Remote Data Auditing in Cloud Computing Environments: A Survey, Taxonomy, and Open Issues

MEHDI SOOKHAK, ABDULLAH GANI, HAMID TALEBIAN, and ADNAN AKHUNZADA,
University of Malaya, Malaysia
SAMEE U. KHAN, North Dakota State University, USA
RAJKUMAR BUYYA, University of Melbourne
ALBERT Y. ZOMAYA, University of Sydney, Australia

Cloud computing has emerged as a long-dreamt vision of the utility computing paradigm that provides reliable and resilient infrastructure for users to remotely store data and use on-demand applications and services. Currently, many individuals and organizations mitigate the burden of local data storage and reduce the maintenance cost by outsourcing data to the cloud. However, the outsourced data is not always trustworthy due to the loss of physical control and possession over the data. As a result, many scholars have concentrated on relieving the security threats of the outsourced data by designing the Remote Data Auditing (RDA) technique as a new concept to enable public auditability for the stored data in the cloud. The RDA is a useful technique to check the reliability and integrity of data outsourced to a single or distributed servers. This is because all of the RDA techniques for single cloud servers are unable to support data recovery; such techniques are complemented with redundant storage mechanisms. The article also reviews techniques of remote data auditing more comprehensively in the domain of the distributed clouds in conjunction with the presentation of classifying ongoing developments within this specified area. The thematic taxonomy of the distributed storage auditing is presented based on significant parameters, such as scheme nature, security pattern, objective functions, auditing mode, update mode, cryptography model, and dynamic data structure. The more recent remote auditing approaches, which have not gained considerable attention in distributed cloud environments, are also critically analyzed and further categorized into three different classes, namely, replication based, erasure coding based, and network coding based, to present a taxonomy. This survey also aims to investigate similarities and differences of such a framework on the basis of the thematic taxonomy to diagnose significant and explore major outstanding issues.

Categories and Subject Descriptors: H.3.2 [**Information Storage and Retrieval**]: Information Storage; E.3 [**Data Encryption**]

General Terms: Security, Reliability, Performance

Additional Key Words and Phrases: Remote data auditing, cloud computing, replication, erasure coding, network coding

This work was carried out as part of the Mobile Cloud Computing research project funded by the Malaysian Ministry of Higher Education as the University of Malaya High Impact Research Grant (UM.C/625/1/HIR/MOHE/FCSIT/03).

Authors' addresses: M. Sookhak, A. Gani, H. Talebian, and A. Akhunzada, Center for Mobile Cloud Computing Research, University of Malaya, Kuala Lumpur, Malaysia; emails: m.sookhak@ieee.org, abdullah@um.edu.my, {talebian, a.adnan}@siswa.um.edu.my; Samee U. Khan, Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58108-6050; email: samee.khan@ndsu.edu; R. Buyya, Department of Computing and Information Systems, The University of Melbourne, Doug McDonell Building, Parkville Campus, Melbourne, Vic. 3010, Australia; email: rbuyya@unimelb.edu.au; A. Y. Zomaya, School of Information Technologies, Building J12, The University of Sydney, Sydney, NSW 2006, Australia; email: albert.zomaya@sydney.edu.au.

DOI: http://dx.doi.org/10.1145/2764465

ACM Reference Format:
Mehdi Sookhak, Abdullah Gani, Hamid Talebian, Adnan Akhunzada, Samee U. Khan, Rajkumar Buyya, and Albert Y. Zomaya. 2015. Remote data auditing in cloud computing environments: A survey, taxonomy, and open issues. ACM Comput. Surv. 47, 4, Article 65 (May 2015), 34 pages.
DOI: http://dx.doi.org/10.1145/2764465

1. INTRODUCTION

Cloud computing (CC) is a significant information technology (IT) shift and a new model of computing over shared computing resources, such as bandwidth, storage, servers, processing power, services, and applications [Armbrust et al. 2010, 2011]. Today, this new paradigm has become popular and is receiving a lot of attention from researchers in the academic and industrial communities. A recent survey indicates that more than 79% of organizations attempt to utilize data outsourcing because it relieves the burden of maintenance cost as well as the overhead of storing data locally [Buyya et al. 2009]. Moreover, the users are able to access information from anywhere and at any time instead of having to use dedicated machines [Wang et al. 2009; Xie et al. 2007].

Although CC offers several advantages for users, there are some security concerns that prevent a full adoption of the new technology [Zhibin and Dijiang 2012]. When users outsource data files in a distant server, the physical access to the file is actually lost and the administration of files is delegated to a cloud provider as an unreliable third party [Wang et al. 2010; Wei et al. 2014]. Although the cloud's infrastructure is much more robust and reliable as compared to the client's hardware, the data in the cloud space is still susceptible to various kinds of inside and outside threats that might risk the integrity, confidentiality, and availability of data [Wang et al. 2009; Zhifeng and Yang 2013]. Recently, various companies reported data corruption in servers with major cloud infrastructure providers and many events of cloud service outages, such as the Amazon S3 breakdown [Gohring 2008], Gmail mass deletion [Arrington 2006], sidekick cloud disaster, and Amazon EC2 services outage [Whittaker 2012; Miller 2010]. Moreover, the Privacy Rights Clearinghouse (PRC) reports more than 535 data breaches during 2013, namely, breaching of cloud-based email service provider in Epsilon [Storm 2011]; compromising of Sony Online Entertainment, Sony PlayStation Network, and Sony Pictures; stealing of customers' information on EMC's RSA; and stealing of 3.3 million patients' medical data of Sutter Physicians Services [Schwartz 2012].

After outsourcing the data to the remote clouds, the cloud users must ensure that the data remains intact. However, the conventional integrity verification methods, for example, hash functions and signatures, are inapplicable in CC because of the lack of a local data copy [Ateniese et al. 2008]. On the other hand, downloading of possibly a large-size file is impractical. The aforementioned situation worsens when users are accessing data using mobile devices. The cloud users are responsible for devising a suitable audit mechanism that can remotely verify the intactness of distant data [Ateniese et al. 2008; Chen 2013].

The remote data auditing service comprises a set of protocols designed to prove the intactness of the remote data that resides in cloud storage more reliably and efficiently, devoid of downloading the entire data. Moreover, the outsourced data is also subject to administration by unreliable third-party cloud providers [Ateniese et al. 2011]. The RDA frameworks use a spot-checking technique to validate the outsourced data, in which only a small fragment of the whole data is required to be accessed by the auditor. This technique provides either a probabilistic or deterministic assurance for the data intactness [Chen et al. 2010]. In the design and implementation of the RDA technique, the following significant properties need to be considered: (1) efficiency: to audit the data with the least possible computational complexity; (2) public verifiability: to allow for delegating the auditing process to a trustworthy Third Party Auditor (TPA) instead of the client; the goal of this property is to reduce the computational burden on the client's side; (3) frequency: to allow the verification process to be repeated as frequently as possible with different challenge messages; (4) detection probability: the probability of a potential data corruption detection; (5) recovery: the ability to restore corrupted data to the original state; and (6) dynamic update: to still be able to audit data while the cloud user is allowed to perform delete, modify, and append operations on his or her outsourced file without requiring retrieval of the entire uploaded data [Wang et al. 2010; Yang and Jia 2012].

The RDA methods are applicable for the single and distributed cloud servers [Chen et al. 2010]. In the single cloud server, such algorithms are only responsible for preventing unauthorized parties from altering the outsourced data. In other words, the auditor must check the data integrity through an RDA algorithm to detect data corruption [Erway et al. 2009; Cash et al. 2012; Wang et al. 2011; Ateniese et al. 2007; Wang 2012]. However, when data corruption is detected, a majority of the single-server RDA techniques do not have the necessary capabilities to recover data. Therefore, an RDA technique is complemented with data storage redundancy because the Data Owner (DO) is able to restore the corrupted data by using the remaining healthy servers [Chen et al. 2010].

This article delineates an extensive survey of the RDA techniques within the distributed cloud server domain and presents a basis for classifying the present and future developments within this area. The thematic taxonomy of distributed storage auditing is presented based on important factors such as security level, security measures, security requirements, update mode, and auditing mode.

The core contributions of the survey are (1) the review of the most recent RDA techniques in distributed servers that have not been sufficiently covered in previous works and (2) analysis and classification of the current RDA approaches into three different categories on the basis of the data redundancy feature, namely, replication based, erasure coding based, and network coding based. This survey also aims to investigate the similarities and differences of such schemes based on the thematic taxonomy to diagnose the significant and outstanding issues for further studies.

The complete organization of the remainder of the article is as follows. The principal concepts of distributed storage systems and cloud computing are described in Section 2. Section 3 discusses the concept of RDA and the different architectures for distributed storage systems. It also compares distributed data auditing with the single-server RDA. Section 4 shows an extensive review of the most recent RDA techniques for distributed storage systems. The strengths and weaknesses of the current RDA methods are also investigated in this section. Section 5 presents the thematic taxonomy of the current RDA approaches for distributed storage systems. The section primarily presents a comparison of the current RDA techniques by using the similarities and differences within the important parameters included in the taxonomy. Section 6 addresses the challenges and issues within the contemporary RDA approaches. Finally, Section 7 concludes the article and presents some future research directions.

2. BACKGROUND

This segment of the article describes the concepts of cloud computing and distributed storage systems. We also explain the mechanism of the RDA.

2.1. Cloud Computing

CC has emerged as the latest utility-oriented distributed computing model and has been envisioned as a significant shift of IT, with the aim of augmenting abilities of the client devices by providing access to a shared pool of rented platforms, applications, and infrastructures without having to actually own them. The different service models of the cloud offer on-demand, affordable, rapid elasticity, ubiquitous resource access and measured service [Whaiduzzaman et al. 2014; Hüfer and Karagiannis 2011]. The cloud systems have the capability of conveniently adjusting the virtual allocated resources on the basis of the current requirements with a minimal managerial effort and service interruption. Such elastic characteristics reduce the wastage of resources in case of overprovisioning [Rong et al. 2013; Aceto et al. 2013].

The cloud service models rely on a pricing model of pay as you go that charges the clients on the basis of the amount of usage and some service metrics [Zhifeng and Yang 2013]. For example, the Dropbox service can be measured as gigabytes per year. The CC also has led to the appearance of a new type of collaboration and communication service by creating social networks and online communities, which facilitates scientists' constructing research communities by sharing data and analysis tools [Barga et al. 2011]. The virtualization of resources is the core technology of cloud computing to inculcate a vision of infinite resources to the clients [Fernando et al. 2013].

From the perspective of deployment, the CC is classified into four modes: public, private, hybrid, and community clouds, which are detailed next [Zissis and Lekkas 2012]. (1) Public cloud: In the public cloud, service providers deliver different applications as service and facilitate clients by providing access to more computing resources through centralized cloud servers over the Internet, such as Amazon Web Services, Google App Engine, Microsoft Azure platform, Salesforce, and Aneka [Fox et al. 2009]. Amazon Web Services allow users to store data in Simple Storage Services (S3) [Kristensen 2009], the Google App Engine offers deployment platforms and hosts web applications in Google's data centers [wesley 2011], the Microsoft Azure platform provides a powerful platform for building and deploying web applications in Microsoft data centers, and Aneka is used as a platform to build applications and deploy them on private or public clouds [Calheiros et al. 2012]. (2) Private cloud: The services and infrastructure are used and managed entirely by a solo organization. (3) Community cloud: The services and infrastructure are shared by a set of organizations with common interests or objectives that are managed either internally or by a trusted third party. (4) Hybrid cloud: It simply denotes the combination of clouds having different providers [Zhang et al. 2010].

