Enabling Indirect Mutual Trust for Cloud Storage Systems

Chintal Maisheri
ME student (Computer)
K. J. Somaiya College of Engineering, Mumbai
INDIA

Deepak Sharma
Associate Professor
K. J. Somaiya College of Engineering, Mumbai
INDIA

ABSTRACT
Cloud computing is a forthcoming revolution in information technology (IT) industry because of its performance, accessibility, low cost and many other luxuries. It provides gigantic storage for data and faster computing to customers over the internet. It essentially shifts the database and application software to the large data centers, i.e., Cloud, where management of data and services may not be completely trustworthy. That is why companies are reluctant to deploy their business in the cloud even cloud computing offers a wide range of luxuries. Security of data in the cloud is one of the major issues which acts as an obstacle in the implementation of cloud computing.

In the current era of digital world, the amount of sensitive data produced by many organizations is outpacing their storage ability. The management of such huge 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 misbehaviour committed by the CSP. On the other hand, the CSP needs a protection from any false accusations that may be claimed by the owner to get illegal 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 two important features: (i) It allows the owner to outsource sensitive data to a CSP, and it ensures that only authorized users (i.e., Those who have the right to access the owner's file) receive the outsourced data i.e. It enforces the access control of the outsourced data. (ii) It enables indirect mutual trust between the owner and the CSP.

Keywords
Access control, Cloud computing, Cloud service provider, Data security, Data outsourcing, Mutual trust, Storage-as-a-Service

1. INTRODUCTION
Cloud computing has received considerable attention from both academia and industry due to a number of important advantages including: cost effectiveness, low management overhead, immediate access to a wide range of applications, flexibility to scale up and down information technology (IT) capacity, and mobility where customers can access information wherever they are, rather than having to remain at their desks. 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. In the current era of digital world, various organizations produce a large amount of sensitive data including personal information, electronic health records, and financial data. The local management of such huge amount of data is problematic and costly due to the requirements of high storage capacity and qualified personnel. Therefore, Storage-as-a-Service offered by cloud service providers (CSPs) emerged as a solution to mitigate the burden of large local data storage and reduce the maintenance cost by means of outsourcing data storage. Through outsourcing data storage scenario, data owners delegate the storage and management of their data to a CSP in exchange for pre-specified fees metered in GB/month. Such outsourcing of data storage enables owners to store more data on remote servers than on private computer systems. Moreover, the CSP often provides better disaster recovery by replicating the data on multiple servers across multiple data centers achieving a higher level of availability. Thus, many authorized users are allowed to access the remotely stored data from different geographic locations making it more convenient for them.

Since the data owner physically releases sensitive data to a remote CSP, there are some concerns regarding confidentiality, integrity, and access control of the data. The confidentiality feature can be guaranteed 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 remainder of the paper is organized as follows. Section II contains related work on areas of integrity verification of outsourced data, cryptographic file systems in distributed networks, data storage security on untrusted remote servers and access control of outsourced data. Section III describes different system components and their relations. Threat model and security requirements namely, confidentiality, integrity, access control, and CSPs defence are also discussed in section III. In Section IV, cloud-based storage scheme is proposed to achieve mutual trust between a data owner and a CSP and

hence to build trusted environment in cloud. In section V, security of the scheme is analyzed against its fulfilment of assigned security requirements, namely, confidentiality, integrity, access control, and CSPs defence. In section VI, prototype implementation on Eucalyptus cloud computing platform and walrus storage is presented. Finally, Section VII summarizes our conclusion and gives a number of future work directions.

2. RELATED WORK

Existing research work can be found in the areas of integrity verification of outsourced data, data storage security on untrusted remote servers and access control of outsourced data.

The term cloud had already come into commercial use in the early 1990s to refer to large Asynchronous Transfer Mode networks. By 21st century, he term ‘‘cloud computing’’ had appeared, although major focus at this time was on Software as a Service (SaaS). 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. On the other side along with these benefits’ cloud computing faces big challenge i.e. data storage security problem, which is an important aspect of Quality of Service (QoS). Once user puts data on the cloud rather than locally, he has no control over it i.e. unauthorized users could modify user’s data or destroy it and even cloud server collude attacks. Cloud users are mostly worried about the security and reliability of their data in the cloud. Amazon’s S3 [1] is such a good example.

2.1 Integrity verification of outsourced data

For verifying data integrity over cloud servers, researchers have proposed provable data possession technique to validate the intactness of data stored on remote sites. A number of PDP protocols have been presented to efficiently validate the integrity of data, e.g., [2]–[5]. Proof of retrievability was introduced as a stronger technique than PDP in the sense that the entire data file can be reconstructed from portions of the data that are reliably stored on the servers. Juels et al. [6] Described a formal “proof of retrievability” (POR) model for ensuring the remote data integrity. Their scheme combines spot-checking and error-correcting code to ensure both possession and retrievability of files on archive service systems. Shacham et al. [7] Built on this model and constructed a random linear function based homomorphic authenticator which enables unlimited number of queries and requires less communication overhead.

2.2 Data storage security on untrusted remote servers

Commonly, traditional access control techniques assume the existence of the data owner and the storage servers 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 unable to access the data since they do not have the decryption key. This general solution has been widely incorporated into existing schemes [8]–[11], which aim at providing data storage security on untrusted remote servers.

