International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 - 5161

Available at http://inpressco.com/category/ijcet

Review Article

# A Survey of Various Security Management Models for Cloud Computing Storage Systems

Yasha Gawande[Á*], Anuradha Jagadale[Á] and Shilpa Gite[Á]

[Á]Department of Computer Science, Symbiosis Institute of Technology, SIU, Pune, India

Accepted 20 June 2014, Available online 30 June 2014, **Vol.4, No.3 (June 2014)**

***Abstract***

*Cloud computing is an emerging technology and it is internet based computing, where shared resources, software and information, are provided to clients. Using Storage-as-a-Service, users and organizations can store their data remotely which poses new security risks towards the correctness of data in cloud. In order to achieve secure cloud storage, there exist different techniques such as storage auditing mechanism, trusted third party technique, PDP, CPDP, Plutus, SiRiUS etc. These techniques support secure and efficient dynamic data storage in the cloud. In this paper, we discuss important security issues related to storage and present a comprehensive survey of the security services provided by the existing storage systems.*

***Keywords:*** *cloud computing, CSP, owner, data correctness, distributed data integrity, auditing, security, access control, mutual trust, confidentiality, privacy.*

## 1. Introduction

Cloud storage can provide on-demand, scalable and Quality of Service (QoS) guaranteed storage resource, and users can operate their data anytime and anywhere. Cloud computing is a system whereby data storage, applications and to some degree processing power are freed from local constraints and are available via servers or computers elsewhere in the world. The need for huge local drives is negated, those space-hogging office suites no longer sit on local machines and the option of an extra power boost when needed is within our reach. Facing the powerful and appealing advantages of cloud storage, there are certain problems in existing cloud computing which is impending the fast growth of cloud computing technology, among them few causes are Data security, Data confidentiality and Data access control. So a lot of people and companies are hesitant to put their data in cloud. The main reason is that people and companies are afraid of loss of control on their data. And there are some incidents of data leakage and losing which verify people's fears.

To protect stored data, it is not sufficient to use traditional network security techniques that are used for securing messages between pairs of users or between clients and servers. This paper presents a survey of techniques for securely storing data, including theoretical approaches, prototype systems, and existing systems currently available. This paper provides an overview of the prominent characteristics of several systems like Plutus, Sirius, PDP, CPDP, Auditing systems & protocols (M. Kallahalla *et al*, 2003; E. J. Goh *et al*, 2003; G. Ateniese, 2007; Y. Zhu *et al*, 2012; Kan Yang *et al*, 2013; Ayad Barsoum *et al*, 2013). Secure file systems like Plutus and SiRiUS, which strives to provide strong security even with an un-trusted server. Plutus maintains key distribution in decentralized manner and also data is stored in encryption format. Cryptographic schemes maintained by users rather than servers. The disadvantages are replication leads to over storage, complete key distribution is insecure, cryptographic operations maintained by user is not safe. . But placing complete key in user system may not be a secure one thus key distribution used.

SiRiUS is a system which establishes a secure file sharing environment without significantly modifying the performance of an existing network storage medium. SiRiUS can provide security to an existing system without requiring any hardware modifications. Often times, organizations cannot afford to upgrade their current systems and must continue to operate with limited security until which time the option to upgrade security measures becomes available; SiRiUS can provide an interim solution. Provable data possession (PDP) is such a probabilistic proof technique for a storage provider to prove the integrity and ownership of clients' data without downloading data. The proof-checking without downloading makes it especially important for large-size files and folders (typically including many clients' files) to check whether these data have been tampered with or deleted without downloading the latest version of data. Various PDP schemes have been recently proposed, such as Scalable PDP (Y. Zhu *et al*, 2012) and Dynamic PDP (Kan Yang *et al*, 2013). However, these schemes mainly focus on PDP issues at untrusted servers in a single cloud storage provider and are not suitable for a multicloud

*Corresponding author: **Yasha Gawande**

environment. Whereas a Cooperative PDP is a system which is built over PDP to support multicloud environment.

Most existing secure storage solutions require the creators of data to trust the storage server to control all users' access to this data as well as return the data intact. So solution on this is provided in (Kan Yang *et al*, 2013) (Ayad Barsoum *et al*, 2013) by introducing a trusted third party or an auditor which will audit the data storage after a regular interval of time and maintain the log of data on its side for same purpose.

