Cooperative provable data retention for integrity authentication in multi-cloud Storage

Krishna Kumar Singh, Rajkumar Gaura, Sudhir Kumar Singh

Abstract: Provable data retention (PDR) is a technique which certain the integrity of data in storage outsourcing. In this paper we propose an efficient PDR protocol that prevent attacker in gaining information from multiple cloud storage node. Our technique is for distributed cloud storage and support the scalability of services and data migration. This technique Cooperative store and maintain the client's data on multi cloud storage. To insure the security of our technique we use zero-knowledge proof system, which satisfies zero-knowledge properties, knowledge soundness and completeness. We present a Cooperative PDR (CPDR) protocol based on hash index hierarchy and homomorphic authentication response. In order to optimize the performance of our technique we use a novel technique for selecting optimal parameter values to reduce the storage overhead and computation costs of client for service providers. Our experiment shows that our solution reflects less communication and computation costs in comparison to non-cooperative approaches.

Keyword: provable, Data Retention, integrity, scalability, homomorphic, zero knowledge, storage outsourcing, multiple cloud, Cooperative, data Retention.

1 INTRODUCTION

IN past few years, a cloud storage service has become a faster profitable growth point by providing their clients a reasonably scalable, low-cost, position-independent platform for client's data. As cloud computing environment is made based on open architectures and interfaces, it has the capability to incorporate multiple internal or/and external cloud services together to provide high interoperability. We say such a distributed cloud environment as a hybrid cloud (or multi-Cloud). Very often, we use virtual infrastructure management (VIM) [2], a multi-cloud allows clients to easily access his or her resources remotely through interfaces such as Web services provided by Amazon EC2. There exist various tools and technologies for multicloud, such as Vmware vSphere, Platform VM Orchestrator and Ovirt. These tools help cloud providers to construct a distributed cloud storage platform (DCSP) for managing client's data. However, such an important platform is vulnerable to be compromised, especially in a hostile environment and it would bring irretrievable losses to the clients. For example, the confidential data in an enterprise may be illegally accessed through a remote interface provided by a multi-cloud, or confidential data and archives may be lost or altered with when they are stored into a hostile storage pool outside the enterprise. Therefore, it is important and necessary for cloud service providers (CSPs) to provide security techniques for managing their storage services. Provable data retention (PDR) [1] (or proofs of retrievability (POR) [2]) is such a probabilistic proof technique for a storage provider to prove the integrity and ownership of clients' data without downloading data. The authentication without downloading makes it especially important for large-size files and folders (typically including many clients' files) to check whether these data have been altered with or deleted without downloading the latest version of data. Thus, it is able to replace traditional hash and signature functions in storage outsourcing. Various PDR techniques have been recently proposed, such as Scalable PDR [4] and Dynamic PDR [5]. However, these techniques mainly focus on PDR issues at untrusted servers in a single cloud storage provider and are not suitable for a multi-cloud environment (see the comparison of POR/PDR techniques in Table 1).

Motivation: In order to provide a low-cost, scalable, location-independent platform for managing clients' data, current cloud storage

TABLE 1: Comparison of POR/PDR schemes for a file consisting of n blocks

Scheme	Type	CSP Comp.	Client Comp.	Comm.	Flag.	Privacy	Multiple Clouds	Prob. Of Detection
PDR[2]	$HomT$	O(t)	O(t)	O(1)		▯	#	$1-(1-\rho)t$
SPDR[4]	MHT	O(t)	O(t)	O(t)	▯	▯		$1-(1-\rho)t\cdot s$
DPDR-[5]	MHT	O(t.log n)	O(t.log n)	O(t log n)		▯		$1-(1-\rho)t$
DPDR-II[5]	MHT	O(t log n)	O(t log n)	O(t log n)				$1-(1-\rho)\Omega(n)$
CPOR-[6]	$HomT$	O(t)	O(t)	O(1)			#	$1-(1-\rho)t$
CPOR-II{6}	$HomT$	O(t+s)	O(t+s)	O(s)	▯		#	$1-(1-\rho)$t-s
OurScheme	HomR	O(t+c.s)	O(t+s)	O(s)	▯	▯	▯	$1 - \Pi Pk \in \mathcal{P}$ $(1-\rho k) rk\cdot t\cdot s$

s is the number of sectors in each block, c is the number of CSPs in a multi-cloud, t is the number of sampling blocks,ρ and ρk are the probability of block corruption in a cloud server and k-th cloud server in a multi-cloud $\mathcal{P}$ = {Pk}, respective, ♯ denotes the verification process in a trivial approach, and MHT,$HomT$,$HomR$ denotes Merkle Hash tree, homomorphic tags, and homomorphic response respectively.

systems adopt several new distributed file systems, for example, Google File System (GFS), Apache Hadoop Distribution File System (HDFS), Amazon S3 File System, CloudStore etc. These file systems share some similar features: a single metadata server provides centralized management by a global namespace; files are split into blocks or chunks and stored on block servers; and the systems are comprised of interconnected clusters of block servers. Those features enable cloud service providers to store and process large amounts of data. However, it is crucial to offer an efficient authentication on the integrity and availability of stored data for detecting faults and automatic recovery. Moreover, this authentication is necessary to provide reliability by automatically maintaining multiple copies of data and automatically redeploying processing logic in the event of failures. Although existing techniques can make a false or true decision for data retention without downloading data at untrusted stores, they are not suitable for a distributed cloud storage environment since they were not originally constructed on interactive proof system. For example, the techniques based on Merkle Hash tree (MHT), such as Dynamic PDR-I, Dynamic PDR-II [1] and scalable PDR [4] in Table-1. Use an authenticated skip list to check the integrity of file blocks adjacently in space Unfortunately, they did not provide any algorithms for constructing distributed Merkle trees that are necessary for efficient authentication in a multi-cloud environment. In addition, when a client asks for a file block, the server needs to send the file block along with a proof for the correctness of the block. However, this process incurs significant communication overhead in a multi-cloud environment, since the server in one cloud typically needs to generate such a proof with the help of other cloud storage services, where the adjacent blocks are stored. The other techniques, such as PDR [1], CPOR-I, and CPOR-II [6] in Table 1, are constructed on homomorphic authentication tags, by which the server can generate tags for multiple file blocks in terms of a single response value. However, that doesn't mean the responses from multiple clouds can be also combined into a single value on the client side. In case of lack of homomorphic responses, clients must invoke the PDR protocol repeatedly to check the integrity of file blocks stored in multiple cloud servers. Also, clients need to know the exact position of each file block in a multi-cloud environment. In addition, the authentication process in such a case will lead to high communication overheads and computation costs at client sides as well. Therefore, it is of utmost necessary to design a Cooperative PDR model to reduce the storage and network overheads and enhance the transparency of authentication activities in cluster-based cloud storage systems. Moreover, such a Cooperative PDR technique should provide features for timely detecting abnormality and renewing multiple copies of data. Even though existing PDR techniques have addressed various security properties, such as public verifiability [1], dynamics [5], scalability [4], and privacy preservation [7], we still need a careful consideration of some potential attacks, including two major categories: Data Leakage Attack by which an adversary can easily obtain the stored data through authentication process after running or wire-tapping sufficient authentication communications and Tag Forgery Attack by which a dishonest CSP can deceive the clients. These two 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. Although various security models have been proposed for existing PDR techniques [1], [7], [6], these models still cannot cover all security requirements, especially for provable secure privacy preservation and ownership authentication. To establish a highly effective security model, it is necessary to analyze the PDR technique within the framework of zero-knowledge proof system (ZKPS) due to the reason that PDR system is essentially an interactive proof system (IPS), which has been well studied in the cryptography community. In summary, an authentication technique for data integrity in distributed storage environments should have the following features: Usability aspect: A client should utilize the integrity check in the way of collaboration services. The technique should conceal the details of the storage to reduce the burden on clients; Security aspect: The technique should provide adequate security features to resist some existing attacks, such as data leakage attack and tag forgery attack; Performance aspect: The technique should have the lower communication and computation overheads than non-Cooperative solution.