Kallahalla et al. [8] 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.

Goh et al. [9] 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.

Based on proxy re-encryption [15], Ateniese et al. [10] have introduced a secure distributed storage protocol. 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 of the data. Vimercati et al. [11] have constructed a scheme for securing data on semi-trusted storage servers based on key derivation methods of [16]. 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 [11] 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 [17].

2.3 Access control of outsourced data

The concept of over-encryption to enforce access control has also been used by Wang et al. [17]. 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.

Another class of solution utilizes attribute-based encryption to achieve fine-grained access control [12], [13]. However these schemes do not implement mutual trust between the data owner and the remote servers.

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 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 [14]. In this work, a scheme is proposed that addresses important issues related to outsourcing the storage of data, namely confidentiality, integrity and access control. Mutual trust between the data owner and the CSP is another imperative issue, 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. The proposed cloud-based storage scheme has the following features: (i) it allows a data owner to outsource the data to a CSP, and it ensures that only authorized users (i.e., Those who have the right to access the owner's file) receive the outsourced data i.e. It enforces the access control of the outsourced data. (ii) It establishes indirect mutual trust between the data owner and the CSP since each party resides in a different trust domain.

3. OUR SYSTEM AND ASSUMPTIONS

3.1 System components and relations

3.1.1 Data owner

That can be an organization / individual generating sensitive data to be stored in the cloud and made available for controlled external use.

3.1.2 Cloud service provider (CSP)

Who manages cloud servers and provides paid storage space on its infrastructure to store the owner's files and make them available for authorized users.

3.1.3 Authorized users

A set of owner's clients who have the right to access the remote data.

3.1.4 Trusted third party (TTP)

An entity who is trusted by all other system components, and has capabilities to detect/specify dishonest parties.

The cloud computing storage model considered in this work consists of four main components as illustrated in Figure 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 relationship between the data owner and the authorized users.

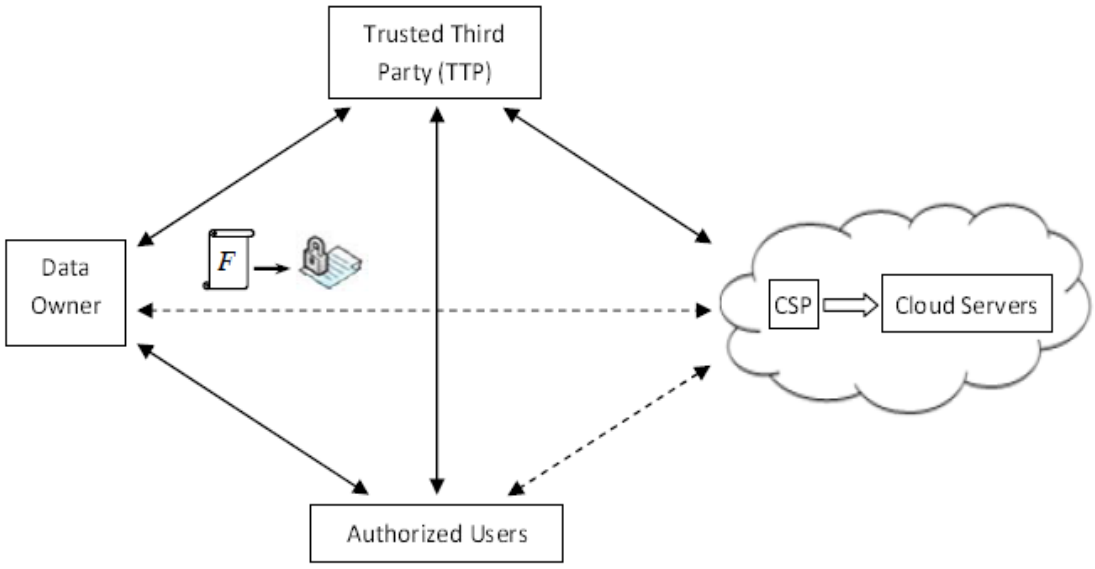


Fig 1: Cloud computing data storage system model

3.2 Outsourcing and accessing

For confidentiality, the owner encrypts the data before sending to cloud servers. To access the data, the authorized user sends a data-access request to the CSP, and receives the data file in an encrypted form that can be decrypted using a secret key generated by the authorized user. It is assumed that the interaction between the owner and the authorized users to authenticate their identities has already been completed, and it is not considered in this work. The TTP is an independent entity, and thus has no incentive to collude with any party. However, any possible leakage of data towards the TTP must be prevented to keep the outsourced data private. The TTP and the CSP are always online, while the owner is intermittently online. The authorized users are able to access the data file from the CSP even when the owner is offline.

3.3 Threat model

The CSP is untrusted, and thus the confidentiality and integrity of data in the cloud may be at risk. For economic incentives and maintaining a reputation, the CSP may hide data loss, or reclaim storage by discarding data that have not been or is rarely accessed. On the other hand, a data owner and authorized users may collude and falsely accuse the CSP to get a certain amount of reimbursement. They may dishonestly claim that data integrity over cloud servers has been violated.