## 2. Cloud Computing Principles

Cloud computing plays a prominent role in this modern era due to its compelling benefits and services. Cloud computing has become hot issue since 2007 and many companies used to attempt to use the cloud computing services. Typical cloud computing services are Amazon EC2, Google's Google app engine, Microsoft, Yahoo etc

Cloud Computing means "Internet based Computing." It is a technology that uses the internet and central remote servers to maintain data and application. Cloud storage system is a collection of cloud servers which provide continuous access. Data storage is the one which grabs a tremendous part in technology of world. Computing is a general term for anything that involves delivering hosted services over the Internet. In Figure1 the services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform as-a-Service (PaaS) and Software-as-a-Service (Saas).

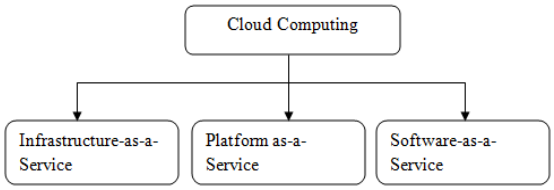


**Fig.1** Cloud computing delivery models

Infrastructure as a service sometimes referred as HaaS or hardware as a service. IaaS includes storage, hardware, servers and networking components. IaaS Provides user computing resources and storage comprised with many servers as an on demand and "pay per use" service: Data Center, Bandwidth, Private Line Access, Servers and Server Room, Firewall PaaS model provides a platform for creating applications. PaaS solutions are essentially development platforms for which the development tool itself is hosted in the Cloud and accessed through a browser. With PaaS, developers can build Web applications without installing any tools on their computers. PaaS bundles all stack components (hardware, infrastructure, storage) together with database, security, workflow, user interface, and other tools that allow users to create and host powerful business applications, web sites, and mobile apps. In the Saas model, cloud providers install and operate application software in the cloud and cloud users access the software from clients. Saas can be defined through five key ideas: services are fully managed and hosted, have regular recurring payments, allow for anytime and anywhere access, have multiple tenants on servers, don't require installation of specialized software.

Cloud computing deployment models are classified into four categories: public cloud, private cloud, hybrid cloud, community cloud as shown in Figure2. Public cloud computing environment are open for use to anyone who wants to sign up and use them. A private cloud is basically an organization that needs more control over their data than they can get by using a vendor hosted service. A hybrid cloud combines both public and private cloud models.

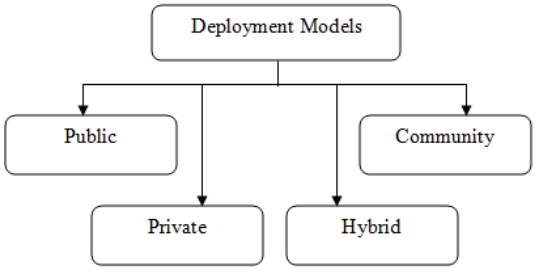


**Fig.2** Cloud computing deployment model

### *2.1 Characteristics*

A cloud service should have five essential characteristics:

- *On-demand Self-service*: Managers can obtain services as and when needed, without human intervention.
- *Broad Network Access*: Users can access services using a broad range of network client devices, such as browsers, mobile phones, tablets, laptops, and workstations.
- *Resource Pooling*: Resources are not dedicated to particular consumers but are pooled and allocated as required.
- *Rapid Elasticity*: The amount of resource allocated to a consumer can be expanded and contracted easily and quickly to meet demand.
- *Measured Service*: The amount of resource consumed is measured, to enable automatic expansion and contraction, and to give visibility of usage to providers and consumers.

## 3. Security properties of cloud

### *3.1 Data integrity*

Data integrity is defined as the accuracy and consistency of stored data, in absence of any alteration to the data between two updates of a file or record. Cloud services should ensure data integrity and provide trust to the user privacy. Cloud computing poses privacy concerns primarily, because the service provider at any point in time, may access the data that is on the cloud. The Cloud service provider could accidentally or deliberately alter or delete some information from the cloud server. Hence, the system must have some sort of mechanism to ensure the data integrity. The current Cloud security model is based

on the assumption that the user/customer should trust the provider. This is typically governed by a Service Level Agreement (SLA) that in general defines mutual provider and user expectations and obligations.

*3.2 Confidentiality*

The confidentiality of a system is guaranteed providing it prevents unauthorized gathering of information. In data secure systems, the "confidentiality" characteristic requires authorizations and checks to be defined, to ensure that information cannot be accessed by subjects who do not have corresponding rights. It must be possible to assign and withdraw the rights that are necessary to process this data, and checks must be implemented to enforce compliance. Cryptographic techniques and access controls based on strong authentication are normally used to protect confidentiality.

*3.3 Authenticity*

The authenticity of a subject or object is defined as its genuineness and credibility; these can be verified on the basis of its unique identity and characteristic features. Information is authentic if it can be reliably assigned to the sender, and if it can be proved that this information has not been changed since it was created and distributed. A secure technique for identifying the communication partners and mechanisms for ensuring authenticity are essential here. These mechanisms must be capable of confirming or disproving the authenticity of the protected information. None of the system participants can create or distribute messages and data on behalf of another subject.

