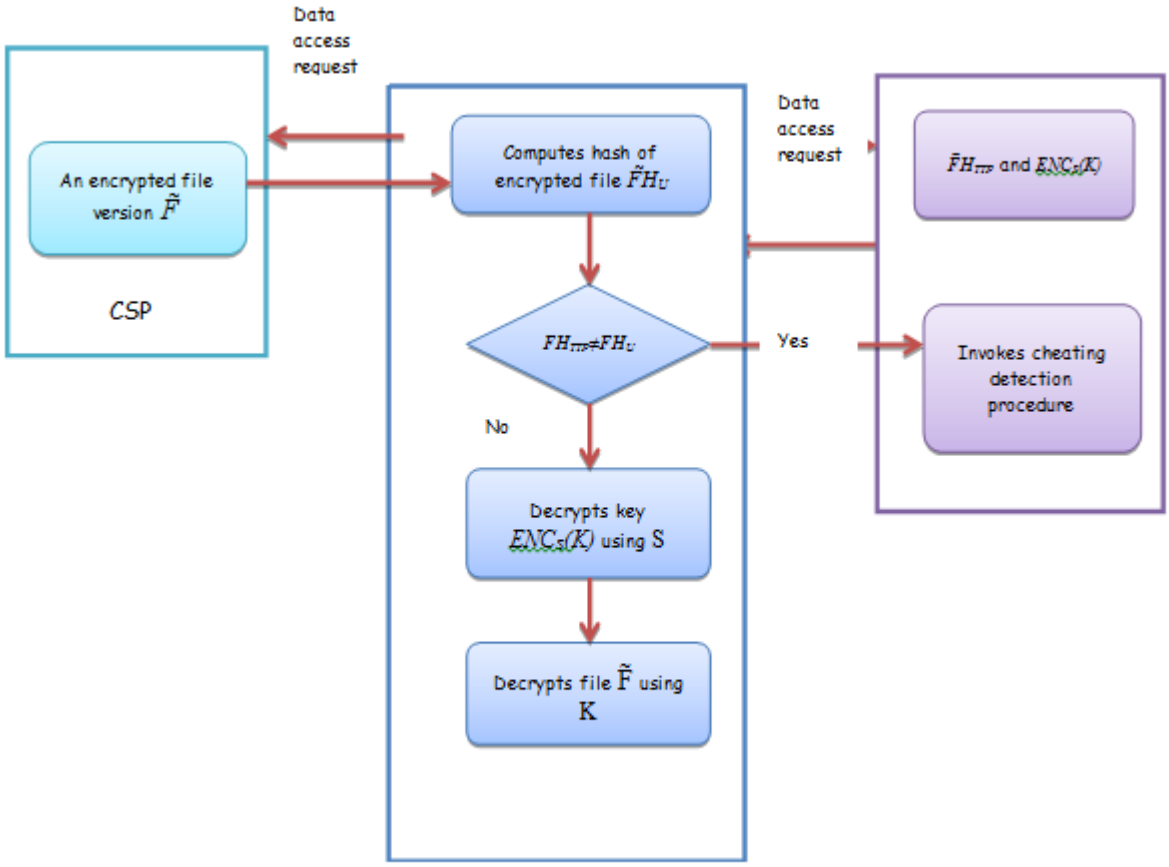

Fig.3, Data access and Verification of encrypted data file

Observations

Storage overhead It is the additional storage space used to store necessary information other than the outsourced file $\tilde{F}$. The overhead on the owner side is due to storing BST_O. An entry of $BST_O$is of size 8 bytes (two integers), and the total number of entries equals the number of file blocks *m*.