3.4 Security requirements

3.4.1 Confidentiality

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

3.4.2 Integrity

Outsourced data are required to remain intact on cloud servers. The data owner and authorized users must be enabled to recognize data corruption over the CSP side.

3.4.3 Access control

Only authorized users are allowed to access the outsourced data.

3.4.4 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.

4. PROPOSED FRAMEWORK

4.1 Existing system

A straight forward solution to detect cheating from any side is through digital signatures. For each file owner attaches digital signature before outsourcing. The CSP first verifies digital signature of owner before storing data on cloud. In case of failed verification, the CSP rejects to store data and asks the owner to resend the correct signature. If the signature is valid, both the file and signature are stored on the cloud servers. The digital signature achieves non-repudiation from the owner side. When an authorized user (or the owner) requests to retrieve the data file, the CSP sends file, owner's signature and CSP's signature on (file || owner's signature). The authorized user first verifies the CSP's signature. In case of failed verification, the user asks CSP to re-perform the transmission process. If CSP's signature is valid, the user then verifies owner's signature. If verification fails, this indicates the corruption of data over the cloud servers. The CSP cannot repudiate such corruption for the owner's signature is previously verified and stored by the CSP along with file. Since CSP's signature is attached with the received data, a dishonest owner cannot falsely accuse the CSP regarding data integrity.

The above solution increases the storage overhead on cloud as owner's signature is stored along with the file on cloud servers. Moreover, there is an increased computation overhead, CSP has to verify signature of owner before storing file on cloud, and the authorized user verifies two signatures for each received file. If the CSP receives file from trusted entity other than the owner, the signature verification is not needed since the trusted entity has no incentive for repudiation or collusion. Therefore, delegating small part of owner's work to the TTP reduces both the storage and computation overheads. However the outsourced data must be kept private and any leakage of data toward the TTP must be prevented.

4.2 Notations

- F is a data file to be outsourced
- h is a cryptographic hash function
- K is a data encryption key/secret key
- E_K is a symmetric encryption algorithm under K, e.g., AES (advanced encryption standard)
- $E^{-1}K$ is a symmetric decryption algorithm under K
- $\tilde{F}$ is an encrypted version of the file F
- $\tilde{F}H_{TTP}$ is a hash value for $\tilde{F}$, and is computed and stored by the TTP
- $\tilde{F}H_U$ is a hash value for $\tilde{F}$, and is computed by the authorized user
- $ENC_S(K)$ is an encrypted version of secret key under S
- S is a secret shared between owner and his authorized users.

4.3 Setup and file preparation for data outsourcing

The system setup has two parts: one is done on the Owner side, and the other is done on the TTP side as shown in figure 2.

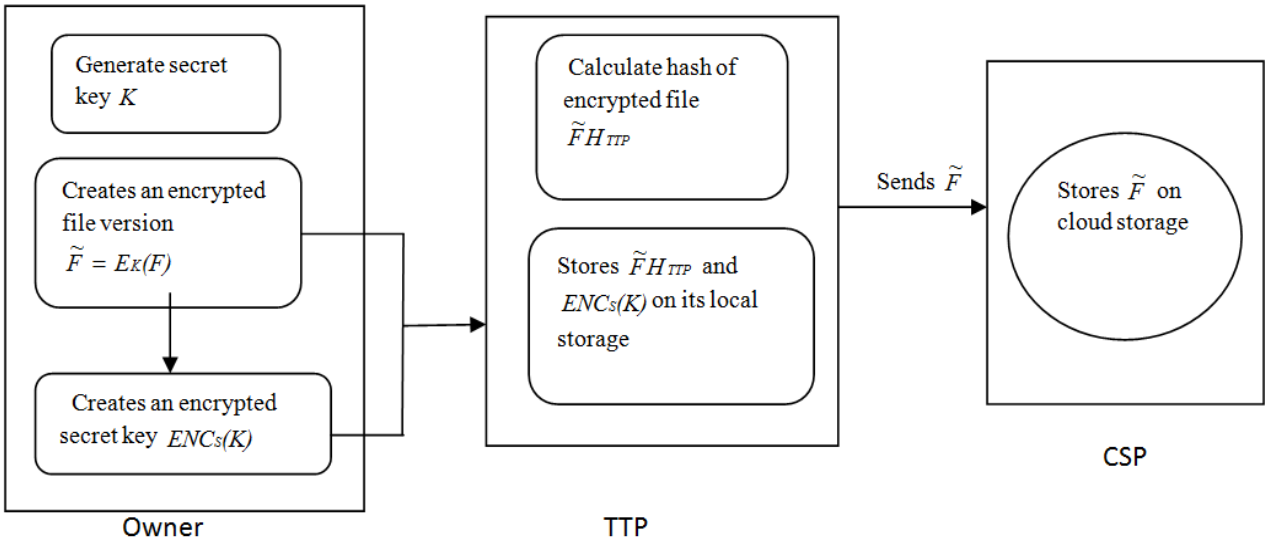


Fig 2: Setup and file preparation for data outsourcing

4.3.1 Owner role

The data owner generates a secret key K for a file. To achieve privacy-preserving, the owner creates an encrypted file version $\tilde{F} = E_K(F)$. For access control he creates encrypted secret key $ENC_S(K)$ enables only authorized users to decrypt secret key and access the outsourced file. The owner sends $\tilde{F}$ and $ENC_S(K)$ to the TTP, and deletes the data file from its local storage.