*3.4 Storage correctness*

It is to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud.

*3.5 Newness property*

Receiving the most recent version of the outsourced. Data file is an imperative requirement of cloud-based storage systems. Newness property ensures that the authorized users receive the most recent version of the Data. There must be a detection mechanism if the cloud provider ignores any data-update requests issued by the owner.

*3.6 Cloud portability*

Cloud portability means the ability to move applications and its associated data between one cloud provider and another or between public and private cloud environments.

## 4. Threat Model

In this paper, we are considering two types of attacks for cloud data storage those are: Internal Attacks and External Attacks.

*4.1 Internal Attacks*

These are initiated by malicious Cloud Service Provider (CSP) or malicious users. Those are intentionally corrupting the user's data inside the cloud by modifying or deleting. They are also able obtain all the information and may leaked it to outsiders.

*4.2 External Attacks*

These are initiated by unauthorized parties from outside the cloud. The external attacker, who is capable of comprising cloud servers and can access the user's data as long as they are internally consistent i.e. he may delete or modify the customer's data and may leaked the user private information.

Besides this, the CSP is untrusted, and thus the confidentiality and integrity of data in the cloud may be at risk. 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 the misuse of confidential data (Ayad Barsoum *et al*, 2013).

On the other hand, a data owner and authorized users may collude and falsely accuse the CSP that data integrity over cloud servers has been violated, or it may happen the CSP has returned a stale file that does not match the most recent modifications issued by the owner(Ayad Barsoum *et al*, 2013). At certain times sever could be dishonest and may launch the following attacks:

*Replace attack*: The server may choose another valid and uncorrupted pair of data block and data tag to replace the challenged pair of data block and data tag, when it already discarded data block or data tag.

*Forge attack*: The server may forge the data tag of data block and deceive the auditor; if the owner's secret tag keys are reused for the different versions of data. Tag Forgery Attack by which a dishonest CSP can deceive the clients.

*Replay attack*: The server may generate the proof from the previous proof or other information, without retrieving the actual owner's data.

*Data Leakage Attack*: By which an adversary can easily obtain the stored data through verification process after running or wiretapping sufficient verification communications.

*Denial of service*: an attacker capable of compromising the remote server can delete files.

*Rollback Attack*: a rollback attack replaces the newest version of the file with an older version. A rollback attack involves misleading users into accessing stale data.

Also, there are several attacks like data leakage attacks on the physical device, such as by an untrusted administrator, a stolen laptop, or a compromised server; and attacks may be on public and private keys so there should be a way by which users can set arbitrary policies for key distribution (and therefore file sharing). These attacks may cause potential risks for privacy leakage and ownership cheating. Also, these attacks can more easily compromise the security of a distributed cloud system than that of a single cloud system.

## 5. TPA- new approach towards securing cloud data storage

Traditionally, owners can check the data integrity based on two-party storage auditing protocols. In cloud storage system, however, it is inappropriate to let either side of cloud service providers or owners conduct such auditing, because none of them could be guaranteed to provide unbiased auditing result. In this situation, third party auditing is a natural choice for the storage auditing in cloud computing. A third party auditor that has expertise and capabilities can do a more efficient work and convince both cloud service providers and owners (Kan Yang *et al*, 2013).

## 6. Various schemes for secure data storage in cloud

### *6.1 PDP*

Verifying the authenticity of data has emerged as a critical issue in storing data on untrusted servers. It arises in peer-to-peer storage systems (J. Kubiatowicz *et al*, 2000; A. A. Muthitacharoen *et al*, 2002), network file systems (J. Li *et al*, 2004; M. Kallahalla *et al*, 2003), long-term archives (P.Maniatis *et al*, 2005), web-service object stores (A. Y. Yumerefendi *et al*, 2007), and database systems (U. Maheshwari *et al*, 2000). Such systems prevent storage servers from misrepresenting or modifying data by providing authenticity checks when accessing data. However, archival storage requires guarantees about the authenticity of data on storage, namely that storage servers possess data. It is insufficient to detect that data have been modified or deleted when accessing the data, because it may be too late to recover lost or damaged data. Archival storage servers retain tremendous amounts of data, little of which are accessed. They also hold data for long periods of time during which there may be exposure to data loss from administration errors as the physical implementation of storage evolves, e.g., backup and restore, data migration to new systems, and changing memberships in peer-to-peer systems. Archival network storage presents unique performance demands. Given that file data are large and are stored at remote sites, accessing an entire file is expensive in I/O costs to the storage server and in transmitting the file across a network. Reading an entire archive, even periodically, greatly limits the scalability of network stores. Furthermore, I/O incurred to establish data possession interferes with on-demand bandwidth to store and retrieve data. We conclude that clients need to be able to verify that a server has retained file data without retrieving the data from the server and without having the server access the entire file.