Related Works: To ensure the integrity and availability of outsourced data in cloud storages, researchers have proposed two basic approaches called Provable data retention (PDR) [1] and Proofs of Retrievability (POR) [1]. Ateniese et al. [1] first proposed the PDR model for ensuring retention of files on untrusted storages and provided an RSA-based technique for a static case that achieves the (1) communication cost. They also proposed a publicly verifiable version, which allows anyone, not just the owner, to challenge the server for data retention. This property greatly extended application areas of PDR protocol due to the separation of data owners and the users. However, these techniques are insecure against replay attacks in dynamic scenarios because of the dependencies on the index of blocks. Moreover, they do not fit for multi-cloud storage due to the loss of homomorphism property in the authentication process. In order to support dynamic data operations, Ateniese et al. developed a dynamic PDR solution called Scalable PDR [4]. They proposed a lightweight PDR technique based on cryptographic hash function and symmetric key encryption, but the servers can deceive the owners by using previous metadata or responses due to the lack of randomness in the challenges. The numbers of updates and challenges are limited and fixed in advance and users cannot perform block insertions anywhere. Based on this work, Erway etal. [5] Introduced two Dynamic PDR techniques with a hash function tree to realize (log n) communication and computational costs for a n-block file. The basic technique, called DPDR-I, retains the drawback of Scalable PDR, and in the 'blockless' technique, called DPDRII, the data blocks $\{mij\ \}j\in[1,t]$ can be leaked by the response of a challenge, $M\ =\sum_{j=1}^{t} aj\ \ mij$, where aj is a random challenge value. Furthermore, these techniques are also not effective for a multi-cloud environment because the authentication path of the challenge block cannot be stored completely in a cloud [8]. Juels and Kaliski [3] presented a POR technique, which relies largely on preprocessing steps that the client conducts before sending a file to a CSP. Unfortunately, these operations prevent any efficient extension for updating data. Shacham and Waters [6] proposed an improved version of this protocol called Compact POR, which uses homomorphic property to aggregate a proof into (1) authenticator value and $O(t)$ computation cost for t challenge blocks, but their solution is also static and could not prevent the leakage of data blocks in the authentication process. Wang et al. [7] presented a dynamic technique with (log n) cost by integrating the Compact POR technique and Merkle Hash Tree (MHT) into the DPDR. Furthermore, several POR techniques and models have been recently proposed including [9], [10]. In [9] Bowers et al. introduced a distributed cryptographic system that allows a set of servers to solve the PDR problem. This system is based on an integrity-protected error Correcting code (IP-ECC), which improves the security and efficiency of existing tools, like POR. However, a file must be transformed into l distinct segments with the same length, which are distributed across l servers. Hence, this system is more suitable for RAID rather than cloud storage.

Our Contributions, in this paper, we address the problem of provable data retention in distributed cloud environments from the following aspects: high performance, transparent authentication, and high security. To achieve these goals, we first propose a authentication framework for multi-cloud storage along with two fundamental techniques: homomorphic verifiable response (HVR) and hash index hierarchy (HIH). We then demonstrate that the possibility of constructing a Cooperative PDR (CPDR) technique without compromising data privacy based on modern cryptographic techniques, such as interactive proof system (IPS). We further introduce an effective construction of CPDR technique using above-mentioned structure. Moreover, we give a security analysis of our CPDR technique from the IPS model. We prove that this construction is a multi-prover zero-knowledge proof system (MP-ZKPS) [11], which has zero-knowledge properties, completeness and knowledge soundness. These properties ensure that CPDR technique can implement the security against data leakage attack and tag forgery attack. To improve the system performance with respect to our technique, we analyze the performance of probabilistic queries for detecting abnormal situations. This probabilistic method also has an inherent benefit in reducing computation and communication overheads. Then, we present an efficient method for the selection of optimal parameter values to minimize the computation overheads of CSPs and the clients' operations. In addition, we analyze that our technique is suitable for existing distributed cloud storage systems. Finally, our experiments show that our solution introduces very limited computation and communication overheads.

Organization: The rest of this paper is organized as follows. In Section 2, we describe a formal definition of CPDR and the underlying techniques, which are utilized in the construction of our technique. We introduce the details of Cooperative PDR technique for multicloud storage in Section 3. We describe the security and performance evaluation of our technique in Section 4 and 5, respectively. We discuss the related work in Section and Section 6 concludes this paper.

2 STRUCTURE AND TECHNIQUES

In this section, we present our authentication framework for multi-cloud storage and a formal definition of CPDR. We introduce two fundamental techniques for constructing our CPDR technique: hash index hierarchy (HIH) on which the responses of the clients' challenges computed from multiple CSPs can be combined into a single response as the final result; and homomorphic verifiable response (HVR) which supports distributed cloud storage in a multi-cloud storage and implements an efficient construction of collision resistant hash function, which can be viewed as a random oracle model in the authentication protocol.

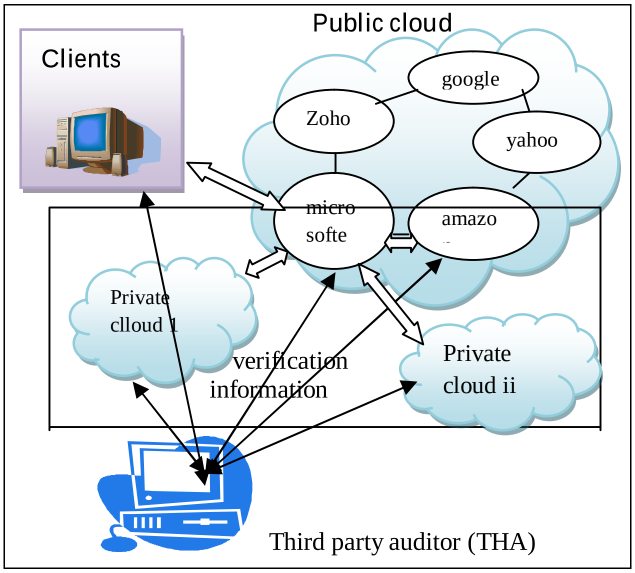


Fig 1: Verification architecture for data integrity.

2.1 Authentication Framework for Multi-Cloud: Although existing PDR techniques offer a publicly accessible remote interface for checking and managing the tremendous amount of data, the majority of existing PDR techniques is incapable to satisfy the inherent requirements from multiple clouds in terms of communication and computation costs. To address this problem, we consider a multi-cloud storage service as illustrated in Figure 1. In this architecture, a data storage service involves three different entities: Clients who have a large amount of data to be stored in multiple clouds and have the permissions to access and manipulate stored data; Cloud Service Providers (CSPs) who work together to provide data storage services and have enough storages and computation resources; and Trusted Third Party (TTP) who is trusted to store authentication parameters and offer public query services for these parameters. In this architecture, we consider the existence of multiple CSPs to Cooperative store and maintain the clients' data. Moreover, a Cooperative PDR is used to verify the integrity and availability of their stored data in all CSPs. The authentication procedure is described as follows: Firstly, a client (data owner) uses the secret key to pre-process a file which consists of a collection of n blocks, generates a set of public authentication information that is stored in TTP, transmits the file and some authentication tags to CSPs, and may delete its local copy; Then, by using a authentication protocol, the clients can issue a challenge for one CSP to check the integrity and availability of outsourced data with respect to public information stored in TTP. We neither assume that CSP is trust to guarantee the security of the stored data, nor assume that data owner has the ability to collect the evidence of the CSP's fault after errors have been found. To achieve this goal, a TTP server is constructed as a core trust base on the cloud for the sake of security We assume the TTP is reliable and independent through the following functions [12]: to setup and maintain the CPDR cryptosystem; to generate and store data owner's public key; and to store the public parameters used to execute the authentication protocol in the CPDR technique. Note that the TTP is not directly involved in the CPDR technique in order to reduce the complexity of cryptosystem.

2.2 Definition of Cooperative PDR: In order to prove the integrity of data stored in a multi-cloud environment, we define a framework for CPDR based on interactive proof system (IPS) and multi-prover zero-knowledge proof system (MPZKPS), as follows: Definition 1 (Cooperative-PDR): A Cooperative provable data retention $\mathcal{S}$ = ($KeyGen$, $T\ agGen$, $Proof$) is a collection of two algorithms ($KeyGen$, $TagGen$) and an interactive proof system $Proof$, as follows: (1^k): takes a security parameter k as input, and returns a secret key sk or a public-secret key-pair (pk, sk); $TagGen$(sk, F,$\mathcal{P}$): takes as inputs a secret key sk, a file F, and a set of cloud storage providers $\mathcal{P}$ = {Pk}, and returns the triples (ζ,ψ, σ), where ζ is the secret in tags, ψ = (u,$\mathcal{H}$) is a set of authentication parameters u and an index hierarchy $\mathcal{H}$ for F, σ = {$\sigma^{(k)}$}p_k∈ $\mathcal{P}$ denotes a set of all tags, $\sigma^{(k)}$ is the tag of the fraction $F^{(k)}$ of F in P_k; ($\boldsymbol{\mathcal{P}}$, V): is a protocol of proof of data retention between CSPs ($\mathcal{P}$ = {P_k}) and a verifier (V), that is,
$\langle\sum_{P_k \in p} P_k(F^{(k)}, \sigma^{(k)}) \leftrightarrow V\rangle(pk,\psi)$=
$\begin{cases} 1, & F = \{F^{(k)}\} is\ intact \\ 0, & F = \{F^{(k)}\} is\ changed \end{cases}$