4.3.2 TTP role

A small part of the owner's work is delegated to the TTP to reduce the storage overhead and lower the overall system computation. For the TTP to resolve disputes that may arise regarding data integrity it computes and locally stores hash value for the encrypted file $\tilde{F}H_{TTP}$. The TTP sends encrypted file $\tilde{F}$ to the CSP. The TTP keeps only $\tilde{F}H_{TTP}$ and $ENC_S(K)$ on its local storage.

4.4 Data Access and Cheating Detection

An authorized user sends a data-access request to both the CSP and the TTP. The authorized user receives $\tilde{F}$ from the CSP and ($\tilde{F}H_{TTP}$, $ENC_S(K)$) from the TTP.

4.4.1 Verification of encrypted data file

The authorized user computes hash of encrypted file $\tilde{F}H_U$ received from the CSP and compare it with one received from the TTP $\tilde{F}H_{TTP}$. If $\tilde{F}H_{TTP} \neq \tilde{F}H_U$ then invoke cheating detection procedure at TTP. And if $\tilde{F}H_{TTP} = \tilde{F}H_U$ then decrypts $ENC_S(K)$ to get Secret Key K and hence decrypts file. Data Access and Verification of encrypted data file is shown in figure 3.

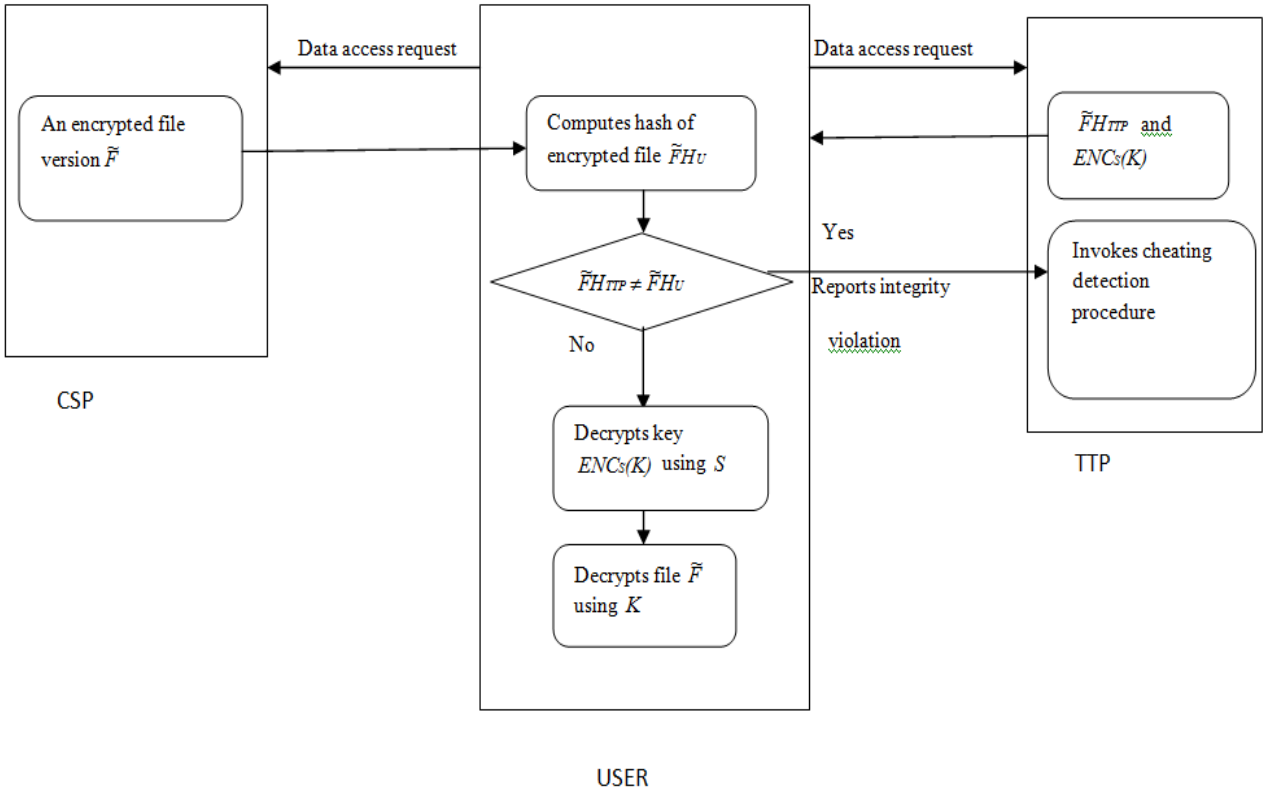


Fig 3: Data Access and Verification of encrypted data file

4.4.2 Cheating detection procedure

TTP is invoked to determine the dishonest party. The TTP receives encrypted file $\tilde{F}$ from the CSP and computes temporary hash value for encrypted file $\tilde{F}H_{temp}$. If $\tilde{F}H_{TTP} \neq \tilde{F}H_{temp}$ then reports "dishonest CSP and data is corrupted" to owner. If $\tilde{F}H_{TTP} = \tilde{F}H_{temp}$ then reports "dishonest owner/user and data is not corrupted". Cheating detection procedure is shown in figure 4.