Cloud service providers offer three forms of service models: Software as Service (SaaS), Infrastructure as Service (IaaS), and Platform as Service (PaaS) [Vaquero et al. 2011]. Users of the SaaS layer are allowed to use any type of software from their corresponding mobile devices remotely. For instance, Microsoft Live Mesh offers file and folder sharing among various computing devices. The PaaS model provides developers with a runtime environment according to their specific requirements [Gonçalves and Ballon 2011]. The PaaS also provides a programming framework, libraries, and toolkits for developers to enable them to develop, deploy, and maintain applications. Some of the well-known PaaS services like Amazon Elastic MapReduce (EMR), Google App Engine, and Microsoft Azure are available in the market. The IaaS offers computation, storage, and networking in the form of a flexible Virtual Machine (VM) to business users. For instance, S3 (Simple Storage Service) and EC2 (Elastic Cloud Computing) are two noticeable samples of such services [Jing et al. 2013; Subashini and Kavitha 2011].

2.2. Distributed Storage Systems

Distributed storage systems are created by combining networking and storage to allow users to remotely store data and provide novelty services, such as archiving, publishing, federation, and anonymity. The advances in networking technology have directly caused the emergence of new distributed storage systems. For example, a new generation of distributed system reappeared by evolving the networks from the private

Local Area Networks (LANs) to public global Wide Area Networks (WANs), such as the Internet [Horn 2001; Himmel and Grossman 2014].

Distributed storage systems are classified into the following groups based on the application's functional requirements: (1) Archival: The archival system is introduced as a nonvolatile storage, in which the users are able to store, retrieve, and back up files. The stored data in such systems rarely needs to make updates and has write-once and read-many workloads. Examples of archival storage include a large-scale peer-to-peer persistent storage utility (PAST) [Druschel and Rowstron 2001] and Cooperative File System (CFS) [Dabek et al. 2001]. (2) Filesystem: The systems that fall in this category offer persistent nonvolatile storage with a file system for the users and permit the applications to use storage without having to modify the rebuild, such as the SUN Network File System (NFS) [Sandberg et al. 1985]; Coda [Mahadev 1990]; serverless network file system (xFS) [Anderson et al. 1996]; Federated, Available, and Reliable Storage for an Incompletely Trusted Environment (Farsite) [Adya et al. 2002]; and read/write peer-to-peer file system (Ivy) [Muthitacharoen et al. 2002]. (3) Publish and Share: The main aim of such a service is to support the share and publish purpose. Contrary to the previous two models, where the objectives of the storage service are persistent, the publish and share model is unstable and reliant on the reputation of the shared or published file. Systems that fall into this class include Haven [Dingledine et al. 2001], Freenet [Clarke et al. 2001], Publius [Waldman et al. 2001], Gnutella [Oram 2001], and BitTorrent [Hasan et al. 2005]. (4) Performance: The applications that need a high level of performance belong to this category, and most of them are categorized as Parallel File Systems (PFSs). Such systems operate within the nodes that are interconnected by a high-bandwidth and low-latency network [Venugopal et al. 2006]. Example systems that fall into this category are the Portable Parallel File System (PPFS) [Huber Jr et al. 1995], Zebra [Hartman and Ousterhout 1993], Parallel Virtual File System (PVFS) [Ross and Thakur 2000], General Parallel File System (GPFS) [Schmuck and Haskin 2002], and Frangipani [Thekkath et al. 1997]. (5) Federation Middleware: This system enables organizations to integrate a large number of storage systems through the Internet. The federation middleware is responsible for offering a homogeneous interface, processing of data, and managing of replicas [Venugopal et al. 2006]. (6) Hybrid: In the last group, a new storage system is created by combining the previous system categories. Examples include Google File System (GFS) [Ghemawat et al. 2003] and OceanStore [Kubiatowicz et al. 2000].

The architecture of a distributed storage system can mainly be classified into two groups: client server and peer to peer. In the client-server architecture, each entity has to be the property of either a client or a server, and the server is in charge of authentication, replication, backup, and servicing requests to clients. Such an architecture is used widely by distributed storage systems [Sandberg et al. 1985; Mahadev 1990; Thekkath et al. 1997; Vazhkudai et al. 2005]. On the contrary, in a peer-to-peer architecture, every participant has the capability to behave as a server and a client. FastTrack [Hasan et al. 2005; Ding et al. 2005], Clippee [Albrecht et al. 2003], and eDonkey [Tutschku 2004] are examples of such an architecture.

3. REMOTE DATA AUDITING

Today, most individuals and organizations are motivated to reduce the cost and time involved in procurement and maintenance of local data storage infrastructure by outsourcing the data to the cloud. In cloud computing, the Cloud Service Provider (CSP) is in charge of managing the cloud storage services. As a result, the DOs are unable to maintain their possession and direct control over the uploaded data and instead the data is exclusively managed by an untrustworthy third party. On the other hand, the

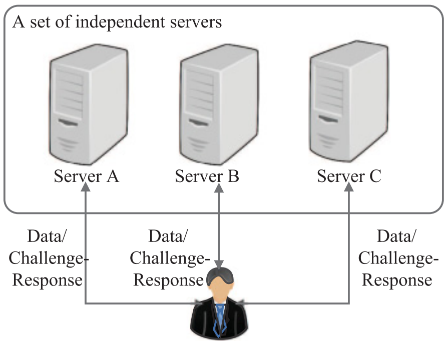

Fig. 1. Multiserver audit architecture.

CSP or any insider adversary is able to maliciously manipulate data content without user consent or knowledge [Chen et al. 2010].

The RDA technique attempts to sample data on the cloud and analyze it with several criteria such as integrity, correctness, and validity as benchmarks to ensure the reliability and trustworthiness of cloud service providers [Wang et al. 2010]. This section details the architecture of the RDA for the distributed servers and compares the distributed data auditing with the single-server RDA.

3.1. Architecture of Remote Data Auditing for Distributed Servers

The RDA schemes for distributed cloud servers often consist of four main entities: (1) Data Owner: the person who uploads his or her data to the cloud space and later might perform delete, insert, and append operations on the outsourced data. (2) Cloud Service Provider: Has a tremendous amount of computing resources and stores and manages the DO's data. The CSP is also responsible for managing cloud servers. (3) Third Party Auditor: In order to alleviate the computation burden on the DO's side, the auditing process is often assigned to a TPA with adequate skills and capabilities to accomplish the auditing task on behalf of the DO. The TPA's role is particularly important when DOs possess relatively poor computing devices in terms of processing power, storage space, and bandwidth. While the TPA is regarded as a trustful and reliable entity, it might be inquisitive at the same time. Consequently, one significant countermeasure during data auditing is to prevent the TPA from obtaining knowledge of the DO's data content and to protect privacy of data. (4) User (individual or enterprise): Is enrolled and authenticated by the DO and permitted to have a predetermined type of access to the outsourced data [Koo et al. 2013; Sood 2012; Yu et al. 2012]. The RDA architecture for distributed storage systems is classified into three categories: multiserver, single cloud and multiserver, and multicloud and multiserver, which we detail next.

(1) Multiserver model: In this model, the DO distributes multiple copies of the data among several servers and separately checks each of them. Figure 1 shows the architecture of the multiserver data auditing model.
(2) Single cloud and multiserver: In this model, all of the servers are distributed within a single cloud where the CSP is in charge of managing the servers. As is shown in Figure 2, the DO and the TPA are directly connected to the CSP rather than all of the servers.
(3) Multicloud and multiserver: Instead of a single cloud, the DO outsources the data among multiple clouds. Similar to the previous model, one of the CSPs, namely, the organizer, is responsible for managing all of the servers and the other CSPs. As

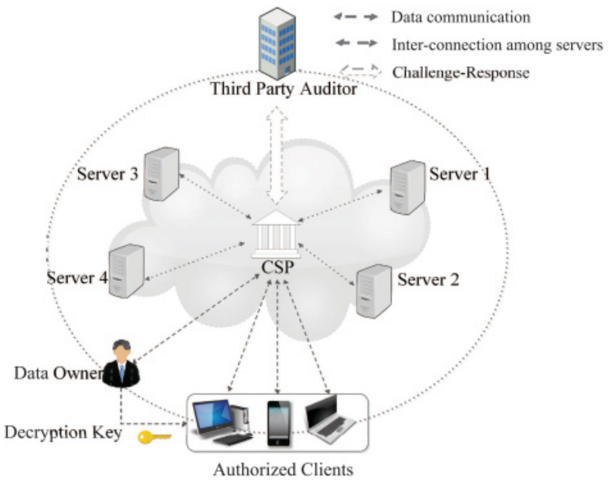

Fig. 2. Single cloud and multiserver audit architecture.

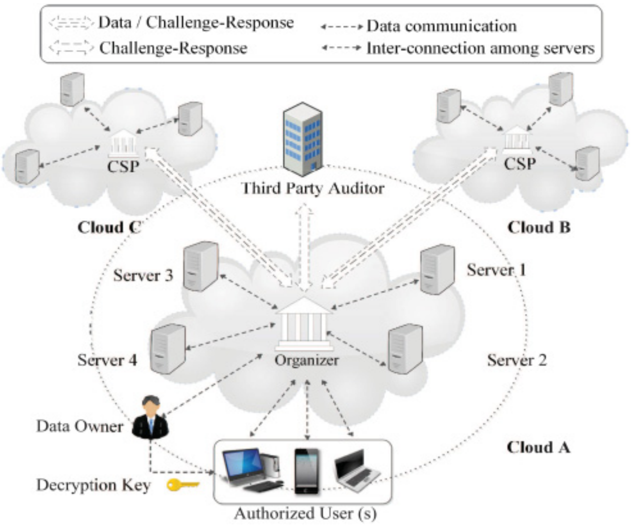

Fig. 3. Multicloud and multiserver audit architecture.

shown in Figure 3, the organizer that is directly connected to the owner receives data and a challenge from the DO to distribute among the clouds and the servers. Moreover, the organizer aggregates the received proofs from the servers and sends them to the DO.

A typical RDA service works according to the following essential response-challenge procedure: First, the DO performs a preliminary process on his or her file to generate some metadata to be passed to the TPA. Hereafter, the DO does not need to be engaged in the rest of the auditing process. In order to verify the integrity and correctness of the remote data residing on the cloud, the TPA selects a random index of the outsourced data as a challenge message and directs that message to either the organizer or the CSP (in case the TPA is not supported by auditing service architecture, the DO him- or herself must generate the challenge). When the organizer or the CSP receives the challenge, it is distributed among the servers, and then the organizer computes the corresponding response by aggregating the received messages from the servers. After receiving a response from the organizer or the CSP, the verification is carried out by the auditor to ensure the reliable placement of the file in the cloud storage. It is worth mentioning here that to lessen the computational cost of the auditing process, only a minor portion of the entire data is inquired about [Xiao et al. 2012].

3.2. Single Server Versus Multiservers

In a single server, the RDA techniques are classified into three groups: integrity based, recovery based, and deduplication based [Sookhak et al. 2014b]. The first category of RDA methods in the single servers is called the integrity-based schemes, in which the auditor is only permitted to validate the correctness of the outsourced data directly or by using a third party. Examples of such schemes include Provable Data Possession (PDP) [Ateniese et al. 2007], Proxy PDP [Wang 2012], Scalable PDP [Ateniese et al. 2008], Dynamic PDP [Erway et al. 2009], Efficient PDP [Hanser and Slamanig 2013], Robust PDP [Curtmola et al. 2008a], Efficient and Secure PDP [Yang and Jia 2012], and DRDA [Sookhak et al. 2014a]. The second category of RDA schemes is the recovery-based models that are capable of verifying the data integrity and recovering the corrupted data when an error is detected. In other words, aside from verifying data integrity, such models support forward error-correcting codes (FECs) by using the Reed-Solomon erasure-correcting code [Plank 2005]. The core difference between the recovery-founded and integrity-based approaches is that in the recovery-based approaches, the all of the client's data must be placed on the server. However, the integrity-based approaches are only responsible for ensuring that most parts of the outsourced data are in the remote server. Therefore, a minor fragment of data might be missed [Cash et al. 2012]. Furthermore, recovery-based methods save a redundant encoded form of the client's data on the server [Küpçü 2010]. Examples of recovery-based methods include Proof of Retrievability (POR) [Juels and Kaliski 2007], compact POR [Shacham and Waters 2008], public POR [Yuan and Yu 2013a], scalable POR [Stefanov et al. 2012], practical dynamic POR [Shi et al. 2013], and fair dynamic POR [Zheng and Xu 2011]. Deduplication-based approaches are the last group of RDA that facilitate the integrity and efficiency of data in a single server by removing data redundancy and increasing data storage optimization. Examples of deduplication-based methods include Proof of Storage with Deduplication (POSD) [Zheng and Xu 2012], Improved POSD [Shin et al. 2012], and Public Auditing with Deduplication (PAD) [Yuan and Yu 2013b].