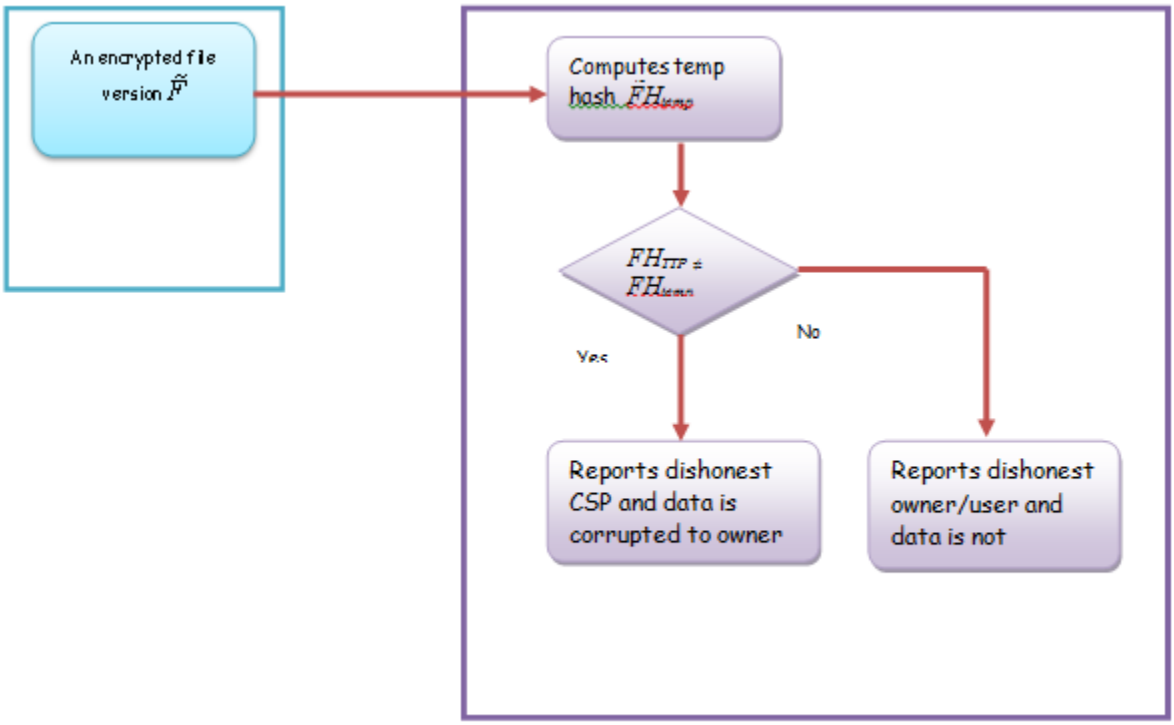

Fig.4, Cheating detection procedure

During implementation *SN* is not needed to be stored in BST_O; *SN* is considered to be the entry/table index ($BST_O$is implemented as a linkedlist). The size of $BST_O$for the file F is only2MB (0.2% of F). BST_O size can be further reducedif the file *F* is divided into larger blocks (e.g., 16KB). Like the owner, the storage overhead on the CSP side comes from the storage of BST_C. To resolve disputes that may arise regarding data integrity or newness property, the TTP stores FH_{TTP} and TH_{TTP} , each of size 256 bits. Besides, the TTP stores *Rot* =<*ctr*, $bENC(K_{ctr})$>that enables the data owner to enforce access control for the outsourced data. The *ctr* is 4 bytes, and bENC has storage complexity O(√n), which is practical for an organization (data owner) with n = 100,000 users. A point on the elliptic curve used to implement bENC can be represented by 257 bits (≈ 32 bytes) using compressed representation .Therefore, the storage overhead on the TTP side is close to 10KB, which is independent of the outsourced file size. Overall, the storage overhead for the file F is less than 4.01MB (≈ 0.4% of F).

Communication overhead It is the additional information sent along with the outsourced data blocks. Duringdynamic operations, the communication overhead on the owner side comes from the transmission of a block operation *BlockOP* (can be represented by 1 byte), a table entry $TEntry_{BlockOP}$ (8 bytes), and a block index j (4bytes). If a block is to be modified following a revocation process, KV_j(4 bytes) is sent to the TTP. Moreover, incase of a block modification/deletion, the owner sendsa hash (32 bytes) of the block to be modified/deleted to the TTP for updating FH_{TTP} . Recall that the owner also sends *Rot (4 + 32 √n bytes)* to the TTP if block modifications/ insertions are to be performed following user revocations. Therefore, in the worst case scenario (i.e., block modifications following revocations), the owner's overhead is less than 10KB. The Rot represents the major factor in the communication overhead, and thus the overhead is only 45 bytes if block modification/deletion operations are to be performed without revocations (only 13 bytes for insertion operations). In practical applications, the frequency of dynamic requests to the outsourced data is higher than that of user revocations. 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, the proposed scheme has communication overhead close to 2.01MB (≈ 0.2% of F).