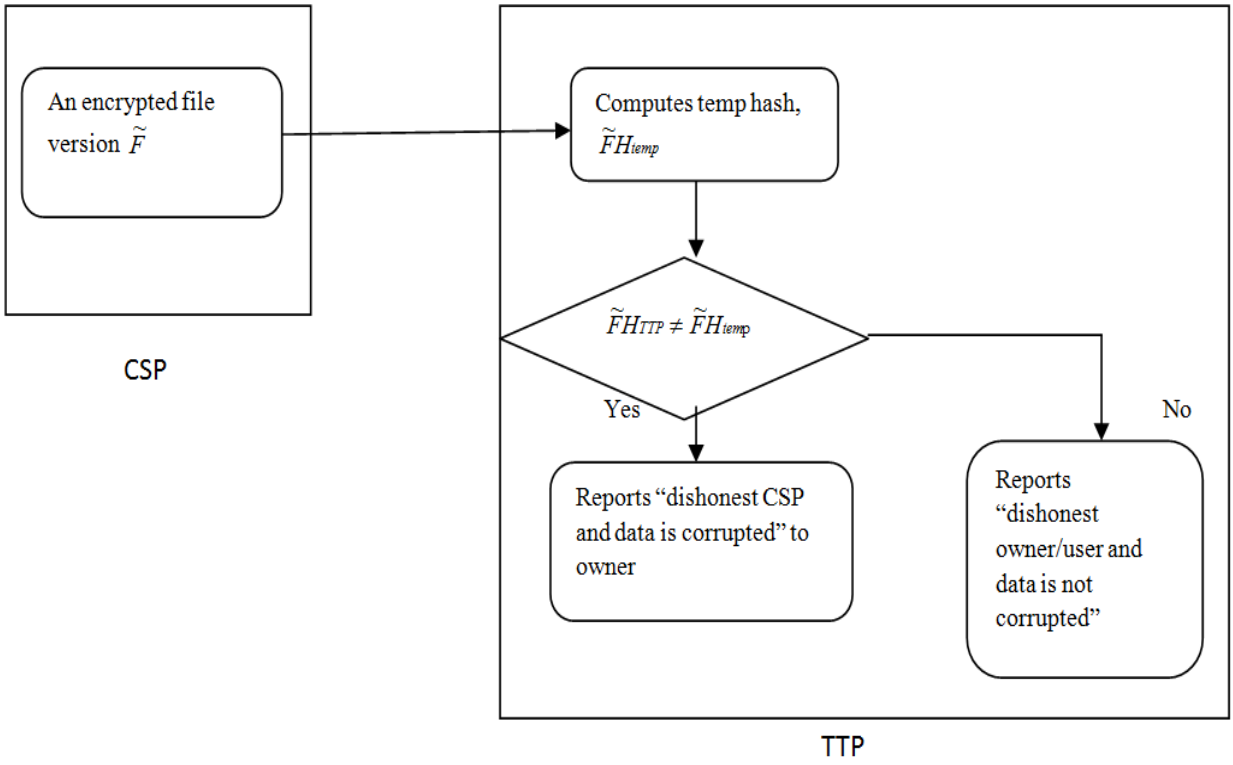


Fig 4: Cheating detection procedure

5. SECURITY ANALYSIS

5.1 Data confidentiality

The outsourced data are kept secret; the data owner creates an encrypted version of the data file $\tilde{F}$. The encryption of a file is done using a secret key K generated by owner, where K is accessed only by the data owner and the authorized users.

5.2 Detection of data integrity violation

Due to preimage and second-preimage resistance properties of the used cryptographic hash function h along with the non-collusion incentive of the TTP, the data cannot be corrupted on cloud servers without being detected. During the data access phase of the proposed scheme, the authorized user receives the encrypted file from the CSP and $\tilde{F}H_{TTP}$ from the TTP. The authorized user computes hash of encrypted file $\tilde{F}H_U$ and compares $\tilde{F}H_{TTP}$ and $\tilde{F}H_U$.

If $\tilde{F}H_{TTP} \neq \tilde{F}H_U$, then $\tilde{F}$ has been corrupted on the server. For violating data integrity without being detected there are two possible scenarios. First, the CSP has to send a file $\tilde{F}'$ where $\tilde{F}' \neq \tilde{F}$ but $h(\tilde{F}') = h(\tilde{F})$. Due to the second-preimage resistance property of h , there is no such a file $\tilde{F}'$. Second, the CSP has to generate $h(\tilde{F}')$ such that $FH_{TTP} = h(\tilde{F}')$ and $h(\tilde{F}') \neq h(\tilde{F})$. If CSP could create file $\tilde{F}'$ with $h(\tilde{F}')$, the cheating is possible. Due to the preimage-resistance property of h (one-way function), the CSP cannot generate such a file $\tilde{F}'$.