Previous solutions do not meet these requirements for proving data possession. Some schemes (P. Golle *et al*, 2002) provide a weaker guarantee by enforcing storage complexity. The server has to store an amount of data at least as large as the client's data, but not necessarily the same exact data. Moreover, all previous techniques require the server to access the entire file, which is not feasible when dealing with large amounts of data.

We define a model for provable data possession (PDP) that provides probabilistic proof that a third party stores a file. The model is unique in that it allows the server to access small portions of the file in generating the proof; all other techniques must access the entire file. Within this model, we give the first provably-secure scheme for remote data checking. The client stores a small O(1) amount of metadata to verify the server's proof. Also, the scheme uses O(1) bandwidth1. The challenge and the response are each slightly more than 1 Kilobit. We also present a more efficient version of this scheme that proves data possession using a single modular exponentiation at the server, even though it provides a weaker guarantee.

Both schemes use homomorphic verifiable tags. Because of the homomorphic property, tags computed for multiple file blocks can be combined into a single value. The client pre-computes tags for each block of a file and then stores the file and its tags with a server. At a later time, the client can verify that the server possesses the file by generating a random challenge against a randomly selected set of file blocks. Using the queried blocks and their corresponding tags, the server generates a proof of possession. The client is thus convinced of data possession, without actually having to retrieve file blocks.

*PDP Definition:* A PDP protocol checks that an outsourced storage Site retains a file, which consists of a collection of n blocks. The client C (data owner) pre-processes the file, generating a piece of metadata that is stored locally, transmits the file to the server S, and may delete its local copy. The server stores the file and responds to challenges issued by the client. Storage at the server is in (n) and storage at the client is in O (1), conforming to our notion of an outsourced storage relationship.

Finally, PDP focused on the problem of verifying if an untrusted server stores a client's data. We introduced a model for provable data possession, in which it is desirable to minimize the file block accesses, the computation on the server, and the client-server communication. Our solutions for PDP fit this model: They incur a low (or even constant) overhead at the server and re quire a small, constant amount of communication per challenge. Key components of our schemes are the homomorphic verifiable tags. They allow verifying data possession without having access to the actual data file.

### *6.2 CPDP*

Even though existing PDP schemes have addressed various security properties, such as public verifiability, dynamics, scalability, and privacy preservation, these models still cannot cover all security requirements, especially for provable secure privacy preservation and ownership authentication, we still need a careful consideration of some potential attacks, including two major categories: Data Leakage Attack & Tag Forgery Attack.

Cooperative PDP model is developed to reduce the storage and network overheads and enhance the transparency of verification activities in cluster based cloud storage systems. Moreover, such cooperative PDP scheme provides features for timely detecting abnormality and renewing multiple copies of data.

Possibility of constructing a cooperative PDP (CPDP) scheme without compromising data privacy based on modern cryptographic techniques, such as interactive proof system. This construction is a multiprover zero-

knowledge proof system (MP-ZKPS) (L. Fortnow *et al*, 1988), which has completeness, knowledge soundness, and zero-knowledge properties. These properties ensure that CPDP scheme can implement the security against data leakage attack and tag forgery attack. Finally, our experiments show that our solution introduces very limited computation and communication overheads.

Definition of Cooperative PDP: In order to prove the integrity of data stored in a multicloud environment, defined a framework of CPDP based on IPS and multiprover zero-knowledge proof system (MPZKPS), as follows:

Cooperative-PDP- A cooperative provable data possession KeyGen; TagGen; Proof is a collection of two algorithms (KeyGen; TagGen) and an interactive proof system Proof.

To support distributed cloud storage, representative architecture used in cooperative PDP scheme. This architecture has a hierarchy structure which resembles a natural representation of file storage. This hierarchical structure H consists of three layers to represent relationships among all blocks for stored resources. They are described as follows:

*Express layer*: Offers an abstract representation of the stored resources;

*Service layer*: Offers and manages cloud storage services; and

*Storage layer*: Realizes data storage on many physical devices.