Computation overhead: A cloud storage system for static data with only confidentiality requirement has computation cost for encrypting the data before outsourcing and decrypting the data after being received from the cloud servers. For the proposed scheme, the computation overhead on the owner side due to dynamic operations (modification/insertion) comes from computing DEK = h(Kctr) and encrypting the updated/inserted block, i.e., the overhead is one hash and one encryption operations. If a block modification/insertion operation is to be performed following a revocation of one or more users, the owner performs *FR* to roll K_{ctr} forward, and bENC to generate the Rot. Hence, the computation overhead on the owner side for the dynamic operations is $h + E_{DEK} + FR + bEnc$(worst case scenario). Updating BST_O and BST_C is done without usage of cryptographic operations (add, remove, or modify a table entry).

To reflect the most recent version of the outsourced data, the TTP updates the values FH_{TTP} and TH_{TTP} .If no revocation has been performed before sending a modify request, only FH_{TTP} is updated on the TTP side. Therefore, the maximum computation overhead on the TTP side for updating both FH_{TTP} and TH_{TTP} is *4h*. Before accessing the data received from the CSP, theauthorized user verifies two signatures (generated by the CSP), BSTC entries, and the data file. These verifications cost $2V_{\sigma}$+ *2mh*. Moreover, the authorized user decrypts $bENC(K_{ctr})$ part in the *Rot* to get K_{ctr}. For each received block, $K_{ctr}$is rotated backward to obtain the actual key that is used to decrypt the data block. The optimized way of key rotation (using the set Q) highly affects the performance of data access; many blocks need a few or no rotations. Moreover, one hash operation is performed per block to compute DEK. Overall, the computation overhead due to data access is $2\ V_{\sigma}+ 3mh + bENC^{-1} + [BR]$ on the owner side, and $2S_{\sigma}$on the CSP side. For determining a dishonest party, the TTP verifies $\sigma_T$andσ_F. In case of valid signatures, the TTP proceeds to compute TH_{temp} and FH_{temp}. The values TH_{temp} and $FH_{temp}$are compared with TH_{TTP} and FH_{TTP} ,respectively. Hence, the maximum computation overhead on the TTP side due to cheating detection is $2V_{\sigma}+ 2mh$.

VII. ENACTMENT AND EXPERIMENTAL VALUATION

7.1 Enactment

We have implemented the proposed scheme on top of Amazon Elastic Compute Cloud (Amazon EC2) andAmazon Simple Storage Service (Amazon S3) cloud platforms. Our implementation of the proposed scheme consists of four modules: *OModule*(owner module), *CModule* (CSP module), *UModule*(user module), and *TModule* (TTP module). *OModule*, which runs on the owner side, is a library to be used by the owner to perform the owner role in the setup and file preparation phase. Moreover, this library is used by the owner during the dynamic operations on the outsourced data. *CModule* is a library that runs on Amazon EC2 and is used by the CSP to store, update, and retrieve data from Amazon S3. *UModule*is a library to be run at the authorized users' side, and include functionalities that allow users to interact with the TTP and the CSP to retrieve and access the outsourced data. *TModule*is a library used by the TTP to perform the TTP role in the setup and file preparation phase. Moreover, the TTPuses this library during the dynamic operations and todetermine the cheating party in the system.