Where each P_k takes as input a file $F^{(k)}$ and a set of tags $\sigma^{(k)}$, and a public key pk and a set of public parameters ψ are the common input between P and V. At the end of the protocol run, V returns a bit {1|0} denoting true and false. Where, $\sum P_k \in p$ denotes Cooperative computing in $P_k \in \mathcal{P}$. A trivial way to realize the CPDR is to check the data stored in each cloud one by one, i.e. $\wedge_{P_k \in p}\langle P_k(F^{(k)}, \sigma^{(k)} \leftrightarrow$ V⟩(pk, ψ) Where ∧ denotes the logical AND operations among the Boolean outputs of all protocols $\langle P_k, V\ \rangle$ for all $P_k \in \mathcal{P}$. However, it would cause significant communication and computation overheads for the verifier, as well as a loss of location-transparent. Such a primitive approach obviously diminishes the advantages of cloud storage: scaling arbitrarily up and down on demand [13]. To solve this problem, we extend above definition by adding an organizer (O), which is one of CSPs that directly contacts with the verifier, as follows: $\langle\sum_{P_k \in p} P_k(F^{(k)}, \sigma^{(k)})$ ↔O↔$V\rangle$ (pk, ψ), Where the action of organizer is to initiate and organize the authentication process. This definition is consistent with aforementioned architecture, e.g., a client (or an authorized application) is considered as, the CSPs are as $\mathcal{P}$ = {P_i}$i \in [1, c]$, and the Zoho cloud is as the organizer in Figure 1. Often, the organizer is an independent server or a certain CSP in $\mathcal{P}$. The advantage of this new multi-prover proof system is that it does not make any difference for the clients between multi-prover authentication process and single-prover authentication process in the way of collaboration. Also, this kind of transparent authentication is able to conceal the details of data storage to reduce the burden on clients. For the sake of clarity, we list some used signals in Table 2.

TABLE 2: The signal and its explanation.

Sig.	Repression
n	the number of blocks in a file;
s	the number of sectors in each block;
t	the number of index coefficient pairs in a query;
c	the number of clouds to store a file;
F	the file with $n \times s$ sectors, i.e., $F = \{mi,j\}i \in [1,n]$,$j \in [1,s]$;
σ	the set of tags, i.e., $\sigma = \{\sigma i\}i \in [1,n]$;
Q	the set of index-coefficient pairs, i.e., $Q = \{(i, vi)\}$;
θ	the response for the challenge Q.

2.3 Hash Index Hierarchy for CPDR: To support distributed cloud storage, we illustrate a representative architecture used in our Cooperative PDR technique as shown in Figure 2. Our architecture has a hierarchy structure which resembles a natural representation of file storage. This hierarchical structure $\mathcal{H}$ consists of three layers to represent relationships among all blocks for stored resources. They are described as follows: 1) Express Layer: offers an abstract representation of the stored resources; 2) Service Layer: offers and manages cloud storage services; and 3) Storage Layer: realizes data storage on many physical devices. We make 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. For example, in Figure 2 the resources in Express Layer are split and stored into three CSPs, which are indicated by different colors, in Service Layer. In turn, each CSP fragments and stores the assigned data into the storage servers in Storage Layer. We also make use of colors to distinguish different CSPs. Moreover, we follow the logical order of the data blocks to organize the Storage Layer. This architecture also provides special functions for data storage and management, e.g., there may exist overlaps among data blocks (as shown in dashed boxes) and discontinuous blocks but these functions may increase the complexity of storage management. In storage layer, we define a common fragment structure that provides probabilistic authentication of data integrity for outsourced storage. The fragment structure is a data structure that maintains a set of block-tag pairs, allowing searches, checks and updates in (1) time. An instance of this structure is shown in storage layer of Figure 2: an outsourced file F is split into n blocks $\{m1$, m2, $\cdot \cdot \cdot$,$\}$, and each block mi is split into s sectors $\{mi,1,mi,2, \cdot \cdot \cdot ,mi,s\}$. The fragment structure consists of n block-tag pair (m_i, σ_i), where σ_i is a signature tag of block m_i generated by a set of secrets $\tau = (\tau_1, \tau_2, \cdot \cdot \cdot , \tau_s)$. In order to check the data integrity, the fragment structure implements probabilistic authentication as follows: given a random chosen challenge (or query) $Q = \{(i,v_i)\}$ $i \in RI$, where I is a subset of the block indices and v_i is a random coefficient. There exists an efficient algorithm to produce a constant-size response $(\mu_1, \mu_2, \cdot \cdot \cdot , \mu_s, \sigma')$, where μ_i comes from all $\{m_k,i, v_k\}$ k∈I and σ' is from all $\{\sigma_k, v_k\}$ k∈ I. Given a collision-resistant hash function H_k $(\cdot)$, we make use of this architecture to construct a Hash Index Hierarchy $\mathcal{H}$ (viewed as a random oracle), which is used to replace the common hash function in prior PDR techniques, as follows: 1) Express layer: given s random $\{\tau_i\}_{i=1}^{s}$ and the file nameF_n, sets $\xi^{(1)} = H_{\sum_i^s \tau_i} F_n$=1 and makes it public for authentication but makes $\{\tau_i\}_{i=1}^{s}$ secret; 2) Service layer: given the $\xi^{(1)}$ and the cloud nameC_k, sets $\xi^{(2)} = H_{\xi^{(1)}}(C_k)$; 3) Storage layer: given the$\xi^{(2)}$, a block number i, and its index record X_i = "$B_i || V_i || R_i$", sets $\xi_{i,k}^{(3)} = H_{\xi_{i,k}^{(2)}}(X_i)$, where B_i is the sequence number of a block, V_i is the updated version number, and R_i is a random integer to avoid collision. As a virtualization approach, we introduce a simple index-hash table X = $\{X_i\}$ to record the changes of file blocks as well as to generate the hash value of each block in the authentication process. The structure of X is similar to the structure of file block allocation table in file systems. The index-hash table consists of serial number, block number, version number, random integer, and so on. Different from the common index table, we assure that all records in our index table to differ from one another prevent forgery of data blocks and tags. By using this structure, especially the index records $\{X_i\}$, our CPDR technique can also support dynamic data operations [8].The proposed structure can be readily incorporated into MAC-based, ECC or RSA techniques [1], [6]. These techniques, built from collision-resistance signatures (see Section3.1) and the random oracle model, have the shortest query and response with

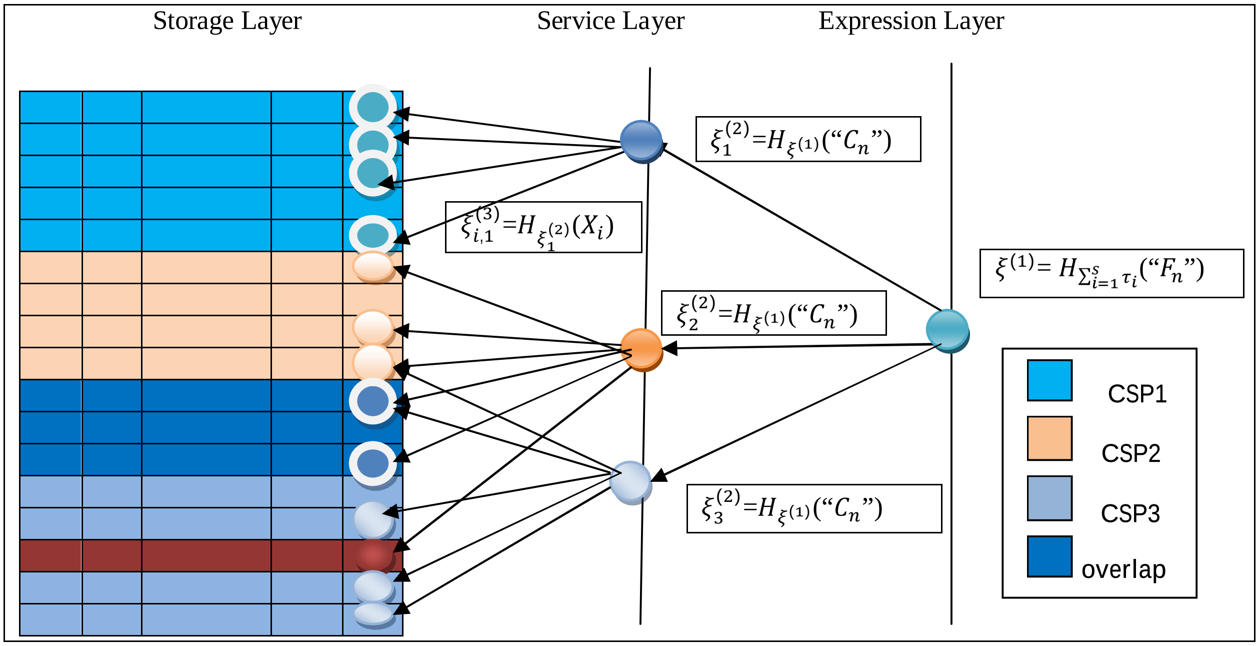

Storage Layer
Service Layer
Expression Layer
$\xi_1^{(2)}=H_{\xi^{(1)}}(\text{“}C_n\text{”})$
$\xi_{i,1}^{(3)}=H_{\xi_1^{(2)}}(X_i)$
$\xi^{(1)}= H_{\sum_{i=1}^{s}\tau_i}(\text{“}F_n\text{”})$
$\xi_2^{(2)}=H_{\xi^{(1)}}(\text{“}C_n\text{”})$
$\xi_3^{(2)}=H_{\xi^{(1)}}(\text{“}C_n\text{”})$
CSP1
CSP2
CSP3
overlap

Fig 2: Index-hash hierarchy of CPDR model.

public verifiability. They share several common characters for the implementation of the CPDR framework in the multiple clouds: 1) a file is split into $n \times s$ 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; 2) a verifier can verify the integrity of file in random sampling approach, which is of utmost importance for large files; 3) these techniques rely on homomorphic properties to aggregate data and tags into a constant size response, which minimizes the overhead of network communication; and 4) the hierarchy structure provides a virtualization approach to conceal the storage details of multiple CSPs.