Currently, individuals and organizations prefer to store data on distributed servers, because the single-server setting has no capability to recovery the data properly when data corruption is detected. For instance, in deep archival applications that use peer-to-peer storage systems [Maniatis et al. 2005], the third party is responsible for managing the data. Therefore, DOs need RDA to verify the integrity and correctness of the large archival datasets, which makes the single-server auditing methods prohibitive. This is because most of the aforementioned RDA approaches are inapplicable to such systems or incur huge computation and communication overhead on the client and server. The RDA must be supplemented by storing data redundantly on multiple servers [Chen et al. 2010].

4. STATE-OF-THE-ART RDA SCHEMES FOR DISTRIBUTED CLOUD SERVERS

The RDA schemes employ various techniques to protect the integrity of the outsourced data for distributed storage systems. This section comprehensively surveys the state-of-the-art RDA methods for distributed storage systems. We classify the survey of RDA algorithms based on the data redundancy feature and critically analyze the surveyed techniques to investigate the strengths and weaknesses of such methods.

4.1. Replication-Based Remote Data Auditing

When DOs store data in an unreliable storage system, as is the case in the cloud and mobile cloud computing paradigms, redundancy plays a fundamental role in improving the reliability against data failures. The simplest and most common way to achieve the aforementioned goal is to use a replication technique in which multiple copies of data are outsourced within the distributed storage systems [Ying and Vlassov 2013]. Whenever a data corruption is detected, the client can use an intact copy of the file with size $|f|$ from any of the r unaffected servers. The main disadvantage of the replication method is that the storage cost is $r|f|$. This is because during the repair phase, the client must retrieve a replica of size $|f|$, and the communication overhead of replication in the recovery mode is equal to one [Agrawal and Jalote 1995].

Although the implementation of this method is relatively straightforward, there is no strong evidence to prove that the cloud actually stores multiple copies of the data files [Chen et al. 2010]. In other words, the replication-based storage systems are vulnerable to collusion attacks in which the servers only store a single copy of the file while appearing to store multiple copies of the file [Curtmola et al. 2008b]. For example, in peer-to-peer networks, servers perform a freeloading attack with the aim of using disproportionately more of the system's resources without contributing a proportionate quantity of resources back to peers [Osipkov et al. 2006]. As a result, the DOs encounter a decline in the durability and availability of the file and the CSP has more storage space to sell to the users [Curtmola et al. 2008b].

The naive way to overcome the collusion attack is to apply a single PDP method [Ateniese et al. 2007] t times for t different servers. However, as an identical copy of the file is stored on all of the t servers, the servers can collude and pretend that t copies of files are stored while only a single copy is stored in reality. Moreover, the computational cost on the client for the preprocessing of the file is t times greater than the computational burden when the single PDP method is used. Therefore, such a method is inapplicable for distributed storage systems, particularly for larger values of t. The remainder of this section critically analyzes several replication-based data auditing techniques.

Curtmola et al. [2008b] were the first to address the collude attack in the replication-based schemes by introducing a provably secure scheme, called Multiple Replica Provable Data Possession (MR-PDP), to store a large number of copies of files. The integrity of the copies was verified through a challenge-response protocol. The MR-PDP is an extension of a previous work (single-replica PDP scheme [Ateniese et al. 2007]) for use in the multiserver environment.

In the MR-PDP scheme, the client generates t unique replicas (R_s) by encrypting the original file and masking the blocks of the encrypted file by using a random value (R_u) for each of the replicas. Thereafter, the client uses a decrypted file to create a tag for each block on the basis of the RSA signature. The client then outsources a generated replica and a set of tags on each server as

$$T_i = (h(\upsilon||i).g^{b[i]})^d \; mod \; n, \tag{1}$$

where T_i is the tag for the i^{th} data block ($b[i]$), n is the number of blocks, d and υ are the clients' private key, and g is the public key. To check for the data possession in each of the servers, the clients select a random portion of the data blocks and compute a random number g_s to prevent the server from using a previous challenge message. When the challenge message is received, the server computes a proof on the basis of the challenge and the corresponding tags. Finally, the DO is able to verify the received proof message based on the random value (R_u).

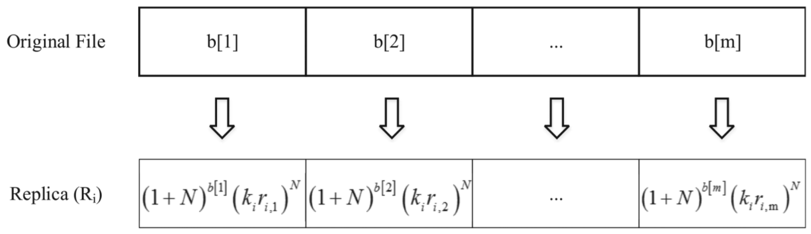
Original File
b[1]
b[2]
...
b[m]
Replica (R$_i$)
$(1+N)^{b[1]}(k_i r_{i,1})^N$
$(1+N)^{b[2]}(k_i r_{i,2})^N$
...
$(1+N)^{b[m]}(k_i r_{i,m})^N$

Fig. 4. Generating a unique replication in the DMR-PDP scheme.

Although the MR-PDP method is suitable for checking the integrity and availability of distributed servers, the DO is unable to delegate the auditing to the TPA because the MR-PDP only supports private verification. Moreover, to update a block of the file, the DOs must retrieve the entire data, which imposes a huge computation and communication overhead on the client and server.

Barsoum and Hasan [2010] proposed the Efficient Multi-Copy Provable Data Possession (EMC-PDP) scheme to devise a public verification method for the replication-based storage based on the Boneh-Lynn-Shacham (BLS) homomorphic linear authenticators [Shacham and Waters 2008]. The EMC-PDP is introduced in two different versions: deterministic (DEMC-PDP) and probabilistic (PEMC-PDP). In the deterministic version, all of the file blocks are verified. The probabilistic scheme relies on the spot-checking approach in which only a random fraction of the file is checked. Even though the DEMC-PDP provides a stronger security guarantee, it is achieved at the expense of a higher storage overhead on the client and the server.

The main idea behind the EMC-PDP method is to generate a unique replication of file (R_i) by attaching a replica number i to the original file F. Therefore, the generated replica is encrypted with a strong diffusion property of an encryption scheme, such as the Advanced Encryption Standard (AES). The DO also generates a distinctive tag for each block of replicas by using the following equation and distributes them along with the replicas among the servers:

$$T_{i,j} = (h(F_{id})u^{b[i][j]})^d, \tag{2}$$

where F_{id} indicates a unique fingerprint for each file that is generated by attaching the filename to the number of blocks and the number of replicas, i indicates the replication number, j is the block number, d is the client's private key, and u is a generator for a bilinear group mapping G. Finally, the authorized users are able to validate the data possession of all of the replicas or a random subset by using a challenge-response protocol. The experimental result shows that the EMC-PDP is more efficient than the MR-PDP scheme [Curtmola et al. 2008b] in the following ways: (1) it supports authorized users, (2) the storage cost for the EMC-PDP is six times less than that of the MR-PDP, (3) the required bandwidth is much less than the MR-PDP due to the application of the aggregation strategy, and (4) the PEMC-PDP is the most efficient protocol in terms of computational cost.

Although Barsoum and Hasan [2010] used BLS homomorphic linear authenticators to delegate the auditing task to the trusted third party, the client must re-encrypt and upload the entire replicas to the servers to update a block of the file, which incurs high overhead on the client and server sides.

Mukundan et al. [2012] proposed a Dynamic Multi-Replica Provable Data Possession (DMR-PDP) scheme to verify the integrity and completeness of multiple copies. The authors utilized a probabilistic encryption algorithm, namely, the Paillier encryption scheme, to generate distinct copies of the original file $F = \{b[1], b[2], \ldots, b[m]\}$. Figure 4 shows the operation to compute the i^{th} replication using the following equation:

$$R_i = \left\{(1+N)^{b[j]}\left(k_i r_{i,j}\right)^N\right\}_{j=1}^{m}, \tag{3}$$

where j is the index of the block, i is the number of replicas, k_j represents a random number that is used to identify the number of replicas, $r_{i,j}$ indicates a random number that is used for the Paillier encryption scheme, and N is the owner's public key.

The DO generates a unique tag for each data $b[i]$ block through the BLS signatures by using $T_i = (h(F).u^{b[i].N})^d$, where h represents a hash function, d is the client's private key, and u is a generator for the bilinear group mapping G. The main contribution of the DMR-PDP method is the introduction of a data block modification operation in which the DO calculates the difference between the blocks ($\triangle b\,[j] = b'\,[j] - b[j]$) and encrypts $\triangle b\,[j]$ using the Paillier encryption $En\,(\triangle b\,[j]) = (1 + \triangle b\,[j]N)\,r^N$. After generating a tag for the new block $T_j = (h(F).u^{b'[j].N})^d$, the DO sends $En\,(\triangle b\,[j])$ and T_j along with a random challenge to ensure the integrity of the modification operation. When the request message is received, the servers make all of the copies of the file up to date by carrying out a homomorphic addition operation, without needing to re-encrypt all of the files ($En\,(b'\,[j]) = En\,(b\,[j])\,.En\,(\triangle b\,[j])$).

Yan et al. [2012] built a replication-based remote data auditing framework for distributed systems by using the homomorphic verification response (HVR) and hash index hierarchy (HIH), called the Cooperative Provable Data Possession (C-PDP). The HIH is a hierarchical structure that is used to present the relationships among the data blocks of various storage service providers and includes three layers: Express, Service, and Storage Layers. In the Express Layer, the original file is divided and distributed among all of the service providers in the Service Layer. The next layer that is in charge of managing cloud storage services fragments the received file data and stores it in the storage servers in the Storage Layer. The last layer that is constructed as a data structure to store a set of block and tag pairs allows the auditor to check the outsourced data integrity. The HVR is another fundamental technique in the C-PDP scheme that takes care of combining the generated responses from numerous cloud providers into one response based on the sum of the challenges. As a result, the communication overhead is reduced and the privacy of data is preserved by hiding the outsourced data location in the distributed storage system.

In the architecture of the C-PDP scheme, an independent server or one of the existing CSPs is assumed as an organizer, who has responsibility for managing all of the CSPs, initiating and organizing the verification process, and communicating directly with the client. Moreover, after a challenge is issued by the client, the organizer aggregates all of the responses received from the CPSs into one response by using the HVR technique, to be sent to the client. Even though the C-PDP scheme has several advantages, it must be assumed that the organizer is a trusted entity. Moreover, a heterogeneous structure of the proposed scheme leads to a high communication load due to the intercommunication between various cloud servers.

Barsoum and Hasan [2011] discussed the problem of verifying the multiple copies of a remote file when the file is liable to update operations, such as modification, insertion, and deletion. Specifically, the authors proposed two Dynamic Multi-Copy Provable Data Possession (DMCPDP) methods: Tree-Based DMCPDP (TB-DMCPDP) and Map-Based DMCPDP (MB-DMCPDP), which are based the Merkle Hash Tree (MHT) [Merkle 1980] and the map-version table to support the dynamic data outsourcing.