5.3 Enforcement of access control

The owner creates $ENC_S(K)$ and only authorized users can decrypt $ENC_S(K)$ using S and get K to read the outsourced data, and thus the access control is achieved in the proposed scheme.

5.4 Detection of dishonest owner/user

If the owner/user falsely accuses the CSP regarding data integrity, the TTP performs cheating detection procedure. In this procedure, TTP retrieves encrypted file from CSP and computes the temporary hash value $\tilde{F}H_{temp}$ and compares $\tilde{F}H_{TTP}$ and $\tilde{F}H_{temp}$. If $\tilde{F}H_{TTP} = \tilde{F}H_{temp}$, then $\tilde{F}$ has not been corrupted on the server and owner/ user is dishonest.

5.5 Detection of dishonest CSP

During the data access phase of the proposed scheme, the authorized user receives the encrypted file $\tilde{F}$ from the CSP and $\tilde{F}H_{TTP}$ from the TTP. The authorized user computes hash of encrypted file $\tilde{F}H_U$ and compares $\tilde{F}H_{TTP}$ and $\tilde{F}H_U$.

If $\tilde{F}H_{TTP} \neq \tilde{F}H_U$, a report is issued to TTP to determine the dishonest party. The TTP retrieves encrypted file from CSP and computes the temporary hash value $\tilde{F}H_{temp}$ and compares $\tilde{F}H_{TTP}$ and $\tilde{F}H_{temp}$. If $\tilde{F}H_{TTP} \neq \tilde{F}H_{temp}$, then $\tilde{F}$ has been corrupted on the server and CSP is dishonest.

6. IMPLEMENTATION

The proposed scheme is implemented on Eucalyptus Cloud controller and walrus storage service cloud platform. The implementation of the proposed scheme consists of three modules: OModule (owner module), UModule (user module), and TModule (TTP module).

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.

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.

TModule is a library used by the TTP to determine the cheating party in the system.

6.1 Implementation settings

In implementation, a laptop HP-630 with Intel® Core™ i5-2540M (2.60 GHz, 3 MB L3 cache) processor and 4 GB RAM running Cent OS 2.6.32-358.2.1.el6.x86_64 is used for installation of *Eucalyptus 3.2.1 cloud*.

A laptop with Intel® Pentium(R) Dual CPU T3200@2.00GHz×2 processor and 2GB RAM running Ubuntu 12.04 LTS operating system is used to execute the *UModule* and *OModule*.

A separate server in the lab is used to run *TModule*. This server has Intel® Core™ i5-2430M, 2.40 GHz processor, 4GB RAM, and Ubuntu 12.04 LTS operating system. Algorithms (hashing, Advanced Encryption standard) are implemented using crypto library.

System setup is as shown in figure 5.

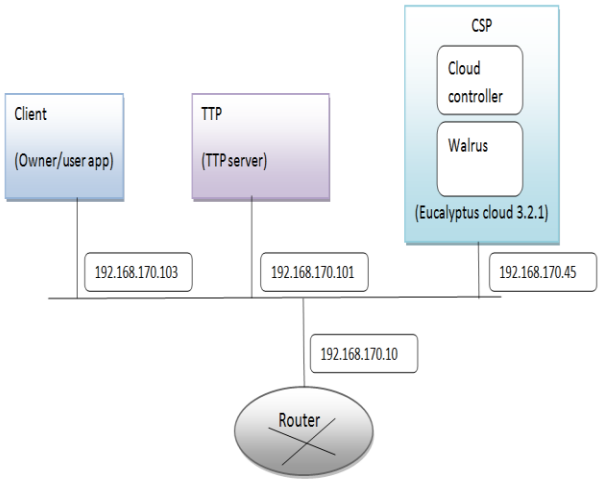


Fig 5: System setup

6.2 User Interaction with Eucalyptus-Walrus

A Eucalyptus user/client has to provide his SECRET KEY and ACCESS KEY while requesting access to buckets and objects. An architectural diagram of Walrus is given in Figure 6; user can access Walrus storage via REST/SOAP API while the cloud controller provides access control to Walrus objects using ACLs and user credentials.

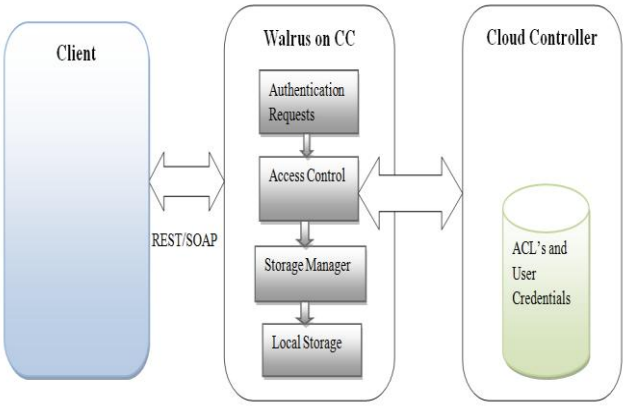
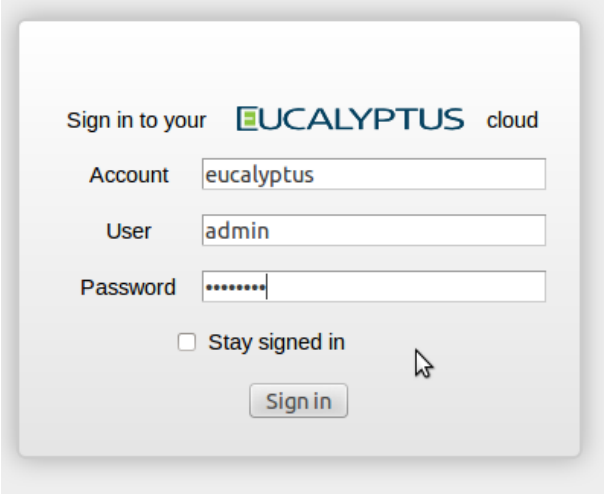