Cooperative PDP scheme is making use of this simple hierarchy to organize data blocks, from multiple CSP services into a large-size file by shading their differences among these cloud storage systems. The proposed structure can be readily incorporated into MAC-based, ECC, or RSA schemes (G. Ateniese *et al*, 2007; H. Shacham *et al*, 2008). These schemes, built from collision-resistance signatures and the random oracle model, have the shortest query and response with public verifiability. They share several common characters for the implementation of the CPDP framework in the multiple clouds:

- A file is split into $n_s$ sectors and each block (s sectors) corresponds to a tag, so that the storage of signature tags can be reduced by the increase of s;
- A verifier can verify the integrity of file in random sampling approach, which is of utmost importance for large files;
- These schemes rely on homomorphic properties to aggregate data and tags into a constant-size response, which minimizes the overhead of network communication; and
- The hierarchy structure provides a virtualization approach to conceal the storage details of multiple CSPs.

In CPDP Scheme, the manager first runs algorithm KeyGen to obtain the public/private key pairs for CSPs and users. Then, the clients generate the tags of outsourced data by using TagGen. Anytime, the protocol Proof is performed by a five-move interactive proof protocol between a verifier and more than one CSP, in which CSPs need not to interact with each other during the verification process, but an organizer is used to organize and manage all CSPs. This protocol can be described as follows:

- The organizer initiates the protocol and sends a commitment to the verifier;
- The verifier returns a challenge set of random index coefficient pairs Q to the organizer;
- The organizer relays them into each Pi in P according to the exact position of each data block;
- Each Pi returns its response of challenge to the organizer; and
- The organizer synthesizes a final response from received responses and sends it to the verifier.

The above process would guarantee that the verifier accesses files without knowing on which CSPs or in what geographical locations their files reside.

CPDP for Integrity Audit Services is based on CPDP scheme, introducing audit system architecture for outsourced data in multiple clouds by replacing the TTP with a third party auditor (TPA). Finally CPDP is an efficient PDP scheme for distributed cloud storage. Based on homomorphic verifiable response and hash index hierarchy, proposed cooperative PDP scheme support dynamic scalability on multiple storage servers and provided all security properties required by zero-knowledge interactive proof system, so that it can resist various attacks even if it is deployed as a public audit service in clouds. Therefore, this solution can be treated as a new candidate for data integrity verification in outsourcing data storage systems.

### *6.3 Plutus*

The primary goal of Plutus is to provide highly scalable key management while providing file owners with direct control over authorizing access to their files (M. Kallahalla *et al*, 2003). It provides secure file sharing while placing minimal trust on the storage server.

All data is encrypted on the disk with the cryptographic and key management operations performed by the clients to alleviate server cryptographic overhead. Users can customize security policies and authentication mechanisms for their own files using the client-based key distribution scheme. This places the responsibility for key management on the user, forcing the file owner to ensure proper secure distribution of keys to those they wish to authorize access. It provides an elegant revocation mechanism and reader-writer distinction similar to SiRiUS. In addition, it provides confidentiality and integrity of network messages (RPCs) sent between the client and the server. A prototype of Plutus is built on OpenAFS. In Plutus, all files with identical sharing attributes are grouped in the same file-group. A file-group is a group of files with identical sharing attributes.

A unique symmetric key called the lockbox − key is associated with every file-group. A unique 3DES (A. J. Menezes *et al*, 1996) symmetric key, called a file − block key is used to encrypt each block of a file. A lockbox securely holds the keys for all the blocks of the files (file−block keys) belonging to one file-group by encrypting all the file−block keys using the symmetric lockbox−key. If a user wants to share his files with other

users, he creates a file-group (which contains permissions for the file-group and a list of members) and a lockbox−key, which has to be distributed by the creator of the file-group to the members of the file-group.

Each file is encrypted block by block using the respective file − block key, which is automatically created during block creation. Associated with each file-group (lockbox) is an RSA key pair where the private part of the pair is the file sign-key and the public part is the file verify-key. The readers are given the lockbox-keys whereas the writers are given the lockbox-keys as well as the file sign-keys. Thus, similar to SiRiUS, in Plutus possession of signing key distinguishes writers from readers. The server authenticates the writers before allowing writes by using the write token associated per file-group, which is given by the owner to all writers of that file-group. This prevents someone from simply overwriting the entire storage space on the storage server.

Plutus relies heavily on file-groups to limit the number of cryptographic keys. This allows users with file-group privileges to access a file within the group even if the owner is not on-line, avoiding the requirement for a user to contact the owner directly to get the key with every file access. The entire design of Plutus is intended to provide scalability. Placing the key management responsibility on the clients instead of on a trusted server prevents a server bottleneck due to computationally expensive cryptographic operations.