In the TB-DMCPDP method, the original form of the MHT is used for each of the replicas, and then the root of each of the trees is placed as a leaf to construct a unique tree, namely, the directory of the MHT. The main concept behind such an approach is

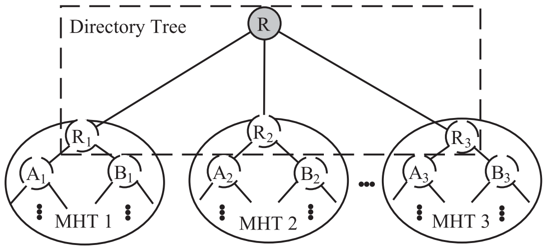

Fig. 5. Directory tree for multiple replicas in TB-DMCPDP scheme.

to verify the integrity of all of the replicas in a hierarchical manner using a directory MHT in which the leaf nodes of the tree are the root node of each file copy's MHT. By using such a structure, the auditor only needs to store $M = hash(ID_F + R)$, where ID_F indicates a unique file identifier for each of the replicas and R is the root value of the directory tree. Figure 5 shows an example of the directory MHT.

When the client requires modifying a block of the outsourced data ($b[i]$), a verification tag of the new block is computed and sent to the servers along with a copy of the data block for each server. Upon receiving the modification request at each server, the old block is replaced by a new version and reconstructs the MHT based on the new data block. The CSP then updates the directory tree and calculates the authentication paths of the updated blocks as a proof of modification. Finally, the CSP sends the authentication path, including a set of nodes from the modified block to the root of each of the MHTs. The DO is able to verify the modify operation by rebuilding the root of the directory tree on the basis of the received authentication path. However, modify, insert, and delete operations impose significant overhead on the server side in the TB-DMCPDP method because the CSP must rebalance all of the MHT structures to perform such operations. On the other hand, storing several MHTs on the servers incurs an enormous storage cost when the size of files is dramatically increasing.

To address the storage and computation overhead, Barsoum and Hasan [2011] implemented a novel data structure known as the map-version table. The table that is used to check the outsourced data integrity contains three columns: Serial Number (SN), Block Number (BN), and Version Number (VN). The SN basically represents the actual (or physical) position of the block in the file, while the BN shows the logical location of the block in the file. The VN for a block indicates the number of dynamic operations applied to that block so far. Upon outsourcing a data block for the first time, the VN is set to one, and for every dynamic operation of this block, the VN is incremented by one.

The map-version table needs to be stored in the local storage of the DO, who is responsible for updating the table during the modify, insert, and delete operations. For example, when the DO decides to insert a data block after position i, a new row must be appended to the table (after the last entity of the table as an actual position) with these characteristics $(SN,\ BN, VN) = (i+1,\ Max\,(BN)+1,\ 1)$. Meanwhile, to delete a data block from the outsourced data, the DO is only required to delete the requested block from the map-version table. An example of a map-version table is shown in Figure 6.

Although the conducted security analysis proves the security of both of the algorithms, the performance analysis reveals that the MB-DMCPDP is more efficient than the TB-DMCPDP. This is because the TB-DMCPDP extends a single-copy DPDP scheme. Such an extension introduces a large amount of storage overhead to the server, as well as a sizable computation overhead to the CSP and the client. Moreover, in the map-version-based approach, the update operation is performed in an efficient way that leads to fewer computational and communication costs. The MB-DMCPDP is also

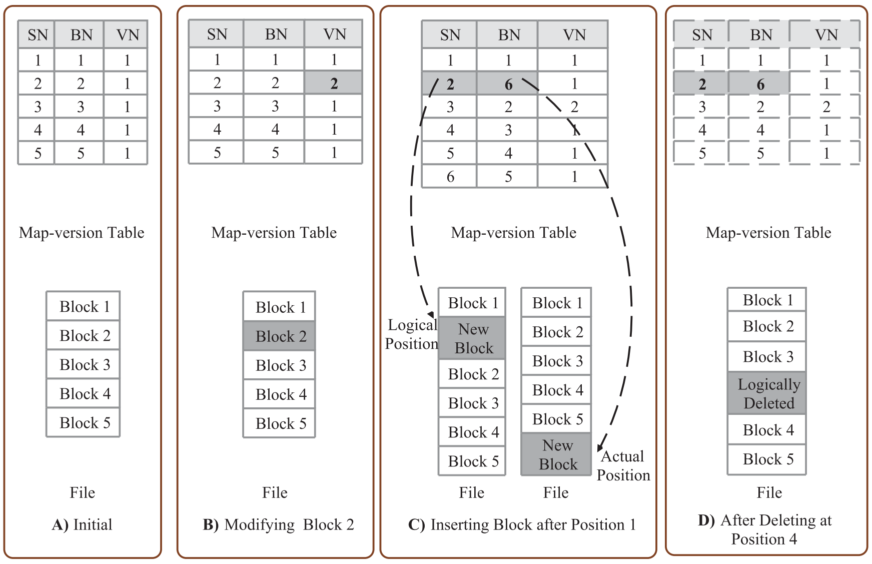

Fig. 6. Map-version table during different file operations.

efficient when there are many verifiers connected to the CSP, as the challenge-response phase requires a lower computational time. Nevertheless, the main disadvantage of the map-version table is that the required storage to keep the table is more than the MHT directory.

Previous works on data auditing in the distributed scenario, such as Curtmola et al. [2008b], Bowers et al. [2009a], Barsoum and Hasan [2010, 2011], and Zhu et al. [2010], disclose the internal architecture of the system to the clients. The reason is that clients must essentially preprocess the files as part of a scheme procedure that entails a computation overhead commensurate with the number of existing servers. Therefore, an excessive amount of computational burden is imposed on the client side. Other than the aforementioned methodology, there is no way to check whether the CSP actually stores the exact number of file copies as agreed on by the client. When the client is aware of the internal architecture of the cloud, the CSP is unable to efficiently improve the architecture without the need to inform the client. Etemad and Küpçü [2013] proposed an extension of the DPDP [Erway et al. 2009], called DR-DPDP, that tends to be transparent to the client to improve scalability, availability, load balancing, and fault tolerance in RDA schemes. Such a transparency permits the CSP to flexibly manage servers and still be able to prove the possession over the outsourced files to the client. From the client's perception, only a single server exists. Consequently, the clients only need to follow the same DPDP protocol. Moreover, the clients are ensured that at least one intact copy of the files is always stored in the cloud.

Similar to the C-PDP scheme [Yan et al. 2012], in the architecture of the DR-DPDP, one of the servers is considered as a logical entity, called the organizer, who is in charge of connecting servers to the clients. The servers are only able to communicate with the organizer, and there is no internal communication among the servers. The central idea behind such an architecture is to break a large authenticated skip list into several smaller sublists. The top sublist is provided to the organizer, and the other low-level parts are stored by the other servers, which causes improvement in the scalability. Each sublist may also be copied to more than one server to enhance availability and reliability.

Table I. Comparison of the Replication-Based RDA Methods

Methods	Objectives	Drawbacks
MR-PDP [Curtmola et al. 2008b]	Extends the PDP scheme to generate multiple replicas for distributed servers without encoding each replica separately	Costly dynamic update Incurs unnecessary overhead on the clients due to performing pre-computation step
EMC-PDP Barsoum and Hasan 2010]	a. Supports the dynamic auditing; b. Resilient against colluding servers attack; c. Less storage overhead than MR-PDP scheme	a. Costly dynamic update; b. Each verification needs the whole file to be transmitted; c. The client needs to perform precomputation based on the number of servers/replicas
TB-DMCPDP [Barsoum and Hasan 2011]	Supports the dynamic auditing by using a type of MHT tree	a. More storage and communication overhead than MB-DMCPDP; b. Cloud needs to store MHT for each file, which affects system performance
MB-DMCPDP [Barsoum and Hasan 2010]	Reduces the storage and computation overhead on client side and server side by introducing a map-version table structure	The clients need to perform precomputation based on the number of servers/replicas that incur unnecessary overhead on them
DMR-PDP [Mukundan et al. 2012]	Introduction of efficient dynamic data update	The performance of system is clear
C-PDP [Yan et al. 2012]	a. Supports the batch auditing for auditing multiple clouds; b. Supports the dynamic data auditing	Its performance depends on existing trusted server as organizer
DR-DPDP [Etemad and Küpçü 2013]	a. Hides the internal architecture of the system from clients; b. Ensures that the CSP stores the right number of replicas	a. Only provides a probabilistic proof of possession; b. Lacks flexibility on dynamic data updates

In the course of the upload phase, the client must divide the input file into n blocks and generate a unique tag for each of the blocks before transmitting the data to the organizer. When the file is received, the organizer splits the file in some partitions and sends them to an agreed-upon number of servers. Each server then constructs the corresponding part of the rank-based authenticated skip list and returns the root value as a response to the organizer. In the last step of the uploading phase, the organizer rebuilds a rank-based authenticated skip list and returns the root value to the auditor. In general, all of the algorithms in the DR-DPDP scheme include three steps: (1) Find the root sublist that contains the requested block. The organizer is responsible for searching the skip list to find the server that stores the blocks. (2) Send the command to the servers. All commands must be sent to the servers through the organizer and the server executes the operations in parallel. (3) Construct the result. Once all of the servers send the partial results to the organizer, the received results are compiled and the authenticated skip list is updated.

Table I compares the replication-based RDA methods.

4.2. Erasure-Coding-Based Remote Data Auditing

The Maximum Distance Separable (MDS) code is a form of data-repairing technique that is used to achieve a reliable distributed storage system. In the MDS code, a given file f of k blocks must be encoded into n blocks, as the original file is retrievable from any k blocks out of the n blocks. The erasure code technique is constructed on the basis of the MDS code and compared to replication, which provides more reliability for the same redundancy [Weatherspoon and Kubiatowicz 2002; Changho and Ramchandran 2010; Mingqiang and Jiwu 2010].

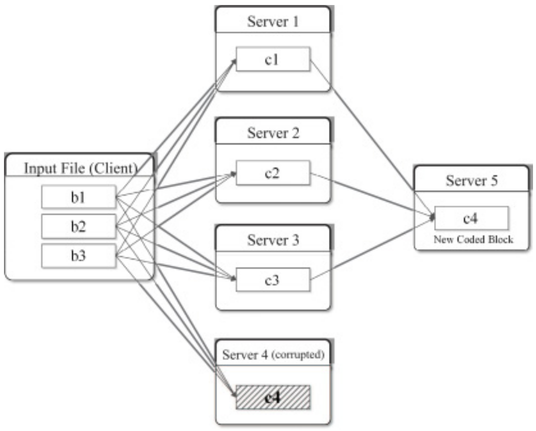

Fig. 7. Erasure-coding-based distributed storage system.

When a client detects that a block of file is corrupted, he or she is able to utilize the remaining intact blocks to recompute the codes of the corrupted block. Therefore, the storage overhead across n servers within the erasure code technique is equal to $n \times |f|/k$. This is due to the fact that to recover the corrupted coded block, only k servers are sufficient. However, compared to the replication-based technique, the communication overhead of such a method is higher because the client wants the entire file to be reconstructed by downloading at least k blocks to create a new block. Therefore, an erasure-coding-based solution for data recovery requires that at least k out of the n coded blocks remain intact. The remainder of this section reviews several erasure-coding-based data auditing techniques. Figure 7 indicates an erasure-coding-based distributed storage system in which the original input file that includes three blocks ($b[1], b[2], b[3]$) is encrypted using the (3:2) erasure code. Each of the codes is stored in a server so that the lost data can be recovered from two of the intact servers.