Fig 6. User Interaction with Eucalyptus-Walrus

6.3 Eucalyptus account login screen



6.4 Eucalyptus Dashboard Screen

6.5 TTP Results

6.5.1 Setup and file preparation for data outsourcing

TTP authenticates owner before receiving an encrypted file, an encrypted key and list of users who has access to file from owner. After successful authentication, TTP calculates hash of file and uploads file on cloud on behalf of owner and stores hash and an encrypted key in TTP DB.

```
Mysql client version 5.5.31
-------------Recieving encrypted file and encrypted key from itdept-------------
--2013-08-28 02:00:50--  http://192.168.170.103/semI.encrypt
Connecting to 192.168.170.103:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 328
Saving to: `semI.encrypt'

100%[=======================================================================================================>] 328         --.-K/s   in 0s

2013-08-28 02:00:50 (20.3 MB/s) - `semI.encrypt' saved [328/328]


----------------Calculating Hash on file semI--------------------
Hash is 6194d6e285229bd148a6fc83d815010ed366d91f for filename semI.encrypt

----------File semI uploading successfull on cloud s3://result/semI-------------
entering keyinfo into file_table
---------------Storing hash and  encrypted key of file semI--------
In function enterkeyinfo into file table
Query is Select *  from `file` where filename = 'semI' and bucketname = 'result' and ownername = 'itdept' ;

Mysql command is UPDATE `file` SET filename='semI',EncKey='66RU9p0jpn2zPQ==',Hash='6194d6e285229bd148a6fc83d815010ed366d91f',bucketname='result'
,ownername='itdept' WHERE filename='semI' and bucketname = 'result' and ownername ='itdept';

Mysql command is Select *  from `file` where filename = 'semI' and bucketname = 'result' and ownername = 'itdept' ;

entering user accessinfo into userfile_accesstable
Mysql client version 5.5.31
mysqlcommand is DELETE  from `user_file_access` where fileid = '44' ;

Mysql client version 5.5.31
mysqlcommand is Select *  from `client` where username = 'nida' and ownername = 'itdept' ;

mysqlcommand is INSERT into `user_file_access`  (fileid,userid) VALUES("44","1");

Mysql client version 5.5.31
mysqlcommand is Select *  from `client` where username = 'ashwini' and ownername = 'itdept' ;

mysqlcommand is INSERT into `user_file_access`  (fileid,userid) VALUES("44","3");
```

6.5.2 Data Access and Cheating Detection

6.5.2.1. User successfully downloads file.

```
accepting a new connection
before receive iByteOffsetis 0
recv returned 680
 680 bytes received
Received a message with message id 1
Client Connect message recevived
bytes processed = 600 for msgid
After process_msg bytesproce: 600 msglen 680
Received a message with message id 6
----------------Authenticating user nida----------------
Mysql client version 5.5.31
mysqlcommand is Select *  from `client` where username= 'nida' and ownername = 'itdept' ;
 and password is nida
-----------------Successfull Authentication---------------
bytes processed = 80 for msgid
After process_msg bytesproce: 80 msglen 80
before receive iByteOffsetis 0
recv returned 0
accepting a new connection
before receive iByteOffsetis 0
recv returned 600
 600 bytes received
Received a message with message id 1
Client Connect message recevived
bytes processed = 600 for msgid
After process_msg bytesproce: 600 msglen 600
before receive iByteOffsetis 0
recv returned 600
 600 bytes received
Received a message with message id 4
Get key msg received
Mysql client version 5.5.31
mysqlcommand is Select *  from `client` where username = 'nida' and ownername = 'itdept' ;

mysqlcommand is Select *  from `file` where filename = 'semI' and bucketname = 'result' ;

-----------------Checking whether user has access to file --------------------
mysqlcommand is Select *  from `user_file_access` where userid= '1' and fileid = '44' ;

----------User has access to file semI----------
-------------Sending Hash and Encrypted Key of file semI-------------
```