### *6.4 SiRiUS – Securing Remote Untrusted Storage*

SiRiUS (E. Goh, H. Shacham *et al*, 2003) is a user-level file system designed to be layered over insecure network and peer-to-peer file systems such as NFS, CIFS, OceanStore, and Yahoo! Briefcase. It provides its own read-write cryptographic access control for file level sharing in small groups. The main goal was to design and implement a security mechanism that improves the security of a networked file system without making any changes to the file server. In addition to confidentiality of data, SiRiUS also ensures integrity of data, and loose integrity of meta-data.

Security SiRiUS is a user level file system implemented on Linux over NFSv3. All files are stored on the server in two parts, namely a data file (d − file), which contains the encrypted data and a Meta data file (md − file), which contains the access control information. Storing access control information in a separate meta-data files allows SiRiUS to run on top of any storage server as long as the SiRiUS client can interact with the server according to the server's semantics. The d − file is encrypted using a symmetric key and signed by the writer. Stored in each directory is the meta-data freshness file (mdf − file), which contains the root of the hash tree (R. Merkle, 1987) of the mdf − files associated with sub-directories. The root of the hash tree is signed periodically by the owner to ensure freshness of the metadata. Files are encrypted using AES in counter mode (M. Dworkin, 2001). The hash tree is built by hashing each md−file with SHA-1 (R. Merkle, 1987) and concatenating with the md−files of each subdirectory. The owner's SiRiUS client will periodically time stamp the root mdf−file and sign it using his private key.

SiRiUS is secure and provides secure file sharing with confidentiality and integrity of data. It does not consider privacy of meta-data. It ensures integrity of meta-data upto the time when the meta-data was last signed by the owner (since owner signs periodically). In addition, it prevents roll-back of md−files. All this is achieved without any modifications on the file server, a prudent design choice. Therefore, SiRiUS can be used on any of the existing file servers.

### *6.5 An Efficient and Secure Dynamic Auditing protocol (TPA)*

In cloud computing, data owners host their data on cloud servers and users (data consumers) can access the data from cloud servers. Due to the data outsourcing, however, this new paradigm of data hosting service also introduces new security challenges, which requires an independent auditing service to check the data integrity in the cloud. Some existing remote integrity checking methods can only serve for static archive data and, thus, cannot be applied to the auditing service since the data in the cloud can be dynamically updated. Thus, an efficient and secure dynamic auditing protocol is desired to convince data owners that the data are correctly stored in the cloud. In this system, the design of an auditing framework for cloud storage systems is given and proposed an efficient and privacy-preserving auditing protocol. Then, further extending this auditing protocol to support the data dynamic operations, which is efficient and provably secure in the random oracle model. This is extended to support batch auditing for both multiple owners and multiple clouds, without using any trusted organizer.

*CPDP drawback*: CPDP scheme cannot support the batch auditing for multiple owners. Because parameters for generating the data tags used by each owner are different, and thus, they cannot combine the data tags from multiple owners to conduct the batch auditing. Another drawback is that their scheme requires an additional trusted organizer to send a commitment to the auditor during the multi cloud batch auditing, because their scheme applies the mask technique to ensure the data privacy. However, such additional organizer is not practical in cloud storage systems.

In this scheme, proposed an efficient and secure dynamic auditing protocol, which can meet the above listed requirements. To solve the data privacy problem, method used is to generate an encrypted proof with the challenge stamp by using the Bilinearity property of the bilinear pairing, such that the auditor cannot decrypt it but can verify the correctness of the proof. Without using the mask technique, this method does not require any trusted organizer during the batch auditing for multiple clouds. On the other hand, in proposed method, let the server compute the proof as an intermediate value of the verification, such that the auditor can directly use this intermediate value to verify the correctness of the proof. Therefore, this method can greatly reduce the computing loads of the auditor by moving it to the cloud server.

The original contributions can be summarized as follows:

- Designing an auditing framework for cloud storage systems and propose a privacy-preserving and efficient storage auditing protocol. This auditing protocol ensures the data privacy by using cryptography method and the Bilinearity property of the bilinear pairing. This auditing protocol incurs less communication cost between the auditor and the server. It also reduces the computing loads of the auditor by moving it to the server.
- Further extending this auditing protocol to support the data dynamic operations, which is efficient and provably secure in the random oracle model.
- Further extending this auditing protocol to support batch auditing for not only multiple clouds but also multiple owners. Here multicloud batch auditing does not require any additional trusted organizer. The multiowner batch auditing can greatly improve the auditing performance, especially in large-scale cloud storage systems.