Bowers et al. [2009a] were the first to propose an erasure-coding-based remote data auditing for distributed storage systems, called the High-Availability and Integrity Layer (HAIL). The HAIL is designed based on the Test-and-Redistribute (TAR) strategy, in which the DO detects the file corruption using the Proofs of Retrievability (POR) scheme [Juels and Kaliski 2007] in each of the servers and proceeds to reallocate the resources when necessary. When a fault is detected in a server, the client recovers the corrupted block based on the erasure coding. The HAIL relies on three fundamental coding constructions: (1) Dispersal code: Instead of replicating files across servers, each file is distributed through the erasure code technique. The main idea behind the dispersal coding is to implement a new cryptosystem based on the universal hash functions (UHFs), pseudo-random function, and error-correcting code to achieve an error-correcting code and corruption-resilient Message Authentication Code (MAC) at the same time. (2) Server code: When the data blocks are received by the servers, the blocks need to be encoded with an error-correcting code to protect the blocks against low-level corruption, if the integrity checks fail. (3) Aggregation code: By using such a code, the responses from all of the servers to the received challenge including the multiple MACs are combined into a single composite MAC.

Figure 8 shows the dispersal coding technique in the HAIL scheme. To distribute a file of k data blocks across n servers ($n > k$), the client first encrypts the data blocks and spreads the blocks across k servers ($S_1,\ S_2\ , \ldots , S_k$). Upon receiving the data blocks, each of the blocks is encrypted using the server code with an error rate ϵ_c to generate the parity codes for each server. Thereafter, the dispersal code is carried out on the servers

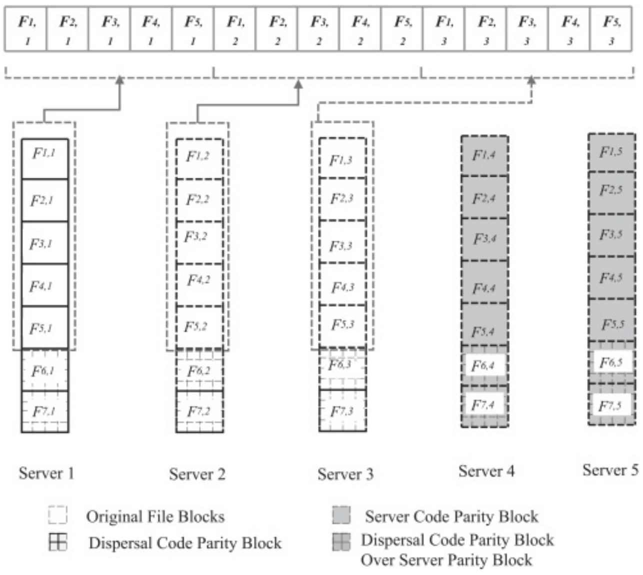

Fig. 8. Encoding of the original file and outsourcing to the servers in HAIL scheme.

to generate the parity blocks on the rest of the servers ($S_{k+1}, S_{k+2}, \ldots, S_n$). Finally, the MAC of each of the matrix rows is computed using the client's shared secret key and embedded in the parity blocks of the dispersal code to overcome a creeping-corruption attack in which any entity has the capability to verify the integrity of the outsourced data [Schwarz and Miller 2006].

Wang et al. [2012] devised a flexible and lightweight auditing scheme, named the Secure Distributed Storage (SDS), based on the Homomorphic Token [Sadeghi et al. 2010] and the Reed-Solomon erasure-correcting code [Plank and Ding 2005], to guarantee the data integrity and availability in an erasure code distributed storage system. Unlike previous works that merely determine whether the file has been tampered with, the SDS supports error localization techniques to determine the location of the errors when data corruption is detected. Such an ability is of great importance because it helps in recovering from the data errors and protects against the potential external attacks. The SDS scheme builds on the following four main components:

(1) Data Dispersal: To distribute the original file across $n = m + k$ servers, the file needs to be divided into m blocks and encrypted by the (m, k) Reed-Solomon erasure code to extend the k parity blocks to the original file in such a way that the original file is retrievable from any m out of the $m + k$ data and parity blocks. As a result, by dispersing $m + k$ data blocks that are developed from a Vandermonde matrix [Plank and Ding 2005], over different servers, the original file has k blocks fault tolerance with the storage overhead k/m.
(2) Token Precomputation: Before distributing the file in the servers, a certain number of tokens must be generated based on a random subset of data blocks. Therefore, the client randomly selects r sectors ($\{I_k\}_{k=1}^{r}$) of the jth block of the extended client's file G by using pseudo-random permutation with K_{chal} as a key and generates r coefficients $\{\alpha_k\}_{k=1}^{r}$ by pseudo-random function with K_{coe} as a key, to calculate the linear combination of the data block as a verification token of the block j:

$$\nu_j = \sum_{k=1}^{r} \alpha_i^k G^j[I_k], \tag{4}$$

where ν_j indicates the verification token of the block j and $G^j[I_k]$ is the I_k sector of the j^{th} block of the extended client's file G. The DO must also blind the verification tokens to protect the data privacy before sending the tokens to the server.

(3) Error Localization: The objective of this component is to identify malicious servers. To achieve this goal, the client is only required to reveal the $\{\alpha_k\}_{k=1}^{r}$ and the pseudo-random permutation key K_{chal} to all of the servers and asks them to compute the linear combination of the certain block by using Equation (4). Upon receiving the tokens from all of the servers, the client removes the blind value and verifies it by using his or her stored secret matrix. As a result, the client is able to determine the file block corruptions.

(4) Dynamic Update: The SDS scheme permits the data owners to perform dynamic update operations on the outsourced data blocks, such as updates, deletes, appends, and inserts. The client only needs to prepare a matrix including changed blocks $\triangle f$, in which unchanged blocks are shown by the zero value. The client updates the verification tokens of the changed blocks on the basis of homomorphic tokens without retrieving other blocks. Therefore, the user blinds the data blocks and sends them to the servers along with the updated verification tokens.

4.3. Network-Coding-Based Remote Data Auditing

Reliable data storage can be achieved using distributed storage systems for an extended period of time. This is because peer-to-peer networks and modern data centers are usually deployed in unreliable environments, and such systems must support data redundancy to augment reliability against data failures. As previously mentioned in Section 3.1, replication is the simplest way to achieve such a goal, while the erasure coding provides better storage efficiency and orders of magnitude higher reliability than the replication technique for the same redundancy [Weatherspoon and Kubiatowicz 2002]. However, the main disadvantage of the erasure-coding-based solution is the communication overhead for the repair component.

Network coding (NC) is a critical technique that has the capability to overcome the communication overhead during the repair process. This is because when a data block is lost due to server failures, a new data block is created on the basis of a linear combination of the stored data blocks across the intact servers during the repair process [Dimakis et al. 2010, 2011]. For example, given an input file including m blocks ($F = b[1], b[2], \ldots, b[m]$), the network coding of the original file as a linear combination of the data blocks is generated on the basis of a coding coefficient vector of m random value ($v_1, v_2, \ldots, v_m$):

$$NC = \sum_{i=1}^{m} v_i b[i]. \tag{5}$$

These code blocks have the same size as an original file and are divided into m blocks. Similar to the erasure coding, the client stores the m code blocks across n servers redundantly, so that the DO has the capability to retrieve a file from any k servers. Figure 9 shows a network-coding-based distributed storage system for the original file including three blocks ($b[1], b[2], b[3]$). When data corruption is detected on Server 3, both Server 1 and Server 2, as intact servers, are selected to generate two linear combinations of the server's blocks. The coefficient values used for the linear combinations are represented by the arrows.

There are some challenges and difficulties that must be overcome when the network-coding-based RDA methods are used to validate the integrity and correctness of the outsourced files in distributed storage systems: (1) Error localization: After detecting the file corruption in distributed storage systems, the client must distinguish the faulty servers from intact servers to restore the faulty servers. (2) Loss of a fixed layout of the

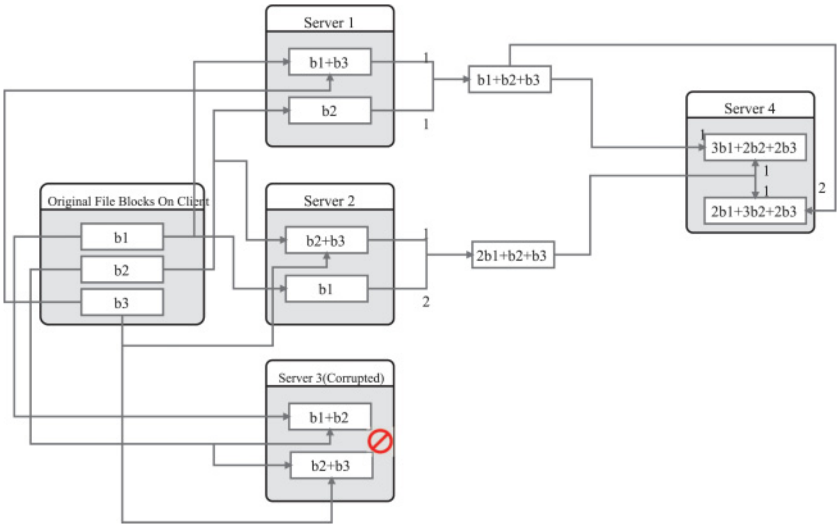

Fig. 9. Network-coding-based distributed storage system.

file: Unlike the erasure-coding-based storage, the network-coding-based storages can have various layouts because they create a new code block during the repair process. This variety of layout makes auditing the code block more challenging. (3) Reply attack: Here, the adversary reuses old coded blocks to decrease the redundancy on the storage servers until the client is unable to recover the original file. (4) Pollution attack: In this attack, the server is able to offer a correct message as a proof in the challenge step, while the corrupted data are used to generate a new block during the repair process. Consequently, the auditor must check the integrity of the new block without having access to the original blocks [Zhang et al. 2012; Oliveira et al. 2012].

Chen et al. [2010] proposed a holistic approach for remote data auditing in the network-coding-based storage systems (RDC-NC) by extending the single-server data auditing methods, such as PDP [Ateniese et al. 2007] and POR [Juels and Kaliski 2007], to validate the integrity of the stored blocks on each server. The first difficulty in implementing the RDC-NC scheme is to prevent the pollution attack and ensure that the data blocks are stored in the predetermined servers on the basis of the contract. The authors addressed the pollution attack by attaching the block index to the tag of the block. The next difficulty is to provide a fixed file layout when the client recovers new coded blocks from the intact servers in the repair process. The core idea behind the RDC-NC method to address the issue and preserve the storage in stable condition is to allocate logical identifiers to the coded data blocks outsourced on each of the servers. These identifiers are presented in the form of $S_{i.}b_j$, where S_i is the server number and $b_j$is the index of the block stored in the considering server. For instance, the identifier $i.j$ indicates that block j is stored on server i. Therefore, if server i fails, then the client must compute new code blocks to store them on a new server with the same identifier $(i.j)$. On the other hand, the logical identifiers also need to be attached to the verification tags of every single coded block. As a result, during the challenge step, the server is able to regenerate the logical identifiers $i.j$.

However, applying logical identifiers allows the malicious servers to pass the challenge step successfully by using old coded blocks with the identical logical identifier, known as the reply attack [Chen et al. 2010]. To overcome this issue, the client must encrypt the coefficients of the coding instead of storing them in plain text at the servers.

Therefore, even though the adversaries succeed in corrupting the servers, they are unable to obtain the original file to perform harmful replay attacks.

The NC-RDC scheme performs the verification by using two types of verification tags in two different steps: (1) Challenge verification tags: In the challenge step, all of the α-coded blocks in each of the n servers are checked to ensure that the servers actually possess the blocks based on a spot-checking-based challenge in POR [Juels and Kaliski 2007] and PDP [Ateniese et al. 2011]. Each server utilizes the blocks itself and the verification block to respond to the client's challenge. A challenge tag for a segment is actually responsible for binding the data segment in a coded block with a logical identifier of the block and with the considering coefficient vector. (2) Repair verification tags: If a defective server is detected in the repair step, the auditor retrieves a new coded block from each of the l healthy servers. Thereafter, the client creates α new coded blocks by combining the l received coded blocks and keeps the new blocks in a new healthy server. Such tags are employed in the repair step to ensure that the correct blocks and the coefficients are used to produce new coded blocks.

The main limitation of this method is that the verifier must download the remaining intact nodes to check the integrity of the new coded blocks, which incurs heavy bandwidth and computational overhead on the verifier's side.