2.4 Homomorphic Verifiable Response for CPDR: A homomorphism is a map f:$\mathbb{P}$→$\mathbb{Q}$ between two groups such that $f(g_{1\oplus}g_2)$ = $f(g_1)$ $\otimes f(g_2)$ for all g_1, g_2 □ $\mathbb{P}$, where $\oplus$ denotes the operation in $\mathbb{P}$ and $\otimes$ denotes the operation in $\mathbb{Q}$. This notation has been used to define Homomorphic Verifiable Tags (HVTs) in [1]: Given two values σ_i and σ_j for two messages m_i and m_j , anyone can combine them into a value $\sigma_i{}'$ corresponding to the sum of the messages $m_i + m_j$. When provable data retention is considered as a challenge-response protocol, we extend this notation to the concept of Homomorphic Verifiable Responses (HVR), which is used to integrate multiple responses from the different CSPs in CPDR technique as follows: *Definition* 2 (Homomorphic Verifiable Response): A response is called homomorphic verifiable response in a PDR protocol, if given two responses Θ_i and Θ_j for two challenges Q_i and Q_j from two CSPs, there exists an efficient algorithm to combine them into a response θ corresponding to the sum of the challenges $Q_i \cup Q_j$. Homomorphic verifiable response is the key technique of CPDR because it not only reduces the communication bandwidth, but also conceals the location of outsourced data in the distributed cloud storage environment.

3 COOPERATIVE PDR TECHNIQUES:

In this section, we propose a CPDR technique for multi-cloud system based on the above-mentioned structure and techniques. This technique is constructed on collision-resistant hash, bilinear map group, aggregation algorithm, and homomorphic responses.

3.1 Notations and Preliminaries: Let $\mathbb{H}$ = $\{H_k\}$ be a family of hash functions H_k : $\{0,1\}^n \rightarrow \{0,1\}^*$ index by $k \in \mathcal{K}$. We say that algorithm $\mathcal{A}$ has advantage ϵ in breaking collision resistance of $\mathbb{H}$ if Pr[$\mathcal{A}(k)$ = (m_0, m_1) : $m_0 \not= m_1, H_k$ (m_0) = H_k (m_0)] ≥ ϵ, where the probability is over the random choices of $k \in \mathcal{K}$ and the random bits of $\mathcal{A}$. So that, we have the following definition: Definition 3 (Collision-Resistant Hash): A hash family $\mathbb{H}$ is (t, ϵ)-collision-resistant if no t-time adversary has advantage at least ϵ in breaking collision resistance of $\mathbb{H}$. We set up our system using bilinear pairings proposed by Boneh and Franklin [14]. Let $\mathbb{G}$ and $\mathbb{G}T$ be two multiplicative groups using elliptic curve conventions with a large prime order p. The function e is a computable bilinear map e : $\mathbb{G}\times\mathbb{G} \rightarrow \mathbb{G}T$ with the following properties: for any G,$H \in \mathbb{G}$ and all a, $b \in \mathbb{Z}p$, we have 1) Bilinearity: $e([a]G$, $[b]H)$ = $e(G,H)ab$; 2) Non-degeneracy: $e(G,H) \neq 1$ unless G or H = 1; and 3) Computability: $e(G,H)$ is efficiently computable. Definition 4 (Bilinear Map Group System): A bilinear map group system is a tuple $\mathbb{S}$ = $\langle p,,, e\rangle$ composed of the objects as described above.

3.2 Our CPDR Technique: In our technique (see Fig 3), 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 5-move interactive Proof protocol between a verifier and more than one CSP, in which CSPs need not to interact with each other during the authentication process, but an organizer, is used to organize and manage all CSPs. This protocol can be described as follows: 1) The organizer initiates the protocol and sends a commitment to the verifier; 2) The verifier returns a challenge set of random index-coefficient pair's Q to the organizer; 3) The organizer relays them into each lock; 4) Each Pi returns its response of challenge to the organizer; and 5) The organizer synthesizes a Pi in $\mathcal{P}$ according to the exact position of each data 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. In contrast to a single CSP environment, our technique differs from the common PDR technique in two aspects: 1) Tag aggregation algorithm: In stage of commitment, the organizer generates a random $\gamma \in R$ $\mathbb{Z}p$ and returns its commitment H'_1 to the verifier. This assures that the verifier and CSPs do not obtain the value of γ. Therefore, our approach guarantees only the organizer can compute the final σ' by using γ and σ' k received from CSPs. After σ' is computed, we need to transfer it to the organizer in stage of "Response1". In order to ensure the security of transmission of data tags, our technique employs a new method, similar to the ElGamal encryption, to encrypt the combination of tags $\prod_{(i,v_i)\in Q_k} \sigma_i^{v_i}$, that is, for $sk = s \in \mathbb{Z}p$ and $P_k = (g, S = g^s) \in\mathbb{G}^2$, the cipher of message m is $\mathcal{C} = (\mathcal{C}_1 = gr, \mathcal{C}_2 = m \cdot s^r)$ and its decryption is performed by $m = C_2.C_1^{-s}$.

2) Homomorphic responses: Because of the homomorphic property, the responses computed from CSPs in a multi-cloud can be combined into a single final response. It is obvious that the final response θ received by the verifiers from multiple CSPs is same as that in one simple CSP. This means that our CPDR technique is able to provide a transparent authentication for the verifiers. Two response algorithms, Response1 and Response2, comprise an HVR: Given two responses θi and θj for two challenges Qi and Qj from two CSPs, i.e., θi = $Response1$ (Qi, $\{mk\}$ $k\in Ii$, $\{\sigma k\}$ $k\in Ii$), there exists an efficient algorithm to combine them into a final response θ corresponding to the sum of the challenges Qi ∪, that is, = $Response1$ (Qi ∪, $\{mk\}$ $k\in Ii$ $\cup Ij$, $\{\sigma k\}$ $k\in Ii$ $\cup Ij$) = $Response2$ (θ_i, θ_j). For multiple CSPs, the above equation can be extended to θ = $Response2$ ($\{\theta k\} \in\mathcal{P}$). More importantly, the HVR is a pair of values $\theta = (\pi, \sigma, \mu)$, which has a constant-size even for different challenges.

4 SECURITY ANALYSES: We give a brief security analysis of our CPDR construction. This construction is directly derived from multi-prover zero-knowledge proof system (MPZKPS), which satisfies following properties for a given assertion, L: 1) Completeness: whenever $x \in L$, there existsa strategy for the provers that convinces the verifier that this is the case; 2) Soundness: whenever $x \notin L$, whatever strategy the provers employ, they will not convince the verifier that $x \in L$; 3) Zero-knowledge: no cheating verifier can learn anything other than the veracity of the statement. According to existing IPS research [15], these properties can protect our construction from various attacks, such as data leakage attack (privacy leakage), tag forgery attack (ownership cheating), etc. In details, the security of our technique can be analyzed as follows:

Fig 3: Cooperative provable data retention for integrity authentication in multi-cloud Storage

KeyGen (1^k): Let $\mathbb{S}$ = (p, $\mathbb{G}$, , e) be a bilinear map group system with randomly selected generators $g, h \in \mathbb{G}$, where $\mathbb{G}$,$\mathbb{G}_T$ are two bilinear groups of a large prime order p, $|p|$ = $O(\kappa)$. Makes a hash function (·) public. For a CSP, chooses a random number $s \in R\ \mathbb{Z}p$ and computes $S = gs \in \mathbb{G}$. Thus, sk_p = s and pk_p = (g, S). For a user, chooses two random numbers α, $\beta \in R\ \mathbb{Z}p$ and sets sk_u = (α, β) and pk_u = (g, h, $H_1 = h^{\alpha}$, $H_2 = h^{\beta}$).