6.5.2.2 Cheating detection procedure is invoked.

```
----------Checking whether user has access to file ----------
mysqlcommand is Select *  from `user_file_access` where userid= '1' and fileid = '40' ;

----------User has access to file semIII----------
----------Sending Hash and Encrypted Key of file semIII----------
mysqlcommand is Select *  from `file` where ownername = 'itdept' and bucketname = 'result' and filename = 'semIII' ;

bytes processed = 600 for msgid
After process_msg bytesproce: 600 msglen 600
before receive iByteOffsetis 0
recv returned 600
 600 bytes received
Received a message with message id 5
----------Invoking cheat detection process----------
Query is Select * from owners where ownername = "nida";

Mysql client version 5.5.31
Mysql client version 5.5.31
mysqlcommand is Select *  from `client` where username= 'nida' and ownername = 'itdept' ;
 and password is nida
Mysql client version 5.5.31
mysqlcommand is Select *  from `client` where username = 'nida' and ownername = 'itdept' ;

userid is 1 for user nida, owner itdept
mysqlcommand is Select *  from `file` where filename = 'semIII' and bucketname = 'result' ;

fileid is 40, for result:semIII
mysqlcommand is Select *  from `user_file_access` where userid= '1' and fileid = '40' ;

Query is Select *  from `owners` where ownername = 'itdept' ;

----------Getting file semIII from cloud ----------
----------Calculating temp hash on file semIII----------
mysqlcommand is Select *  from `file` where ownername = 'itdept' and bucketname = 'result' and filename = 'semIII' ;

----------Cheat detection results: Integrity of file semIII is modified and CSP is dishonest----------
bytes processed = 600 for msgid
After process_msg bytesproce: 600 msglen 600
before receive iByteOffsetis 0
recv returned 0
```

7. CONCLUSION

The cloud based storage scheme is proposed that allows owner to benefit from facilities offered by the CSP and enables indirect mutual trust between them. It enables data owners to release their concerns regarding confidentiality, integrity, access control of the outsourced data. To resolve disputes that may occur regarding data integrity, a trusted third party is invoked to determine the dishonest party (owner/users or CSP).

The security features of the proposed scheme are studied, and showed that the scheme satisfies: (i) data confidentiality based on the security of underlying encryption algorithm, (ii) detection of data integrity violation based on the preimage and second-preimage resistance properties of the utilized cryptographic hash function, (iii) enforcement of access control based on, TTP gives encrypted key to only authorized users and only authorized users can decrypt this key and get the key to read the outsourced data; and (iv) detection of dishonest owner/user through a trusted third party.

8. FUTURE RESEARCH DIRECTIONS

The area of cloud computing has attracted many researchers from diverse fields; however, much effort remains to achieve the wide acceptance and usage of cloud computing technology. A number of future research directions stem from our current research. Below, we summarize some problems to address during our future research.

8.1 User authentication for cloud computing systems

The development of cloud computing encourages the use of resource-constrained devices (PDA/cell phones) on the client side. Thus, rather than local data storage and software installation, users will be authenticate to access data and use applications from the cloud. Such computing model makes software piracy more difficult and enables centralized monitoring. Although cloud computing architecture stimulates mobility of users, it increases the need of secure authentication.

Relying on passwords for user authentication is not an efficient approach for sensitive data/applications on the cloud. Passwords is a major point of vulnerability in computer security; they are often easy to guess by automated programs running dictionary attacks, users cannot remember very long passwords, and the common use of meaningful passwords makes them subject to dictionary attacks.

Implicit authentication is another interesting area of research to address user authentication problem. One can use learning algorithms to construct a model for the user based on previous behavior patterns, and then compare the recent behavior with the user model to authorize legitimate users. This may require collaboration with researchers from computer science to develop efficient learning algorithms using artificial intelligence, machine learning, and neural networks tools.

8.2 Outsourcing computation to untrusted cloud servers

Outsourcing computation is a growing desire for resource-constrained clients to benefit from powerful cloud servers. Such clients prefer to outsource computationally-intensive operations (e.g., image processing) to the cloud and yet obtaining a strong assurance that the computations are correctly performed. To save the computational resources, a dishonest CSP may totally ignore the computations, or execute just a portion of them. Sometimes the computations

outsourced to the cloud are so critical that it is essential to preclude accidental errors during the processing.

The ability to verify computations and validate the returned results is a key requirement of cloud customers. Another imperative point is that the amount of work performed by the clients to verify the outsourced computations must be substantially cheaper than performing the actual computations on the client side. One direction of future research is to investigate the area of verifiable computations and outsourcing computational tasks to untrusted cloud servers. It is also interesting to address mutual trust feature, so a client who receives incorrect results from cloud servers can detect and prove this misbehaviour. Moreover, a dishonest client must not be able to falsely accuse a CSP and claim that the outsourced computations are malformed.

9. REFERENCES

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

[2] F. Seb´e, 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.

[3] 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.

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

[5] 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.

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

[7] H. Shacham and B. Waters, "Compact proofs of retrievability," Cryptology ePrint Archive, Report 2008/073, 2008, http:// eprint.iacr.org/.

[8] 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.

[9] 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

[10] 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.

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

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in Proceedings of the 13th ACM conference on Computer and communications security, ser. CCS '06. ACM, 2006, pp. 89–98.

[13] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in Proceedings of the 29th conference on Information communications, ser. INFOCOM'10. IEEE Press, 2010, pp. 534–542.

[14] 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

[15] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in EUROCRYPT, 1998, pp. 127–144.

[16] 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.

[17] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in Proceedings of the 2009 ACM workshop on Cloud computing security, ser. CCSW '09. ACM, 2009, pp. 55–66.