Anh and Markopoulou [2012] built upon an RDA method for network-coding-based storage systems (NC-audit) on the basis of a homomorphic MAC scheme [Agrawal and Boneh 2009] and a customized encryption scheme that takes advantage of random linear combinations. The homomorphic MAC cryptosystem is used to protect the scheme against a pollution attack when the client and server have a shared secret key. This cryptosystem contains three probabilistic polynomial-time algorithms: sign, combine, and verify. The sign algorithm generates a tag for each of the source blocks, and then a linear combination of MAC tags of the blocks is generated by the combined algorithm using the homomorphic property. The server is able to validate the received tags and eliminate all of the invalid tags using the verify algorithm. Due to the ability of homomorphic MAC to aggregate blocks and tags, the auditor is capable of validating the integrity of several data blocks at the same time with the communication and computation overhead of a single block verification. To preserve the privacy of the blocks, the authors exploit the Random Linear Encryption (Ncrypt) scheme, in which the data blocks are masked with a randomly chosen vector so that the auditor has the capability to verify the data blocks.

When a data failure occurs, the new block is generated in the new server based on the linear combination of intact servers. To mitigate the communication and computation cost on the client side, the verification tag of the new data block is computed by combining the verification tags of intact blocks. This releases the client from the burden of generating a verification tag for the new data block. The data auditing in the NC-audit scheme, including the setup, challenge, proof, and verify steps, is illustrated in Figure 10.

Chen and Lee [2013] devised a Data Integrity Protection (DIP) method against data corruption for network coding distributed cloud storage on the basis of Functional Minimum-Storage Regenerating (FMSR) codes [Hu et al. 2012].

The FMSR code is an extended version of the Maximum Distance Separable (MDS) codes that enables a client to reconstruct a specific piece of a part of data with size less than ($|F|$. An $(n,\ k)$-FMSR code divides an original file into $k(n-k)$ fragments and randomly generates $n(n-k)$ block codes using a linear combination of the fragments where the size of the fragments and block codes are $|F|/K(n-k)$ [Rabin 1989]. The primary aim of the FMSR code is to reduce the number of reads from the intact servers to rebuild the lost data that is known as repair traffic. When a server failure is detected, the client needs to randomly select one block code from all of the $n-1$ intact servers,

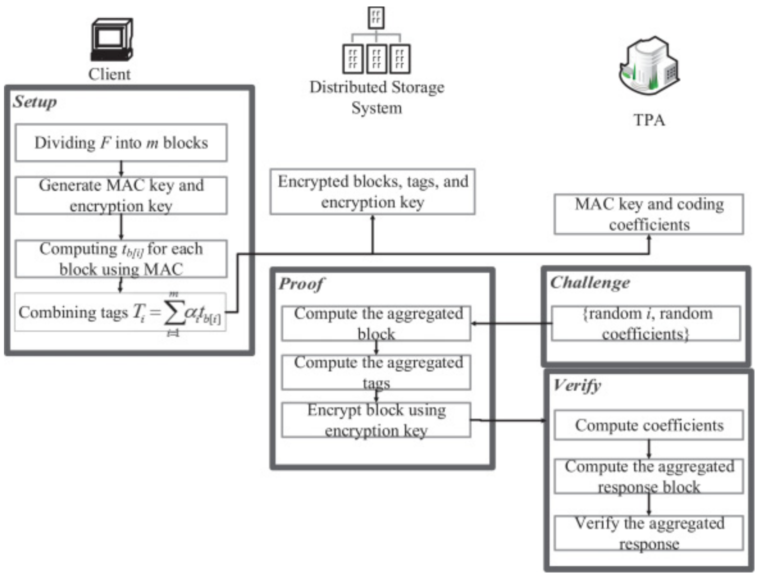
Client
Distributed Storage System
TPA
Setup
Dividing F into m blocks
Generate MAC key and encryption key
Computing $t_{b[i]}$ for each block using MAC
Combining tags $T_i = \sum_{i=1}^{m} \alpha_i t_{b[i]}$
Encrypted blocks, tags, and encryption key
MAC key and coding coefficients
Proof
Compute the aggregated block
Compute the aggregated tags
Encrypt block using encryption key
Challenge
{random i, random coefficients}
Verify
Compute coefficients
Compute the aggregated response block
Verify the aggregated response

Fig. 10. Network code remote data auditing scheme.

and then it generates the new block code based on the linear combinations of them. For example, the experimental results in Hu et al. [2012] indicate that the FMSR code reduces the rate of repair traffic to 50% for a large n, where $k = n - 2$.

The FMSR-DIP scheme consists of four operations: upload, check, download, and repair operations: (1) Upload operation: The client encrypts the input file using the FMSR code method to generate $n(n - k)$ data block codes. Before the block codes are uploaded to the servers, each of them must be encrypted with the FMSR-DIP codes. During this encryption process, first, each of the block codes must be protected against data corruption using the adversarial error-correcting code (AECC) [Bowers et al. 2009b]. After adding k bits parity to the block codes by using the AECC method, the block codes are blended using the pseudo-random functions. Thereafter, the MAC of the first block codes are computed to verify the integrity of the blocks. (2) Check operation: The client randomly selects some rows of block codes at the server to check the integrity by error detection in each row of the data blocks using the AECC method. (3) Download operation: When the clients want to download a file, $k(n - k)$ FMSR-DIP block codes from any k servers are selected to be verified on the basis of MAC. If the clients detect any failure, then the AECC parities must be downloaded to recover the errors. (4) Repair operation: When the number of failures is more than $n - k$, the client uses the FMSR technique to generate the new block on the new server. Figure 11 shows how the client is able to recover the blocks that are stored on the compromised server S_4 with the FMSR code.

The main limitation of the FMSR-DIP method is that it is only applicable to a thin-cloud interface [Vrable et al. 2009] that only supports read and write operations.

Table II summarizes and provides a comparison of the erasure-coding-based and the network-coding-based RDA methods.

5. TAXONOMY AND COMPARISON OF REMOTE DATA AUDITING METHODS

Figure 12 shows the thematic taxonomy of the RDA for the distributed storage systems that is classified on the basis of Scheme Nature (SN), security pattern, objective functions, auditing mode, update mode, cryptography model, and dynamic data structure.

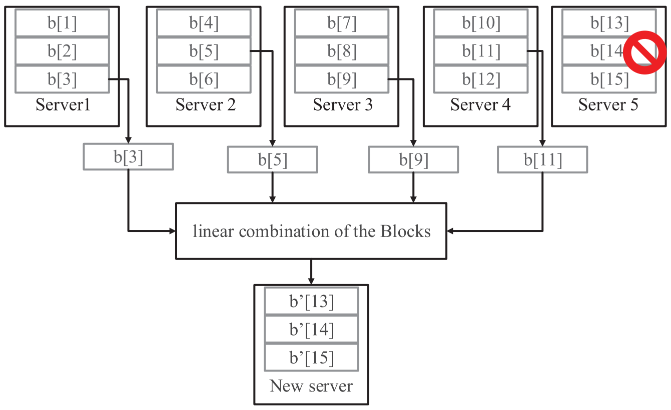

Fig. 11. Recovering data blocks in the (5,3) FMSR-DIP scheme.

Table II. Comparison of the Erasure-Coding-Based and Network-Coding-Based RDA Methods

Methods	Objectives	Drawbacks
HAIL [Bowers et al. 2009a]	Extends POR scheme to manage the integrity and availability across multiple clouds	a. High computation overhead; b. Cannot support data error localization
SDS [Wang et al. 2012]	a. Supports dynamic auditing; b. Resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks; c. Supports data error localizatio	Secure communication link between clients and cloud
RDC-NC [Chen et al. 2010]	a. Reduces significantly the communication cost of the repair phase by applying network-coding-based approach compared to erasure-coding-based methods; b. Supports data error localization; c. Resilient against pollution attacks	High bandwidth and computational overhead on the client during the repair process
NC-Audit [Anh and Markopoulou 2012]	Reduces the computation overhead by using the homomorphic MAC scheme	Secure communication link between clients and cloud
FMSR-DIP [Chen and Lee 2013]	Reduces the number of reads from the intact servers to rebuild the lost data by using FMSR code	a. Costly dynamic update; b. Only applicable to thin-cloud interface

The attribute of the SN indicates the various types of RDA techniques for distributed servers that are categorized based on the type of data redundancy into replication-based, erasure-coding-based, and network-coding-based methods. Replication is the most straightforward way of data redundancy in distributed storage systems, in which multiple copies of a file are outsourced in each server. A number of current RDA methods employ replication-based approaches [Curtmola et al. 2008b; Barsoum and Hasan 2010; Yan et al. 2012]. An erasure code approach encodes a file of k blocks into a larger file with n blocks such that the whole system can tolerate up to k block failures and any subset of the n blocks is sufficient to retrieve the original data blocks [Weatherspoon and Kubiatowicz 2002]. Network-coding-based data auditing indicates storing a linear combination of the original data blocks across various servers [Anh and Markopoulou 2012]. The security pattern attribute represents the cryptosystems that are used to construct the RDA methods.

The characteristic of the security pattern determines the type of cipher used at the cloud side or the client side to dynamically audit the stored data. The contemporary

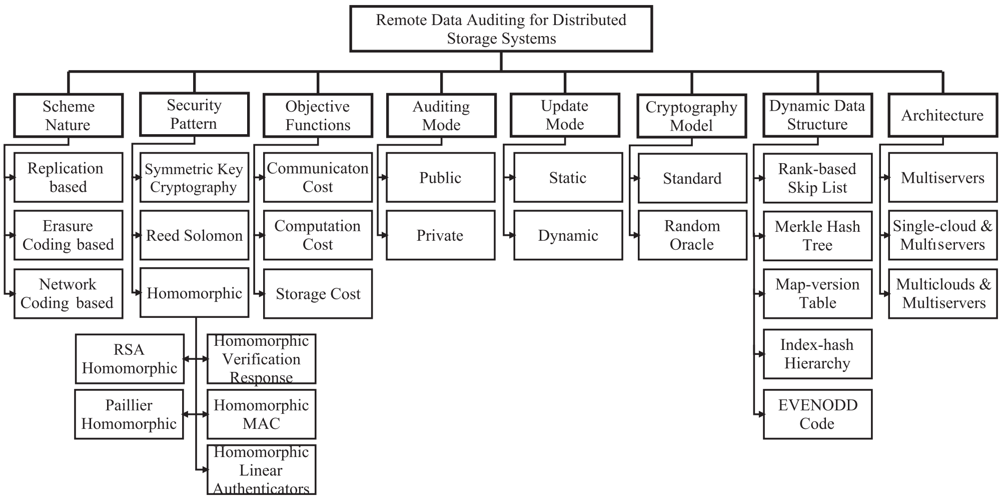

Fig. 12. Taxonomy of RDA techniques for the distributed cloud servers.