TagGen (sk,, $\mathcal{P}$): Splits F into $n \times s$ sectors { $m_{i,j}$}i∈[1,n],j∈[1,s] ∈ $\mathbb{Z}_p^{n\times s}$. Chooses s random$\tau_1, \cdots, \tau_s \in \mathbb{Z}p$ as the secret of this file and computes $u_i = g^{\tau_i} \in \mathbb{G}$ for $i \in [1, s]$. Constructs the index table $\chi = \{\chi i\}_{i=1}^{n}$ and fills out the record χ_i^a in χ for $i \in [1, n]$, then calculates the tag for each block mi as

$$\begin{cases} \xi^{(1)} \leftarrow H_{\sum_{i=1}^{s}\tau_i}(F_n), & \xi_k^{(2)} \leftarrow H_{\xi^{(1)}}(C_k), \\ \xi_k^{(3)} \leftarrow H_{\xi_k^{(2)}}, \quad \sigma_{i,k} & \leftarrow \left(\xi_{i,k}^{(3)}\right)^{\alpha}\cdot\left(\prod_{j=1}^{s} u_j^{m_{i,j}}\right)^{\beta} \end{cases}$$

Where Fn is the file name and Ck is the CSP name of $Pk \in \mathcal{P}$. And then stores ψ = (u, $\xi^{(1)}$, χ) into TTP, and $\sigma_k = \{\sigma_{i,j}\}\ \forall\ j$=$k$ to $P_k \in \mathcal{P}$, where $u = (u_1, \cdots, u_s)$. Finally, the data owner saves the secret $\zeta = (\tau_1, \cdots, \tau_s)$.

Proof($\mathcal{P}$,V): This is a 5-move protocol among the Provers ($\mathcal{P} = \{Pi\}$ i∈[1,c]), an organizer (O), and a Verifier (V) with the common input (pk, ψ), which is stored in TTP, as follows:

1) Commitment($O \rightarrow V$): the organizer chooses a random $\gamma \in R\ \mathbb{Z}p$ and sends H'$_1$ = H_1^{γ} to the verifier;

2) Challenge1($O \leftarrow V$): the verifier chooses a set of challenge index-coefficient pairs $Q = \{(i, v_i)\}\ i \in I$ and sends Q to the organizer, where I is a set of random indexes in [1, n] and vi is a random integer in$\mathbb{Z}_p^*$;

3) Challenge2($\mathcal{P} \leftarrow O$): the organizer forwards $Q_k = \{(i, v_i)\}\ m_i \in p_k \subseteq Q$ to each p_k in $\mathcal{P}$;

4) Response1 ($\mathcal{P}$ → O): p_k chooses a random $r_k \in \mathbb{Z}_p$and s random $\lambda_{j,k} \in \mathbb{Z}_p$for j ∈ [1, s], and calculates a Response $\sigma'_k \leftarrow s^{r_k} \cdot \prod_{(i,v_i)\in Q_k} \sigma_i^{v_i}$, $\mu_{j,k} \leftarrow \lambda_{j,k} + \sum_{(i,v_i)} v_i.m_{i,j}$ $\pi_{j,k} \leftarrow$ e($u_j^{\lambda_{j,k}}$, H_2) , Whereas μ_k = { $\mu_{j,k}$} $_{j\in[1,s]}$ and $\pi_k = \prod_{j=1}^{s} \pi_{j,k}$. Let $\eta_k \leftarrow g^{r_k} \in \mathbb{G}$, each p_k sends $\theta_k = (\pi_k, \sigma'_k, \mu_k, \eta_k)$ to the organizer;

5) Response2 (O → V): After receiving all responses from$\{p_i\}_{i\in[1,e]}$, the organizer aggregates $\{\{\theta_k\}\}_{P_k \in p}$ into a final response θ as:

$$\sigma' \leftarrow \left(\prod_{P_k\in p} \sigma'_k \cdot \eta_k^{-s}\right)^{\gamma}, \mu'_j \leftarrow \sum_{P_k\in p} \gamma . \mu_j, k,\ \pi' \leftarrow \left(\prod_{P_k\in p} \pi_k\right)^{\gamma}. \quad (1)$$

Let $\mu' = \{\mu'_j\}_{j\in[1,s]}$. The organizer sends θ = (π', σ', μ') to the verifier.

Verification: Now the verifier can check whether the response was correctly formed by checking that

$$\pi' . \mathrm{e}(\sigma', h) \overset{?}{=} \mathrm{e}\left(\prod_{(i,v_i\in Q)} H_{\xi_k^{(2)}(X_i)}{}^{v_i}, H'_1\right). \mathrm{e}\left(\prod_{j=1}^{s} u_j{}^{u'_j}, H_2\right). \quad (2)$$

a. For X_i= "B_i, V_i, R_i" in Section 2.3, we can set X_i =(B_i= i, V_i = 1, R_i ∈ R {0, 1}*) at initial stage of CPDR scheme.

4.1 Collision resistant for index-hash hierarchy: In our CPDR technique, the collision resistant of index hash hierarchy is the basis and prerequisite for the security of whole technique, which is described as being secure in the random oracle model. Although the hash function is collision resistant, a successful hash collision can still be used to produce a forged tag when the same hash value is reused multiple times, e.g., a legitimate client modifies the data or repeats to insert and delete data blocks of outsourced data. To avoid the hash collision, the hash value $\xi(3)$ i,k, which is used to generate the tag σi in CPDR technique, is computed from the set of values $\{\tau i\}$, Fn, Ck, $\{\chi i\}$. As long as there exists one bit difference in these data, we can avoid the hash collision. As a consequence, we have the following theorem (see Appendix B): Theorem 1 (Collision Resistant): The index-hash hierarchy in CPDR technique is collision resistant, even if the client generates $\sqrt{2p.\,ln\frac{1}{1-\varepsilon}}$ files with the same file name and cloud name, and the client repeats $\sqrt{2^{L+1}.\,ln\frac{1}{1-\varepsilon}}$ times to modify, insert and delete data blocks, where the collision probability is at least ε, $\tau i \in \mathbb{Z}p$, and $|Ri| = L$ for $Ri \in \chi i$.

4.2 Completeness property of authentication: In our technique, the completeness property implies public verifiability property, which allows anyone, not just the client (data owner), to challenge the cloud server for data integrity and data ownership without the need for any secret information. First, for every available data-tag pair $(F, \sigma) \in (sk, F)$ and a random challenge Q = (i, vi) $i \in I$, the authentication protocol should be completed with success probability according to the Equation (3), that is, Pr $\left[\langle \sum_{P_k \in p} P_k(F^{(k)}, \sigma^{(k)}) \leftrightarrow O \leftrightarrow V\rangle(\mathrm{pk}, \psi) = 1\right] = 1$. In this process, anyone can obtain the owner's public key $pk = (g, h, H_1 = h^{\alpha}, H_2 = h^{\beta})$ and the corresponding file parameter $\psi = (u, \xi^{(1)}, \chi)$ from TTP to execute the authentication protocol, hence this is a public verifiable protocol. Moreover, for different owners, the secrets α and β hidden in their public key pk are also different, determining that a success authentication can only be implemented by the real owner's public key. In addition, the parameter ψ is used to store the file-related information, so an owner can employ a unique public key to deal with a large number of outsourced files.

4.3 Zero-knowledge property of authentication: The CPDR construction is in essence a Multi-Prover Zero-knowledge Proof (MP-ZKP) system [11], which can be considered as an extension of the notion of an interactive proof system (IPS). Roughly speaking, in the scenario of MP-ZKP, a polynomial-time bounded verifier interacts with several provers whose computational powers are unlimited. According to a Simulator model, in which every cheating verifier has a simulator that can produce a transcript that "looks like" an interaction between an honest prover and a cheating verifier, we can prove our CPDR construction has Zero-knowledge property.

Theorem 2 (Zero-Knowledge Property): The authentication protocol $Proof(\mathcal{P}, V)$ in CPDR technique is a computational zero-knowledge system under a simulator model, that is, for every probabilistic polynomial-time interactive machine $V*$, there exists a probabilistic polynomial-time algorithm $S*$ such that the ensembles $View$ (⟨ Σ $Pk \in \mathcal{P}$ $Pk(F(k), \sigma(k)) \leftrightarrow O \leftrightarrow V*\rangle(pk, \psi)$) and $S*(pk, \psi)$ are computationally indistinguishable. Zero-knowledge is a property that achieves the CSPs' robustness against attempts to gain knowledge by interacting with them. For our construction, we make use of the zero-knowledge property to preserve the privacy of data blocks and signature tags. Firstly, randomness is adopted into the CSPs' responses in order to resist the data leakage attacks (see Attacks 1 and 3 in Appendix A). That is, the random integer λj, is introduced into the response μj, i.e., μj, $k = \lambda j$, k + Σ $(i, vi) \in Qk$ $vi \cdot mi$, j. This means that the cheating verifier cannot obtain mi, from μj, because he does not know the random integer λj. At the same time, a random integer γ is also introduced to randomize the authentication tag σ, i.e., $\sigma' \leftarrow (\Pi \, \mathrm{Pk} \in \mathcal{P} \, \sigma' \, \mathrm{k} \cdot \mathrm{R} - \mathrm{s} \, \mathrm{k})^{\gamma}$. Thus, the tag σ cannot reveal to the cheating verifier in terms of randomness.