The main challenge in the design of data storage auditing protocol is the data privacy problem. This is because: 1) for public data, the auditor may obtain the data information by recovering the data blocks from the data proof. 2) For encrypted data, the auditor may obtain content keys somehow through any special channels and could be able to decrypt the data. To solve the data privacy problem, proposed method is to generate an encrypted proof with the challenge stamp by using the bilinearity property of the bilinear pairing, such that the auditor cannot decrypt it, but the auditor can verify the correctness of the proof without decrypting it. To improve the performance of an auditing system, we apply the data fragment technique and homomorphic verifiable tags in our method. The data fragment technique can reduce number of data tags, such that it can reduce the storage overhead and improve the system performance.

This storage auditing protocol consists of the following algorithms:

KeyGen $(\lambda) \rightarrow (skh, skt, pkt)$, TagGen $(M, skt, skh) \rightarrow T$, Chall $(Minfo) \rightarrow C$, Prove $(M, T, C) \rightarrow P$, Verify $(C, P, skh, pkt, Minfo) \rightarrow 0/1$.

Proposed storage auditing protocol consists of three phases: owner initialization, confirmation auditing, and sampling auditing. During the system initialization, the owner generates the keys and the tags for the data. After storing the data on the server, the owner asks the auditor to conduct the confirmation auditing to make sure.

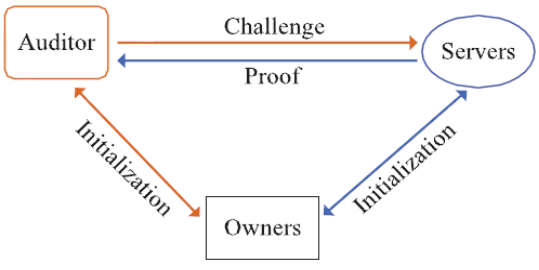


**Fig.3** System model of data storage auditing (Kan Yang *et al*, 2013)

For securing the dynamic auditing, to prevent the replay attack, introduced an index table (ITable) to record the abstract information of the data. The ITable consists of four components: Index, Bi, Vi, and Ti. The Index denotes the current block number of data block mi in the data component M. Bi denotes the original block number of data block mi, and Vi denotes the current version number of data block mi. Ti is the time stamp used for generating the data tag. This ITable is created by the owner during the owner initialization and managed by the auditor. When the owner completes the data dynamic operations, it sends an update message to the auditor for updating the ITable that is stored on the auditor. After the confirmation auditing, the auditor sends the result to the owner for the confirmation that the owner's data on the server and the abstraction information on the auditor are both up-to-date. This completes the data dynamic operation. To deal with the forge attack, modified the tag generation algorithm TagGen. Specifically, when generating the data tag ti for the data block mi, we insert all the abstract information into the data tag by setting Wi=FID||Bi||Vi||Ti, such that the server cannot get enough information to forge the data tag from dynamic operations.

Finally, proposed system is efficient and inherently secure dynamic auditing protocol. It protects the data privacy against the auditor by combining the cryptography method with the bilinearity property of bilinear paring, rather than using the mask technique. Thus, our multicloud batch auditing protocol does not require any additional organizer. Our batch auditing protocol can also support the batch auditing for multiple owners.

### *6.6 TTP for data storage security*

The local management of such huge amount of data is problematic and costly due to the requirements of high storage capacity and qualified personnel. Since the data owner physically releases sensitive data to a remote CSP, and CSP and data storage is untrusted, there are some concerns regarding access control, newness, integrity, and confidentiality of the data

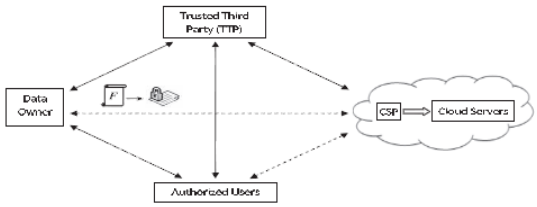


**Fig.4** Mutual trust model for cloud storage (Ayad Barsoum *et al*, 2013)

To maintain trust and to reduce the computational task in the owner and CSP side, the trusted thread party (TTP) is introduced to reduce generation of block tags and signature verification as shown in fig.4. The trust relation between components is shown by solid arrows while distrust relation is denoted by doted arrows. 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. The confidentiality feature can be guaranteed by the owner via encrypt the data file before outsourcing to remote servers. This paper presents a feasible solution that has been presented to enable the owner to enforce access control of the data stored on a remote untrusted CSP by using a trusted third party.

Through this solution, the data is encrypted under a key, which is shared only with the authorized users of data. 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 which aim at providing data storage security on untrusted remote servers. 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. In this work, proposed scheme addresses important issues related to outsourcing the storage of data, namely dynamic data, mutual trust, newness, and access control. The remotely stored data can be updated and scaled by authorized users, and owner. After updating, authorized users should receive the latest version of the data (newness property), that is a technique is required to detect whether the received data is stale. Mutual trust between the CSP and the data owner is another important issue, which is addressed in the proposed scheme. A mechanism called cheating detection procedure is introduced to determine the dishonest party. Last but not least, the access control is considered, which allows the owner to grant or revoke access rights to the outsourced data.