data storage security methods address the following security patterns for auditing the outsourced data: (1) Symmetric key cryptography: In the symmetric encryption scheme, the sender and receiver must establish a secure communication session based on a shared key and also the same key is used to encrypt and decrypt the message. (ii) The (n, k)-Reed-Solomon (RS) code is one of the most important encryption mechanisms to correct block-based error codes and is applicable in a wide range of digital communications and storage domains. The RS code encrypts k blocks to n blocks by adding $d = n - k$ bits parity to the original file to enable the DO to correct up to $\lfloor k/2 \rfloor$ symbols. (3) Homomorphic Encryption: Data encryption is an essential need to ensure data is securely accessed within the cloud. The subsequent issue, however, is how to efficiently perform calculation on the encrypted data to obtain identical results as when calculations were performed on the unencrypted data and without having to decrypt data. An encryption function ($E(.)$) is homomorphic if for any $E(x)$ and $E(y)$, $E(x \bigotimes y)$ can be computable without decrypting x and y for an operation $\bigotimes$:

$$\forall x, y \in M, E(x \otimes y) \leftarrow E(x) \otimes E(y). \tag{6}$$

In the distributed cloud server, the homomorphic encryption methods are divided into five different categories: (1) RSA Homomorphic: Rivest et al. [1978] proposed the first holomorphic encryption on the basis of the RSA scheme, which enables the client to combine tags computed from different data blocks of the file into a single value. For example, two messages (m_1, m_2) are encrypted based on multiplying the corresponding ciphertexts as follows:

$$\left.\begin{array}{l} E_k(m_1) = m_1^e modn \\ E_k(m_2) = m_2^e modn \end{array}\right\} \rightarrow E_k(m_1 m_2) = E_k(m_1)E_k(m_2), \tag{7}$$

where (e, n) indicates the clients' public key. (2) Paillier Homomorphic: The Paillier cryptosystem is a type of homomorphic cryptosystem [Paillier 1999] based on the RSA scheme in which the product of two ciphertexts (m_1, m_2) is calculated by

$$\left.\begin{array}{l} E_k(m_1) = g^{m_1} r^n modn^2 \\ E_k(m_2) = g^{m_2} r^n modn^2 \end{array}\right\} \rightarrow E_k(m_1 + m_2) = E_k(m_1)E_k(m_2), \tag{8}$$

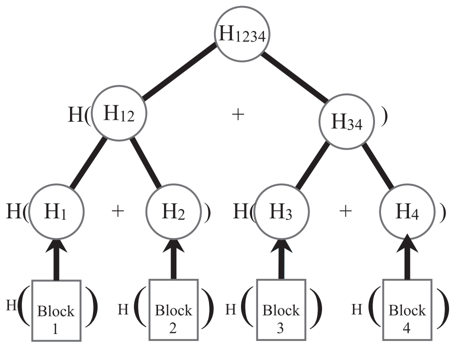
H1234
H(H12 + H34)
H(H1 + H2) H(H3 + H4)
H(Block 1) H(Block 2) H(Block 3) H(Block 4)

Fig. 13. Merkle Hash Tree structure.

where (g, n) is the clients' public key and r is a random number. (3) The Homomorphic Verification Token (HVT) is a mechanism that can be used in distributed clouds to verify the integrity and identify the errors by using the universal hash function [Carter and Wegman 1979] and Reed-Solomon method [Plank and Ding 2005]. (4) Homomorphic Linear Authentication (HLA) uses a linear sum of the discrete input blocks to produce a distinct value as output. Since the HLA benefits from the comparatively small-sized BLS signatures, it imposes less computation burden than HVT. (5) The Homomorphic Verification Response (HVR) is a mechanism that can be used to verify the integrity in distributed cloud storage by combining multiple responses from several clouds into a single value. (6) The Homomorphic MAC cryptosystem is used to protect the scheme against a pollution attack when the client and server have a shared secret key.

The attribute of the auditing mode shows who is responsible for verifying the outsourced data. In a private verification, the DO only has to verify the data integrity. However, in a public verification mode, the DO is able to delegate the auditing task to the TPA.

The attribute of the uploading mode indicates the type of data modification that can be supported by the protocols. The current RDA techniques employ two different strategies for updating the outsourced data blocks in the distributed cloud servers: (1) In the static approach, the user must download the data and upload the modified data on the cloud. This process imposes high communication and computation overheads on the cloud side and the device side. (2) In the dynamic uploading approach, the user is capable of updating the stored data to the cloud by inserting, deleting, and modifying a part of the file or appending to the file remotely rather than downloading the entire file.

The dynamic data structure attribute indicates what the data structure is used for in the scheme to support dynamic data update characteristics. This attribute includes the following: (1) The rank-based skip list is an authentication model that enables the client to efficiently perform the update, delete, and insert operations on the outsourced data. Each of the nodes in this data structure is subject to store the following: (a) the searching variables; (b) the data block's homomorphic tag $(T(b[i]))$; (c) the label of the node; (d) rank of a node, which indicates the number of reachable leaf nodes from that node; and (e) the level of the node. (2) The Hash Index Hierarchy (HIH) is a type of hash function that is used to combine multiple hash responses into a single value. (3) The Merkle Hash Tree (MHT) is a type of a binary tree structure often used for data integrity verification. In MHT, the leaves are the hash values of the individual blocks, while the remaining nodes are calculated on the basis of the hash value for a combination of the two children nodes. Figure 13 shows an MHT data structure for a file including four data blocks. (4) The map-version table is a structure to verify the integrity of outsourced data dynamically. The table includes three columns, Serial Number (SN), Block Number (BN), and Version Number (VN), to store the physical location, the logical location, and the number of updates for each block. (5) The EVENODD code is a type of MDS code used to achieve reliable communication and dynamic load balancing in distributed storage systems [Blaum et al. 1995].

The characteristic of the cryptographic model suggests a more common practice to outline the cryptographic protocols, together with the following: (1) standard model: in this model, the hash functions (SHA, MD5) are employed to ensure the overall security of the method, and (2) random oracle model: this model substitutes the hash function in the former model by a number of truly random functions. It should also be noted that once a method that uses the random oracle model is found secure, the implementation of such systems in the standard model is also secure [Canetti et al. 2004].

The attribute of the architecture indicates the cloud storage service architecture. As previously mentioned in Section 3.1, there are three architectures for the RDA approaches: multiservers, single cloud and multiservers, and multiclouds and multiservers. Table III represents the comparison summary of the RDA methods in the distributed cloud servers based on the presented attributes in the taxonomy.

The RDA methods incur additional communication, computation, and storage costs on the auditor and the prover because of performance of the challenge-response scheme. Moreover, the mobile device has various constraints, such as battery lifetime, limited CPU capacity, and insufficient storage. Therefore, it is necessary to compare the performance of RDA methods based on the objective function attribute. The objective function attribute involves crucial metrics that are used to analyze the performance and efficacy of the data auditing schemes, such as the following: (1) the storage overhead designates the amount of storage that clients or servers need to perform the auditing or repairing task. (2) Computational cost: performing a data auditing technique introduces a specific excessive amount of computation burden to the both the auditor and prover according to the type of cryptographic algorithm used. Generating the challenge message and verifying the correctness of the received proof message are the processes that require the client's computational resources. On the other hand, the main processes in servers are the update step and calculation of a proof for a file block. (3) The communication complexity is correlated with the size of the packet that is exchanged between the client and server during the auditing or repairing step including the challenge message and the proof message. Attribute-based critical comparison of the efficiency between a number of RDA protocols is displayed in Table IV, where n is the number of blocks that each file contains, s is the number of sectors in each block, m refers to the number of symbols within a block, $|f|$ stands for the size of file, t shows the number of blocks that are going to change, and c is the number of cloud providers in a multicloud architecture.

Moreover, the performance analysis of replication-based RDA methods, such as Curtmola et al. [2008b] and Barsoum and Hasan [2010, 2011], are tabulated on the basis of several parameters, such as (1) the computational cost to generate the replications, tags, and metadata before the uploading data block to the servers; (2) client and server storage overhead; (3) communication cost to send the challenge and response message by the client and servers; (4) computation cost to generate the proof by the server and to verify the proof by the client; and (5) the computation and computation cost to support the dynamic data update. Table V shows the storage, communication, and computation costs of the replication-based schemes, where h is cryptographic hashing, H_G is hashing to G, R is a random number, D is division in group G, D_z indicates division in group Z, E shows exponentiation in G, M_G is multiplication in G, M_Z is

Table III. Comparison of Remote Data Auditing Protocols for Distributed Cloud Servers Based on the Basic Parameters of Taxonomy

SN	Protocols	Security Pattern	Cryptography Model	Update Mode	Dynamic Structure	Auditing Mode	Architecture
Replication-based	MR-PDP [Curtmola et al. 2008a]	RSA Homomorphic	Random Oracle	Static	Support	Public	Multiservers
Replication-based	EMC-PDP [Barsoum and Hasan 2010]	HLA-Based BLS Signature	Random Oracle	Static	Support	Public	Single Cloud and Multiservers
Replication-based	DMR-PDP [Mukundan et al. 2012]	Paillier Homomorphic, BLS Signature	Random Oracle	Dynamic	Merkle Hash Tree	Private	Single Cloud and Multiservers
Replication-based	C-PDP [Yan et al. 2012]	Homomorphic Verification Response	Random Oracle	Static	Index-Hash Hierarchy	Public	Multiclouds and Multiservers
Replication-based	DR-DPDP [Etemad and Küpçü 2013]	RSA-Based HVT	Standard	Dynamic	Rank-Based Skip List	Private	Single Cloud and Multiservers
Replication-based	TB-DMCPDP [109]	HLA-Based BLS Signature	Random Oracle	Dynamic	Merkle Hash Tree	Public	Single Cloud and Multiservers
Replication-based	MB-DMCPDP [Schwarz and Miller 2006]	HLA-Based BLS Signature	Random Oracle	Dynamic	Map-Version Table	Public	Single Cloud and Multiservers
Erasure Coding-based	HAIL [108]	RS Codes, Universal Hash Function (UHF)	Standard	Static	Not Supported	Private	Single Cloud and Multiservers
Erasure Coding-based	SDS [Wang et al. 2012]	Homomorphic Verification Token	Random Oracle	Dynamic	Linear Property of Reed-Solomon Code, Verification Token Construction	Public	Single Cloud and Multiservers
Network Coding-based	RDC-NC [Chen et al. 2010]	Homomorphic Linear Authenticators (HLAs)	Random Oracle	Static	Not Support	Private	Single Cloud and Multiservers
Network Coding-based	NC-Audit [Anh and Markopoulou 2012]	Homomorphic MAC	Random Oracle	Dynamic	EVENODD Code	Public	Single Cloud and Multiservers
Network Coding-based	FMSR-DIP [Chen and Lee 2013]	Symmetric Encryption, Message Authentication Code (MAC)	Random Oracle	Static	Not Supported	Private	Single Cloud and Multiservers

Table IV. Efficiency Comparison Between Remote Data Auditing Protocols for Distributed Cloud Servers

	Client Storage Overhead		Server Storage	Client Computation	Server	Communication Complexity	
Protocols	Auditing	Repairing	(Repairing Mode)	(Auditing Mode)	Computation	Auditing	Repairing
C-PDP [Yan et al. 2012]	–	–	–	$O(t+s)$	$O(t+cs)$	$O(s)$	–
NC-Audit [Anh and Markopoulou 2012]	$O(cns)$	$O(cns)$	–	–	–	–	
RDC-NC [Chen et al. 2010]	$O(1)$	$O(cns)$	$O\left(\frac{2c\lvert f\rvert}{n+1}\right)$	–	$O(1)$	–	$O\left(\frac{2\lvert f\rvert}{n+1}\right)$
DR-DPDP [Etemad and Küpçü 2013]	$O(1)$	–		–	–	$O(1+logn)$	
MR-PDP [Curtmola et al. 2008b]	–	–	$O(c\lvert f\rvert)$	–	$O(1)$	–	$O(\lvert f\rvert)$
HAIL [Bowers et al. 2009a]	–	–	$O\left(\frac{c\lvert f\rvert}{n+1}\right)$		$O(1)$	–	$O(\lvert f\rvert)$

multiplication in Z, A_Z is addition in Z, P is bilinear pairing, and E_K shows encryption using K.

6. OPEN ISSUES FOR DISTRIBUTED-BASED REMOTE DATA AUDITING TECHNIQUES

The section represents a number of salient challenges in leveraging and applying the RDA techniques for distributed cloud servers and presents some open issues that serve as a platform for future research works.

6.1. Dynamic Data Update

Supporting dynamic data updates is an important characteristic of RDA methods both for single and distributed cloud servers, since many common applications such as online word processing intrinsically deal with a dynamic form of data or are involved with dynamic log files. During the update operations, such as modify, delete, insert, and opened in static mode, the clients must completely download the outsourced data from the cloud and upload it after performing the corresponding operations [Chen et al. 2013]. If the auditing method supports the dynamic data update, then the client only needs to download the number of blocks that are to be updated. As a result, such a feature reduces the computation and communication overhead of updating data on the client and servers.