4.4 Knowledge soundness of authentication: For every data-tag pairs $(F*, \sigma*) \notin (sk, F)$, in order to prove nonexistence of fraudulent $\mathcal{P}*$ and $O*$, we require that the technique satisfies the knowledge soundness property, that is, Pr $\left[\langle \sum_{P_k \in p^*} P_k(F^{(k)*}, \sigma^{(k)*}) \leftrightarrow O^* \leftrightarrow V\rangle(\mathrm{pk}, \psi) = 1\right] \le \epsilon$, where ϵ is a negligible error. We prove that our technique has the knowledge soundness property by using reduction to absurdity 1: we make use of $\mathcal{P}*$ to construct a knowledge extractor $\mathcal{M}$ [7,13], which gets the common input (pk, ψ) and rewindable blackbox accesses to the prover $P*$, and then attempts to break the computational Diffie-Hellman (CDH) problem in $\mathbb{G}$: given $G, G_1 = G^a$, $G_2 = G^b \in R$ $\mathbb{G}$, output $Gab \in \mathbb{G}$. But it is unacceptable because the CDH problem is widely regarded as an unsolved problem in polynomial-time. Thus, the opposite direction of the theorem also follows.

Theorem 3 (Knowledge Soundness Property): Our technique has (t, ϵ') knowledge soundness in random oracle and rewindable knowledge extractor model assuming the (t, ϵ)-computational Diffie-Hellman (CDH) assumption holds in the group $\mathbb{G}$ for $\epsilon' \ge \epsilon$. Essentially, the soundness means that it is infeasible to fool the verifier to accept false statements. Often, the soundness can also be regarded as a stricter notion of unforgeability for file tags to avoid cheating the ownership. This means that the CSPs, even if collusion is attempted, cannot be tampered with the data or forge the data tags if the soundness property holds. Thus, the Theorem 3 denotes that the CPDR technique can resist the tag forgery attacks (see Attacks 2 and 4 in Appendix A) to avoid cheating the CSPs' ownership.

5 PERFORMANCE EVALUATIONS:

In this section, to detect abnormality in a low overhead and timely manner, we analyze and optimize the performance of CPDR technique based on the above technique from two aspects: evaluation of probabilistic queries and optimization of length of blocks. To validate the effects of technique, we introduce a prototype of CPDR-based audit system and present the experimental results.

5.1 Performance Analysis for CPDR Technique: We present the computation cost of our CPDR technique in Table 3. We use $[E]$ to denote the computation cost of an exponent operation in $\mathbb{G}$, namely, gx, where x is a positive integer in $\mathbb{Z}p$ and $g \in \mathbb{G}$ or $\mathbb{G}T$. We neglect the computation cost of algebraic operations and simple modular arithmetic operations because they run fast enough [16]. The most complex operation is the computation of a bilinear map $(\cdot, \cdot)$ between two elliptic points (denoted as $[B]$). Then, we analyze the storage and communication costs of our technique. We define the bilinear pairing takes the form: $(\mathbb{F}pm) \times (\mathbb{F}pkm) \rightarrow \mathbb{F}* \ pkm$ (The definition given here is from [17], [18]), where p is a prime, m is a positive integer, and k is the embedding degree (or security multiplier). In this case, we utilize an asymmetric pairing: $\mathbb{G}1\times\mathbb{G}2\rightarrow \mathbb{G}T$ to replace the operations, where c is the symmetric pairing in the original techniques. In Table 3, it is easy to find that client's computation overheads are entirely irrelevant for the number of CSPs. Further, our technique has better performance compared with non-Cooperative approach due to the total of computation overheads decrease 3(c–1) times bilinear map number of clouds in a multi-cloud. The reason is that, before the responses are sent to the verifier from c clouds, the organizer has aggregate these responses into a response by using aggregation algorithm, so the verifier only need to verify this response once to obtain the final result. Without loss of generality, let the security parameter κ be 80 bits, we need the elliptic curve domain parameters over $\mathbb{F}p$ with $|p|$ = 160 bits and m = 1 in our experiments. This means that the length of integer is $l0 = 2\kappa$ in $\mathbb{Z}p$. Similarly, we have $l1 = 4\kappa$ in $\mathbb{G}1$, $l2 = 24\kappa$ in $\mathbb{G}2$, and $lT = 24\kappa$ in $\mathbb{GT}$ for the embedding degree $k = 6$. The storage and communication cost of our technique is shown in Table 4. The storage overhead of a file with $(f) = 1M$-bytes is $(f) = n \cdot s \cdot l0 + n \cdot l1 = 1.04M$-bytes for $n = 103$ and $s = 50$. The storage overhead of its index table χ is $n \cdot l0 = 20K$-bytes. We define the overhead rate as $\lambda = (f)\ (f) -1 = l1\ s{\cdot}l0$ and it should therefore be kept as low as possible in order to minimize the storage in cloud storage providers. It is obvious that a higher s means much lower storage. Furthermore, in the authentication protocol, the communication overhead of challenge is $2t \cdot l0 = 40\cdot\ t$-Bytes in terms of the number of challenged blocks t, but its response (response1 or response2) has a constant-size communication overhead $s\cdot\ l0+l1+lT \approx 1.3K$-bytes for different file sizes. Also, it implies that client's communication overheads are of a fixed size, which is entirely irrelevant for the number of CSPs.

TABLE 3: Comparison of computation overheads between our CPDR scheme and non-cooperative (trivial) scheme.

	CPDR Scheme	Trivial Scheme
Commitment	$l2$	$cl2$
Challenge1	$2tl0$	$2tl0$
Challenge2	$2tl0/c$	$2tl0$
Response1	$sl0 + 2l1 + lT$	$(sl0 + l1 + lT\ )c$
Response2	$sl0 + l1 + lT$	$(sl0 + l1 + lT\ )c$

TABLE 4: Comparison of communication overheads between our CPDR and non-cooperative scheme.

	CPDR Scheme	Trivial Scheme
KeyGen	$3[E]$	$2[E]$
TagGen	$(2n + s)[E]$	$(2n + s)[E]$
Proof(p)	$c[B] + (t + cs+1)[E]$	$c[B] +(t +cs - c)[E]$
Proof(V)	$3[B] + (t + s)[E]$	$3c[B] + (t + cs)[E]$

5.2 Probabilistic Authentication: We recall the probabilistic authentication of common PDR technique (which only involves one CSP), in which the authentication process achieves the detection of CSP server misbehavior in a random sampling mode in order to reduce the workload on the server. The detection probability of disrupted blocks P is an important parameter to guarantee that these blocks can be detected in time. Assume the CSP modifies e blocks out of the n-block file, that is, the probability of disrupted blocks is $\rho b = e$ n. Let t be the number of queried blocks for a challenge in the authentication protocol. We have detection probability[2] $(\rho b, t) \geq 1 - \left(\frac{n-e}{n}\right)^t = 1-(1-\varrho b)^t$, Where, $(\rho b, t)$ denotes that the probability P is a function over ρb and t. Hence, the number of queried blocks is $t \approx \frac{\log(1-P)}{\log(1-\varrho b)} \approx \frac{P \cdot n}{e}$ for a sufficiently large n and $t \ll n^3$. This means that the number of queried blocks t is directly proportional to the total number of file blocks n for the constant P and e. Therefore, for a uniform random authentication in a PDR technique with fragment structure, given a file with $sz = n \cdot s$ sectors and the probability of sector corruption ρ, the detection probability of authentication protocol has $P \geq 1 - (1-\varrho)^{sz \cdot \omega}$, where ω denotes the sampling probability in the authentication protocol. We can obtain this result as follows: because $\rho b \geq 1 - (1-\varrho)^s$ is the probability of block corruption with s sectors in common PDR technique, the verifier can detect block errors with probability $P \geq 1 - (1 - p_b)^t \geq 1 - ((1-\varrho)^s)^{z \cdot \omega} = 1 - (1-\varrho)^{sz \cdot \omega}$ for a challenge with $t = n \cdot \omega$ index-coefficient pairs. In the same way, given a multi-cloud $\mathcal{P} = \{Pi\} \in [1, c]$, the detection probability of CPDR technique has $(sz, \{\rho k, rk\} \in \mathcal{P}, \omega) \geq 1 - \prod_{Pk \in \mathcal{P}}((1-\varrho k)^s) r_k^{\cdot \omega} = 1 - \prod_{Pk \in \mathcal{P}}(1-\varrho k)^{sz . r_k \cdot \omega}$, where r_k denotes the proportion of data blocks in the k-th CSP, ρk denotes the probability of file corruption 2. Exactly, we have $P = 1-(1-\frac{e}{n}) \cdot (1-\frac{e}{n-1}) \cdots (1-\frac{e}{n-t+1})$.

Since $1-\frac{e}{n} \geq 1-\frac{e}{n-i}$ for $i \in [0, t-1]$, we have $P = 1-\prod_{i=0}^{t-1}(1-\frac{e}{n-i}) \geq 1-\prod_{i=0}^{t-1}(1-\frac{e}{n}) = 1-(1-\frac{e}{n})^t$.