**Table 1** Comparison of schemes

| Scheme | Privacy | Dynamic | Hash Function | Mechanism Used | Authenticated Third Party |
|---|---|---|---|---|---|
| PDP | Yes | No | No | Setup, challenge | No |
| CPDP | No | No | No | key-gen,tag-gen, proof,verification | No |
| PLUTUS | No | No | Yes | lazy-revocation, key rotation | No |
| SIRIUS | No | No | Yes | key revocation, back-ups | No |
| TPA | Yes | Yes | Yes | key-gen,tag-gen, challenge,proof | Yes |
| TTP | Yes | Yes | Yes | lazy-revocation, key rotation,bENC, cheating detection | Yes |

## Conclusions

There is outsourcing of data over the cloud service provider. Thus there are serious concerns about the cloud storage systems, so there are various schemes have been introduced. These models are about trust and security for the cloud storage systems. In this paper, we have studied different models for cloud-based storage schemes. And finally model given by (Kan Yang *et al*, 2013; Ayad Barsoum *et al*, 2013) which supports outsourcing of dynamic data. Also, we have listed the basic security properties that should be provided by an ideal secure storage system. However, it is evident that for all practical purposes it is difficult to build a storage system that can satisfy all the listed properties.

## References

M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, Plutus (2003): Scalable Secure File Sharing on Untrusted Storage, Proc. Second USENIX Conf. File Storage Technologies, Vol. 24, NO. 12.

E.J. Goh, H. Shacham, N. Modadugu, and D. Boneh, SiRiUS (2003): Securing Remote entrusted Storage, Proc. Network Distributed System Security Symp.

G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song (2007), Provable Data Possession at Untrusted Stores, Proc. 14th ACM Conf.

Y. Zhu, H. Hu, G. Ahn, and M. Yu (Dec. 2012), Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage, IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 12,pp. 2231-2244.

Kan Yang, Student Member, IEEE, and Xiaohua Jia (Sept 2013), An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing, IEEE transactions on parallel and distributed systems, vol. 24, no. 9

Ayad Barsoum and Anwar Hasan (December 2013), Enabling Dynamic Data and Indirect Mutual Trust for Cloud Computing Storage Systems, IEEE transactions on parallel and distributed systems, vol. 24, no. 12.

J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weath-erspoon, W. Weimer, C. Wells, and B. Zhao. (November 2000) Oceanstore, An architecture for global-scale persistent storage, In Proceedings of ACM ASPLOS ’00. ACM.

A. A. Muthitacharoen, R. Morris, T. M. Gil, and B.Chen (2002), Ivy: A read/write peer-to-peer file system. In Proceedings of 5th Symposium on Operating Systems Design and Implementation.

J. Li, M. Krohn, D. Mazi`eres, and D. Shasha (2004), Secure untrusted data repository (SUNDR), In Proceedings of the Symposium on Operating Systems Design and Implementation.

P.Maniatis, M. Roussopoulos, T. Giuli, D. Rosenthal, M. Baker, and Y.Muliadi (2005), The LOCKSS peer-to-peer digital preservation system, ACM Transactions on Computing Systems, 23(1):2–50.

A. Y. Yumerefendi and J. Chase (2007), Strong accountability for network storage, In Proc. of FAST,.

U. Maheshwari, R. Vingralek, and W. Shapiro (2000), How to build a trusted database system on untrusted storage, In Proc. of OSDI.

P. Golle, S. Jarecki, and I. Mironov (2002), Cryptographic primitives enforcing communication and storage complexity, In Financial Cryptography, pages 120–135

L. Fortnow, J. Rompel, and M. Sipser (1988), On the Power of Multi-Prover Interactive Protocols, J. Theoretical Computer Science, vol. 134, pp. 156-161.

H. Shacham and B. Waters (2008), Compact Proofs of Retrievability, Proc. 14th Int’l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT ’08), pp. 90-107.

A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone (October 1996). Handbook of Applied Cryptography, CRC Press.

E. Goh, H. Shacham, N. Modadugu, and D. Boneh, SiRiUS (February 2003): Securing Remote Untrusted Storage, In Proceedings of the Tenth Network and Distributed Systems Security (NDSS) Symposium, pages 131–145,.

M. Dworkin (2001), Recommendation for block cipher modes of operation, In Special Publication 800-38A, NIST.

R. Merkle (1987), A digital signature based on a conventional encryption function, In CRYPTO.