7.2 Implementation settings

In our implementation we use a "large" Amazon EC2 instance to run *CModule.*

This instance type provides total memory of size 7.5GB and 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each). One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0 - 1.2GHz 2007 Opteron or 2007 Xeon processor. A separate server in the lab is used to run *TModule*. This server has Intel(R) Xeon(TM) 3.6GHz processor, 2.75GB RAM, and Windows XP operating system. The *OModule* is executed on a desktop computer with Intel(R) Xeon(R) 2GHz processor and 3GB RAM running Windows XP. A laptop with Intel(R) Core(TM) 2.2GHz processor and 4GB RAM running Windows 7 is used to execute the *UModule*. We outsource a data file of size 1GB to Amazon S3. Algorithms (hashing, broadcast encryption, digital signatures, etc.) are implemented using MIRACL library version 5.5.4. For a 128-bit security level, bENCuses an elliptic curve with a 256-bit group order. In the experiments, we utilize SHA-256, 256-bit BLS signature, and Barreto- Naehrig (BN) curve defined over prime field GF(p) with |p| = 256 bits and embedding degree = 12.

7.3 Experimental Valuation

Here we describe the experimental evaluation of the computation overhead the proposed scheme brings to a cloud storage system that has been dealing with static data with only confidentiality requirement.

Owner computation overhead To experimentally evaluate the computation overhead on the owner side due to the dynamic operations, we have performed 100 different block operations (modify, insert, append, and delete) with number of authorized users ranging from 20,000 to 100,000. We have run our experiment three times, each time with a different revocation percentage. In the first time, 5% of 100 dynamic operations are executed following revocations. We increased the revocation percentage to 10% for the second time and 20% for the third time. Fig. 8 shows the owner's average computation overhead per operation. For a large organization (data owner) with 100,000 users, performing dynamic operations and enforcing access control with 5% revocations add about 63milliseconds of overhead. With 10% and 20% revocation percentages, which are high percentages than an average value in practical applications, the owner overhead is 0.12 and 0.25 seconds, respectively. Scalability (i.e., how the system performs when more users are added) is an important feature of cloud storage systems. The access control of the proposed scheme depends on the square root of the total number of system users. Fig. 8 shows that for a large organization with 105 users, performing dynamic operations and enforcing access control for outsourced data remains practical.

TTP computation overhead In the worst case, the TTP executes only 4 hashes per dynamic request to reflect the change on the outsourced data. Thus, the maximum computation overhead on the TTP side is about 0.04 milliseconds, i.e., the proposed scheme brings light overhead on the TTP during the normal system operations. To identify the dishonest party in the system in case of disputes, the TTP verifies two signatures (σ_F and σ_T), computes combined hashes for the data (file and table), and compare the computes hashes with the authentic values (TH_{TTP} and FH_{TTP}). Thus, the computation overhead on the TTP side is about 3.59 seconds. Through our experiments, we use only one server to simulate the TTP and accomplish its work. The TTP may choose to split the work among a few devices or use a single devicewith a multi-core processor which is becoming prevalent these days, and thus the computation time on the TTP side is significantly reduced in many applications.

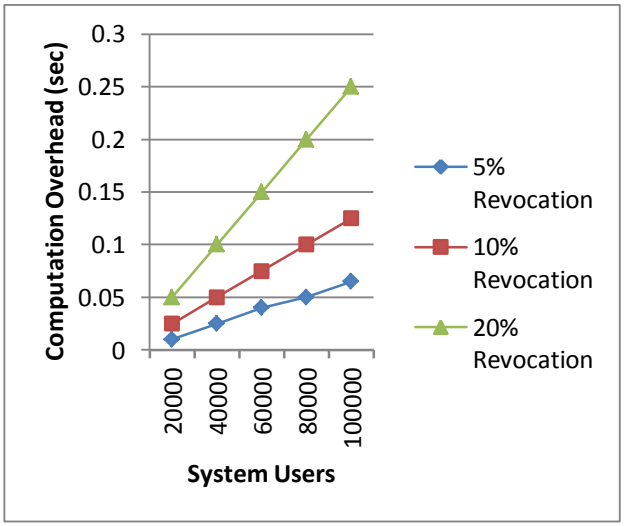

Fig.5, Owner's average computation overhead due to dynamic operations

TABLE 2: Experimental results of the computation overheads

Component	TTP	Users	CSP
Computation Overhead	0.04 ms / 3.59s	0.55 s	6.04 s

User computation overhead The computation overhead on the user side due to data access comes from five aspects divided into two groups. The first group involves signatures verification and hash operations to verify the received data (file and table). The second group involves broadcast decryption, backward key rotations, and hash operations to compute the DEK. The first group costs about 5.87 seconds, which can be easily hidden in the receiving time of the data (1GB file and 2MB table). To investigate the time of the second group, we access the file after running 100 different block operations (with 5% and 10% revocation percentages). Moreover, we implement the backward key rotations in the optimized way. The second group costs about 0.55 seconds, which can be considered as the user's computation overhead due to data access.