3. In terms of $(1-\frac{e}{n})^t \approx (1-\frac{e.t}{n})$, we have $P \approx 1-(1-\frac{e.t}{n}) = \frac{e.t}{n}$. In the k-th CSP and $rk \cdot \omega$ denotes the possible number of blocks queried by the verifier in the k-th CSP. Furthermore, we observe the ratio of queried blocks in the total file blocks w under different detection probabilities. Based on above analysis, it is easy to find that this ratio holds the equation $w \approx \frac{\log(1-P)}{sz \cdot \Sigma Pk \in \mathcal{P}\ rk \cdot \log(1-\varrho k)}$. When this probability ρk is a constant probability, the verifier can detect sever misbehavior with a certain probability P by asking proof for the number of blocks $t \approx \log(1-P)$ $s.\log(1-\rho)$ for PDR or for $t \approx \frac{\log(1-P)}{s \cdot \Sigma Pk \in \mathcal{P}\ rk \cdot \log(1-\varrho k)}$ CPDR, where $t = n \cdot w = \frac{sz \cdot w}{s}$.

TABLE 5: The influence of s, t under the different corruption probabilities ρ and the different detection probabilities P

$\mathcal{P}$	{0.1,0.2, 0.01}	{0.01,0.02 ,0.001}	{0.001,0.002, 0.0001}	{0.0001,0.0002,0.00001}
r	{0.5,0.3, 0.2}	{0.5,0.3,0. 2}	{0.5,0.3, 0.2}	{0.5,0.3,0.2}
0.8/ 3	4 /7	20/ 23	62/ 71	71/202
0.85 /3	5 /8	21/ 26	65/ 79	79/214
0.9 /3	6/ 10	20 /28	73 /87	87/236
0.95 /3	8/ 11	29/ 31	86/ 100	100/267
0.99 /4	10/ 13	31/ 39	105 /119	119/345
0.999 /5	11/ 16	38 /48	128/ 146	146/433

Note that, the value of t is dependent on the total number of file blocks n [2], because it is increased along with the decrease of ρk and $\log(1-\rho k) < 0$ for the constant number of disrupted blocks e and the larger number n. Another advantage of probabilistic authentication based on random sampling is that it is easy to identify the tampering or forging data blocks or tags. The identification function is obvious: when the authentication fails, we can choose the partial set of challenge indexes as a new challenge set, and continue to execute the authentication protocol. The above search process can be repeatedly executed until the bad block is found. The complexity of such a search process is $(\log n)$.

5.3 Parameter Optimization: In the fragment structure, the number of sectors per block s is an important parameter to affect the performance of storage services and audit services. Hence, we propose an optimization algorithm for the value of s in this section. Our results show that the optimal value can not only minimize the computation and communication overheads, but also reduce the size of extra storage, which is required to store the authentication tags in CSPs. Assume ρ denotes the probability of sector corruption. In the fragment structure, the choosing of s is extremely important for improving the performance of the CPDR technique. Given the detection probability P and the probability of sector corruption ρ for multiple clouds $\mathcal{P}$ = $\{Pk\}$, the optimal value of s can be computed by min$s\in\mathbb{N}$\{ $\frac{\log(1-P)}{\Sigma Pk\in\mathcal{P}\ rk\cdot\log(1-\varrho k)}\cdot\frac{a}{s} + b\cdot s + c$\}, where $a\cdot t + b\cdot s + c$ denotes the computational cost of authentication protocol in PDR technique, $a, b, c \in \mathbb{R}$, and c is a constant. This conclusion can be obtained from following process: Let $sz = n\cdot s = (f)/l_0$. According to above-mentioned results, the sampling probability holds $w \geq \frac{\log(1-P)}{sz\cdot\Sigma Pk\in\mathcal{P}\ rk\cdot\log(1-\varrho k)} = \frac{\log(1-P)}{n\cdot s\cdot\Sigma Pk\in\mathcal{P}\ rk\cdot\log(1-\varrho k)}$.

In order to minimize the computational cost, we have min $s\in\mathbb{N}$ $\{a\cdot t + b\cdot s + c\}$ = min $s\in\mathbb{N}$ $\{a\cdot n\cdot w + b\cdot s + c\}$ ≥ min $s\in\mathbb{N}$ { Σ log(1 − P) $Pk\in\mathcal{P}$ rk · log(1 − ρk) a s + b · s + c } . Where rk denotes the proportion of data blocks in the k-th CSP, ρk denotes the probability of file corruption in the k-th CSP. Since $\frac{a}{s}$ is a monotone decreasing function and $b\cdot s$ is a monotone increasing function for $s > 0$, there exists an optimal value of $s \in \mathbb{N}$ in the above equation. The optimal value of s is unrelated to a certain file from this conclusion if the probability ρ is a constant value. For instance, we assume a multi-cloud storage involves three CSPs $\mathcal{P}$ = $\{P1, P2, P3\}$ and the probability of sector corruption is a constant value $\{\rho1, \rho2, \rho3\}$ = {0.01, 0.02, 0.001}. We set the detection probability P with the range from 0.8 to 1, e.g., P = {0.8, 0.85, 0.9, 0.95, 0.99, and 0.999}. For a file, the proportion of data blocks is 50%, 30%, and 20% in three CSPs, respectively, that is, $r1$ = 0.5, $r2$ = 0.3, and $r3$ = 0.2. In terms of Table 3, the computational cost of CSPs can be simplified to $t + 3s$+9. When s is less than the optimal value, the computational cost decreases evidently with the increase of s, and then it raises when s is more than the optimal value. More accurately, we show the influence of parameters, $sz\cdot w$, s, and t, under different detection probabilities in Table 6. It is easy to see that computational cost rises with the increase of P.

TABLE 6: The influence of parameters under different detection probabilities P ($\mathcal{P}$ = $\{\rho1, \rho2, \rho3\}$ = {0.01, 0.02, 0.001}, $\{r1, r2, r3\}$ = {0.5, 0.3, 0.2})

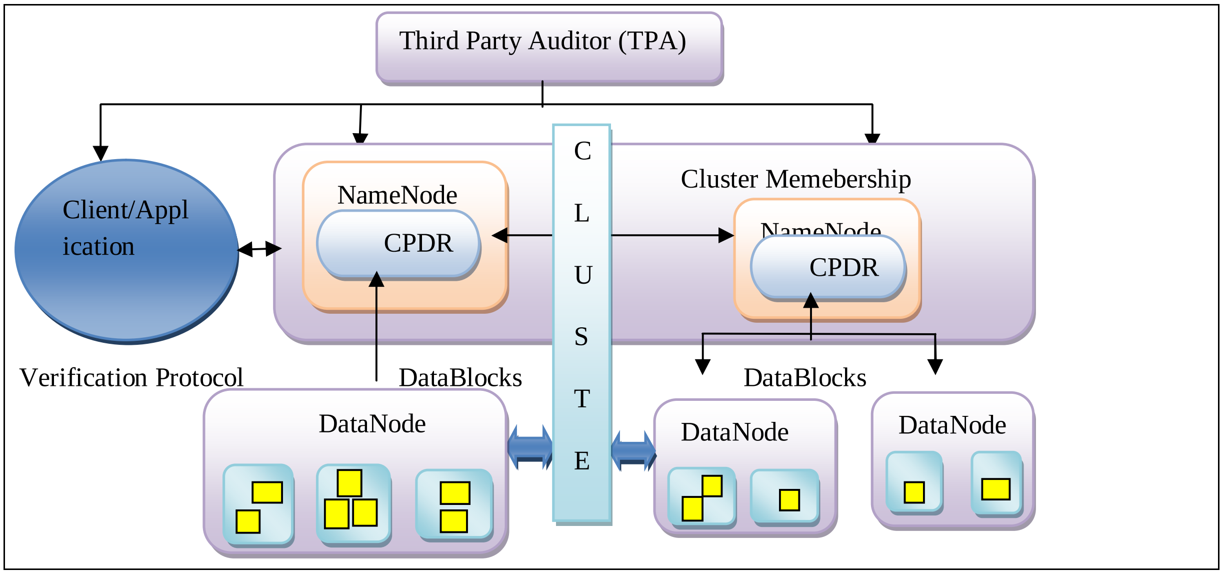
P	0.8	0.85	0.9	0.95	0.99	0.999
$sz\cdot w$	142.60	168.09	204.02	265.43	408.04	612.06
s	7	8	10	11	13	16
t	20	21	20	29	31	38

Moreover, we can make sure the sampling number of challenge with following Conclusion: Given the detection probability P, the probability of sector corruption ρ, and the number of sectors in each block s, the sampling number of authentication protocol are a constant $t = n\cdot w \geq \frac{\log(1-P)}{s\cdot\Sigma Pk\in\mathcal{P}\ rk\cdot\log(1-\varrho k)}$ for different files. Finally, we observe the change of s under different ρ and P. The experimental results are shown in Table 5. It is obvious that the optimal value of s rises with increase of P and with the decrease of ρ. We choose the optimal value of s on the basis of practical settings and system requisition. For NTFS format, we suggest that the value of s is 200 and the size of block is 4KBytes, which is the same as the default size of cluster when the file size is less than 16TB in NTFS. In this case, the value of s ensures that the extra storage doesn't exceed 1% in storage servers.