CSP computation overhead. As a response to the data access request, the CSP computes two signatures: $\sigma_F$andσ_T. Thus, the computation overhead on the CSP side due to data access is about 6.04 seconds and can be easily hidden in the transmission time of the data (1GB file and 2MB table).

VIII. CONCLUSION

The proposed schema for cloud based storage which supports outsourcing of dynamic data, updating and scaling of outsourced data in remote server is ensuring the authorized user receiving most recently updated versions of data. TTP can determine the dishonest party. We have investigated the overheads added by our scheme when incorporated into a cloud storage model for static data with only confidentiality requirement. The storage overhead is ≈ 0.4% of the outsourced data size, the communication overhead due to block-level dynamic changes on the data is ≈ 1% of the block size, and the communication overhead due to retrieving the data is ≈ 0.2% of the outsourced data size.

REFERENCES

[1] Amazon.com, "Amazon Web Services (AWS)," Online at http://aws.amazon.com, 2008.

[2] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks, 2008, pp. 1–10.

[3] 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.

[4] 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.

[5] 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 .

[6] 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.

[7] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," IEEE Trans. on Knowl. And Data Eng., vol. 20, no. 8, 2008.

[8] G. Ateniese, 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 andCommunications Security, ser. CCS '07, 2007, pp. 598–609.

[9] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proceedings of the 16th ACM Conferenceon Computer and Communications Security, 2009, pp. 213–222.

[10] 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.

[11] 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/.

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

[13] 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 InternationalConference on Very Large Data Bases. ACM, 2007, pp. 123–134.

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

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

[16] 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.

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

[18] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in ASIACRYPT '01: Proceedings of the 7th InternationalConference on the Theory and Application of Cryptology andInformation Security, London, UK, 2001, pp. 514–532.

[19] P. S. L. M. Barreto and M. Naehrig, "IEEE P1363.3 submission: Pairing-friendly elliptic curves of prime order with embedding degree 12," New Jersey: IEEE Standards Association, 2006.

[20] D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," Cryptology ePrintArchive, Report 2006/150, 2006.

[21] D. Naor, M. Naor, and J. B. Lotspiech, "Revocation and tracing schemes for stateless receivers," in Proceedings of the 21st AnnualInternational Cryptology Conference on Advances in Cryptology, ser. CRYPTO '01. Springer-Verlag, 2001, pp. 41–62.

[22] M. J. Atallah, K. B. Frikken, and M. Blanton, "Dynamic and efficient key management for access hierarchies," in Proceedings of the 12th ACM Conference on Computer and Communications Security,ser. CCS '05. ACM, 2005, pp. 190–202.

[23] J. Feng, Y. Chen, W.-S. Ku, and P. Liu, "Analysis of integrity vulnerabilities and a non-repudiation protocol for cloud data storage platforms," in Proceedings of the 2010 39th International Conference on Parallel Processing, 2010, pp. 251–258.

[24] J. Feng, Y. Chen, and D. H. Summerville, "A fair multi-party non-repudiation scheme for storage clouds," in 2011 International Conference on Collaboration Technologies and Systems, 2011, pp. 457–465.

AUTHORS

First Author – S.Sandhiya, University College of Engineering Villupuram, India, Department of Information Technology, Email: sksandhiya89@gmail.com
Second Author – D.Saranya, University College of Engineering Villupuram, India, Department of Information Technology, Email: dsaranya93@gmail.com
Third Author – S.Archana, University College of Engineering Villupuram, India, Department of Information Technology, Email: archanaanash@gmail.com
Fourth Author – M.Jayasudha, University College of Engineering Villupuram, India, Department of Information Technology