5.4 CPDR for Integrity Audit Services: Based on our CPDR technique, we introduce audit system architecture for outsourced data in multiple clouds by replacing the TTP with a third party auditor (TPA) in Figure 1. In this architecture, this architecture can be constructed into a visualization infrastructure of cloud-based storage service [1]. In Figure 4, we show an example of applying our CPDR technique in Hadoop distributed file system (HDFS)[4], with a distributed, scalable, and portable file system [19]. HDFS' architecture is composed of NameNode and DataNode, where NameNode maps a file name to a set of indexes of blocks and DataNode indeed stores data blocks. To support our CPDR technique, the index-hash hierarchy and the metadata of NameNode should be integrated together to provide an enquiry service for the hash value $\xi^{(3)}_{i,k}$, or index-hash record χi.

Based on the hash value, the clients can implement the authentication protocol via CPDR services. Hence, it is easy to replace the checksum methods with the CPDR technique for anomaly detection in current HDFS. To validate the effectiveness and efficiency of our proposed approach for audit services, we have implemented a prototype of an audit system. We simulated the audit service and the storage service by using two local IBM servers with two Intel Core 2 processors at 2.16 GHz and 500M RAM running Windows Server 2003. These servers were connected via 250 MB/sec of network bandwidth. Using GMP and PBC libraries, we have implemented a cryptographic library upon which our technique can be constructed. This C library contains approximately 5,200 lines of codes and has been tested on both Windows and Linux platforms. The elliptic curve utilized in the experiment is a MNT curve, with base field size of 160 bits and the embedding degree 6. The security level is chosen to be 80 bits, which means $|p|$ = 160. Firstly, we uantify the performance of our audit technique under different parameters, such as file size sz, sampling ratio w, sector number per block s, and so on. Our analysis shows that the value of s should grow with the increase of sz in order to reduce computation and communication costs. Thus, our experiments were carried out as follows: the stored files were chosen from 10KB to 10MB; the sector numbers were changed from 20 to 250 in terms of file sizes; and the sampling ratios were changed from 10% to 50%. These results dictate that the computation and communication costs (including I/O costs) grow with the increase of file size and sampling ratio. Next, we compare the performance of each activity in our authentication protocol. We have shown the theoretical results in Table 4: the overheads of “commitment” and “challenge” resemble one another, and the overheads of “response” and “authentication” resemble one another as well. To validate the theoretical results, we changed the sampling ratio w from 10% to 50% for a 10MB file and 250 sectors per block in a multi-cloud $\mathcal{P}$ = {$P1$, $P2$, $P3$}, in which the proportions of data blocks are 50%, 30%, and 20% in three CSPs, respectively. Our experimental results show that the computation and communication costs of “commitment” and “challenge” are slightly changed along with the sampling ratio, but those for “response” and “authentication” grows with the increase of the sampling ratio. Here, “challenge” and “response” can be divided into two sub-processes: “challenge1” and “challenge2”, as well as “response1” and “response2”, respectively. Furthermore, the proportions of data blocks in

Figure 4: Applying CPDR Technique in Hadoop distributed file system (HDFS).



each CSP have greater influence on the computation costs of "challenge" and "response" processes. In summary, our technique has better performance than non-Cooperative approach.

6 CONCLUSIONS

We make three key contributions in this paper, first we have proposed a Cooperative PDR technique to support dynamic scalability on multiple storage servers, and second we presented the construction of an efficient PDR technique for distributed cloud storage Based on homomorphic verifiable response and hash index hierarchy. Third we also showed that our technique provided all security properties required by zeroknowledge interactive proof system, so that it can resist various attacks even if it is deployed as a public audit service in clouds. Furthermore, we optimized the probabilistic query and periodic authentication to improve the audit performance. Our experiments clearly demonstrated that our approaches only introduce a small amount of computation and communication overheads. Therefore, our solution can be treated as a new candidate for data integrity authentication in outsourcing data storage systems. As part of future work, we would extend our work to explore more effective CPDR constructions. For a practical point of view, we still need to aPDRess some issues about integrating our CPDR technique smoothly with existing systems, for example, how to match index structure with cluster-network model, how to match index hash hierarchy with HDFS's two-layer name space, and how to dynamically update the CPDR parameters according to HDFS' specific requirements. Finally, it is still a challenging problem for the generation of tags with the length irrelevant to the size of data blocks. We would explore such an issue to provide the support of variable-length block authentication.

ACKNOWLEDGEMENT

The work on "Cooperative provable data retention for integrity authentication in multi-cloud Storage" was supported by many members. We are highly obliged and thankful to almighty who has provided us with the opportunity to thank all the kins who stood by us in the process of working on this project. First of all we would like to thank Mr. Shubham Srivastava our project guide and MR. AJAY KUMAR our project coordinator who was of utmost help to us throughout the proceedings. His valuable guidance is unmatched.

Further, we would like to thank Head of Computer Science department of Institute of Technology and Management Mr. Rajeev Ranjan Kumar Tripathi, who has been a great help and an inspiration in carrying out the present work successfully. So it's our pleasure to present a cordial thanks to him.

Last but not the least, we would like to thank our colleagues and faculty members who made this project a more fruitful and pleasant experience.

REFERENCES

[1]. G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, "Provable data possession at untrusted stores," in *ACM Conference on Computer and Communications Security*, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds. ACM, 2007, pp. 598–609.

[2]. A. Juels and B. S. K. Jr., "Pors: proofs of retrievability for large files," in *ACMConference on Computer and Communications Security*, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds. ACM, 2007, pp. 584–597.

[3]. B. Sotomayor, R. S. Montero, I. M. Llorente, and I. T. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet omputing*, vol. 13, no. 5, pp. 14–22, 2009.

[4]. 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 communi -cation netowrks, SecureComm*, 2008, pp. 1–10.

[5]. C. C. Erway, A. K¨upc¸ ¨u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *ACM Conference on Computer and Communications Security*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 213–222.

[6] H. Shacham and B. Waters, "Compact proofs of retrievability," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, J. Pieprzyk, Ed., vol. 5350. Springer, 2008, pp. 90–107.

[7] Q. Wang, C.Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics

for storage security in cloud computing," in *ESORICS*, ser. Lecture Notes in Computer Science, M. Backes and P. Ning, Eds., vol. 5789. Springer, 2009, pp. 355–370.

[8] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in *SAC*, W. C. Chu, W. E. Wong, M. J. Palakal, and C.-C. Hung, Eds. ACM, 2011, pp. 1550–1557.

[9] K. D. Bowers, A. Juels, and A. Oprea, "Hail: a high-availability and integrity layer for cloud storage," in *ACM Conference on Computer and Communications Security*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 187–198.

[10] Y. Dodis, S. P. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *TCC*, ser. Lecture Notes in Computer Science, O. Reingold, Ed., vol. 5444. Springer, 2009, pp. 109–127.

[11] L. Fortnow, J. Rompel, and M. Sipser, "On the power of multiprover interactive protocols," in *Theoretical Computer Science*, 1988, pp. 156–161.

[12] Y. Zhu, H. Hu, G.-J. Ahn, Y. Han, and S. Chen, "Collaborative integrity verification in hybrid clouds," in *IEEE Conference on the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom, Orlando, Florida, USA, October 15-18*, 2011, pp. 197–206.

[13] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep., Feb 2009.

[14] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology (CRYPTO'2001)*, vol. 2139 of LNCS, 2001, pp. 213–229.

[15] O. Goldreich, *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[16] P. S. L. M. Barreto, S. D. Galbraith, C. O'Eigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties," *Des. Codes Cryptography*, vol. 42, no. 3, pp. 239–271, 2007.

[17] J.-L. Beuchat, N. Brisebarre, J. Detrey, and E. Okamoto, "Arithmetic operators for pairing-based cryptography," in *CHES*, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds., vol. 4727. Springer, 2007, pp. 239–255.

[18] H. Hu, L. Hu, and D. Feng, "On a class of pseudorandom sequences from elliptic curves over finite fields," *IEEE Transactions on Information Theory*, vol. 53, no. 7, pp. 2598–2605, 2007.

[19] A. Bialecki, M. Cafarella, D. Cutting, and O. O'Malley, "Hadoop: A framework for running applications on large clusters built of commodity hardware," Tech. Rep., 2005. [Online]. Available: http://lucene.apache.org/hadoop/

[20] E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds., *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. ACM, 2009.

Krishna Kumar Singh:
Computer Science and Engineering Final Year Students of Institute of Technology and Management GIDA Gorakhpur, Gautam Buddh Technical University (GBTU), LUCKNOW U.P. India.

Rajkumar Gaura:
Computer Science and Engineering Final Year Students of Institute of Technology and Management GIDA Gorakhpur, Gautam Buddh Technical University (GBTU), LUCKNOW U.P. India.

Sudhir Kumar Singh:
Computer Science and Engineering Final Year Students of Institute of Technology and Management GIDA Gorakhpur, Gautam Buddh Technical University (GBTU), LUCKNOW U.P. India.