Among various types of distributed-based remote data auditing approaches, such as replication based, erasure coding based, and network coding based, most of the methods belong to the replication-based category. However, the network-coding-based approaches can be more efficient than the other types because of their specific features. Although it is imperative to implement a dynamic data auditing for NC-based distributed servers, this has received less attention by the researchers due to its complex nature.

6.2. Batch Auditing

The batch auditing feature enables TPA to process multiple auditing tasks received from different users at the same time rather than performing each of the tasks separately. In other words, the batch auditing property utilizes the linear sum of the random

Table V. Storage, Communication, and Computation Costs for Replication-Based Remote Data Auditing Protocols for Distributed Cloud Servers

Cost \ Protocols		MR-PDP [Curtmola et al. 2008b]	EMC-PDP [Barsoum and Hasan 2010]	TB-DMCPDP [Barsoum and Hasan 2011]	MB-DMCPDP [Barsoum and Hasan 2011]
Precomputation Cost	Generating Replications	$E_k + nmR + nmA_z$	–	–	–
	Generating Tags	$2nm\varepsilon_{Z_n^*} + nmM_G + nmH_G$	$2nm\varepsilon_G + nmM_G + nmH_G$	$(s+1)nm\varepsilon_G + mnH_G + (sn+n-1)mM_G$	$(s+1)nm\varepsilon_G + mnH_G + (sn+n-1)mM_G$
	Generating Metadata	–	–	$mnH_G + (2m+1)n\,h$	–
Storage Cost	File Copies	$n\|f\|$	$n\|f\|$	$n\|f\|$	$n\|f\|$
	Server Overhead	$1024m\ bits$	–	$(257+512n)m\ bits$	$257m\ bits$
	Client Overhead	–	–	$256\ bits$	$64m\ bits$
Comm. Cost	Challenge	$1280 + log_2(c)\ bits$	$256 + log_2(c)\ bits$	$256 + log_2(c)\ bits$	$256 + log_2(c)\ bits$
	Response	$1024(n+1)\ bits$	$160(n+1)\ bits$	$257 + 256sn + (256\,log_2(m) + 257)cn\ bits$	$257 + 256sn\ bits$
Computation Cost	Proof	$(c+n)\varepsilon_{Z_n^*} + (cn+c-1)M_Z + (c-1)nA_z$	$c\varepsilon_G + c(n+1)M_Z + cnA_z$	$c\,\varepsilon_G + (c-1)M_G + csnM_Z + (c-1)snA_z + cnH_G$	$c\,\varepsilon_G + (c-1)M_G + csnM_Z + (c-1)snA_z$
	Verify	$(2n+c+1)\varepsilon_{Z_n^*} + (cn+c+n)M_Z + cnA_z + cH_G + 1D_Z$	$2P + (c+2)\varepsilon_G + (c+1)M_G + nA_z + cH_G$	$\left(clog_2(m)+2\right)\ nh + 2P + (c+s)\varepsilon_G + cnH_G + (cn+s-1)M_G + s(n-1)A_z$	$2P + (c+s+1)\varepsilon_G + cH_G + (c+s-1)M_G + s(n-1)A_z$
Dynamic Operations Cost	Comm.	–	–	$“Request” + O(n\,log_2(m))$	$“Request”$
	State update	–	–	$O(n\,log_2(m))\ h$	–

blocks to shorten the proof message and thus mitigate the corresponding communication overhead [Wang et al. 2011]. Due to the redundancy characteristic of the RDA algorithms, addressing batch auditing in the distributed storage systems is more challenging. Only a few existing RDA methods [Yan et al. 2012] focus on the batch auditing issue in the distributed storage systems. A simple way to achieve such a goal is to use a bilinear aggregate signature [Boneh et al. 2003] to combine the proof messages into a single and unique signature that can be verified by the auditor.

6.3. Privacy Preserving

When DOs outsource data to the remote cloud or delegate the auditing task to the trusted third party, it is important for them that the auditors or cloud not be given the opportunity to gain knowledge of the data content or be able to make a copy of the original data [Wang et al. 2013]. That is to say that, most of the data auditing methods for the distributed cloud servers usually assume that the TPA is a trustworthy agent, though such an illogical assumption further leads to data leakage. Randomization of data blocks and tags is a common method to address the privacy issue to prevent tag or data leakage during the entire verification phase.

6.4. Data Deduplication

Data deduplication basically removes duplicate data copies in order to facilitate a cost-effective storage. It is a kind of data compression technique (as a single-instance data storage) that is employed to avoid data redundancy [Mandagere et al. 2008; Meyer and Bolosky 2012]. There is no inconsistency between duplication and the distributed storage system because the technique has to identify a common byte set inside or among files to allow single-instance storage of each fragment in each of the servers on the basis of the replication-based, erasure-coding-based, or network-coding-based approaches.

6.5. Large-Scale Data Auditing

We live in the area of large-scale data where billions of files and petabytes of data are stored in the distributed data storage. Around 2.5 quintillion bytes of data are created every day by Sakr et al. [2013]. By increasing the size of data, the RDA protocols incur communication, storage, and computation costs on both the auditor side and the prover side.

On the other hand, there are many big data applications that employ the cloud to store a huge amount of data with small size, such as Twitter. Although the users of Twitter are only able to write less than 144 characters, they can generate up to 12 terabytes of data per day and update the data very frequently [Liu et al. 2013; Naone 2010]. The problem of big data auditing worsens in the case of dynamic data updates, in which the DO modifies a single bit of the outsourced data. This is because of the type of data structure that is used to support the dynamic data update. For example, after performing an update operation, the MHT must be kept balanced. Therefore, large-scale data auditing with the aim of minimizing computation, communication, and storage overhead is primarily an open issue.

7. CONCLUSIONS

In this article, we have discussed, characterized, and categorized a wide range of research areas relevant to the RDA techniques for distributed cloud servers. We began by explaining the concept of cloud computing and distributed storage systems and discussing the RDA technique to protect the outsourced data in geographically distributed, heterogeneous, and untrusted cloud servers. Moreover, we focused on the architecture of the distributed-based remote data auditing techniques and the fundamental differences between distributed and single auditing approaches. Our research presented an extensive survey on the security of data storage in distributed servers. The article also reported a parametric thematic taxonomy with the aim of classifying the common methods and highlighting the differences and similarities for the comparison of the RDA for the distributed storage systems.

The state-of-the-art RDA techniques were compared for multiple cloud servers to classify them according to the presented taxonomy. We also highlighted the issues and the challenges relating to the security requirements in order to offer an efficient and lightweight security mechanism. Moreover, we critically examined the methods to discover some of the advantages and disadvantages, the significance, and the requirements and identified the research gap in the architecture. The article also depicts some of the important trends for researchers around the globe in this domain. Furthermore, numerous open challenges, particularly, dynamic data update, privacy preserving, batch auditing, and data deduplication, were introduced as prominent upcoming research challenges for further investigation.

REFERENCES

Giuseppe Aceto, Alessio Botta, Walter de Donato, and Antonio Pescap. 2013. Cloud monitoring: A survey. *Computer Networks* 57, 9 (2013), 2093–2115.

Atul Adya, William J. Bolosky, Miguel Castro, Gerald Cermak, Ronnie Chaiken, John R. Douceur, Jon Howell, Jacob R. Lorch, Marvin Theimer, and Roger P. Wattenhofer. 2002. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. *SIGOPS Operating Systems Review* 36, SI (2002), 1–14.

Gagan Agrawal and Pankaj Jalote. 1995. Coding-based replication schemes for distributed systems. *IEEE Transactions on Parallel and Distributed Systems* 6, 3 (1995), 240–251.

Shweta Agrawal and Dan Boneh. 2009. Homomorphic MACs: MAC-based integrity for network coding. In *7th International Conference on Applied Cryptography and Network Security (Lecture Notes in Computer Science)*, Vol. 5536. Springer, 292–305.

Keno Albrecht, Ruedi Arnold, Roger Wattenhofer, Roger Wattenhofer, and Roger Wattenhofer. 2003. *Clippee: A Large-Scale Client/Peer System*. ETH, Eidgenössische Technische Hochschule Zürich, Department of Computer Science.

Thomas E. Anderson, Michael D. Dahlin, Jeanna M. Neefe, David A. Patterson, Drew S. Roselli, and Randolph Y. Wang. 1996. Serverless network file systems. *ACM Transactions on Computer Systems* 14, 1 (1996), 41–79.

Le Anh and A. Markopoulou. 2012. NC-Audit: Auditing for network coding storage. In *International Symposium on Network Coding (NetCod '12)*. 155–160.

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, and Ion Stoica. 2010. A view of cloud computing. *Communications of the ACM* 53, 4 (2010), 50–58.

Michael Arrington. 2006. Gmail Disaster: Reports of Mass Email Deletions. Retrieved from http://techcrunch.com/2006/12/28/gmail-disaster-reports-of-mass-email-deletions/.

Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Osama Khan, Lea Kissner, Zachary Peterson, and Dawn Song. 2011. Remote data checking using provable data possession. *ACM Transactions on Information Systems Security* 14, 1 (2011), 1–34.

Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. 2007. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*. ACM, Alexandria, Virginia, USA, 598–609.

Giuseppe Ateniese, Roberto Di Pietro, Luigi V. Mancini, and Gene Tsudik. 2008. Scalable and efficient provable data possession. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Network*. ACM, Istanbul, Turkey, 1–10.

Roger Barga, Dennis Gannon, and Daniel Reed. 2011. The client and the cloud: Democratizing research computing. *IEEE Internet Computing* 15, 1 (2011), 72–75.

Ayad F. Barsoum and M. Anwar Hasan. 2010. *Provable Possession and Replication of Data over Cloud Servers*. Centre for Applied Cryptographic Research (CACR) Report 32, University of Waterloo, (2010), 1–36.

Ayad F. Barsoum and M. Anwar Hasan. 2011. On verifying dynamic multiple data copies over cloud servers. *IACR Cryptology ePrint Archive* 2011 (2011), 447–476.

Mario Blaum, Jim Brady, Jehoshua Bruck, and Jai Menon. 1995. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computing* 44, 2 (1995), 192–202.

Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*. Lecture Notes in Computer Science, Vol. 2656. Springer, Book section 26, 416–432.

Kevin D. Bowers, Ari Juels, and Alina Oprea. 2009a. HAIL: a high-availability and integrity layer for cloud storage. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*. ACM, Chicago, Illinois, USA, 187–198.

Kevin D. Bowers, Ari Juels, and Alina Oprea. 2009b. Proofs of retrievability: theory and implementation. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*. ACM, Chicago, Illinois, USA, 43–54.

Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 25, 6 (2009), 599–616.

Rodrigo N. Calheiros, Christian Vecchiola, Dileban Karunamoorthy, and Rajkumar Buyya. 2012. The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds. *Future Generation Computer Systems* 28, 6 (2012), 861–870.

Ran Canetti, Oded Goldreich, and Shai Halevi. 2004. The random oracle methodology, revisited. *Journal of the ACM* 51, 4 (2004), 557–594.

J. Lawrence Carter and Mark N. Wegman. 1979. Universal classes of hash functions. *Joural of Computer and System Sciences* 18, 2 (1979), 143–154.

David Cash, Alptekin Küpçü, and Daniel Wichs. 2012. Dynamic proofs of retrievability via oblivious RAM. *IACR Cryptology ePrint Archive* (2012), 550–550. http://eprint.iacr.org/.

The Sidekick Cloud Disaster. Retrieved from http://www.bbc.co.uk/blogs/technology/2009/10/the_sidekick_cloud_disaster.html.

Suh Changho and K. Ramchandran. 2010. Exact-repair MDS codes for distributed storage using interference alignment. In *IEEE International Symposium on Information Theory Proceedings*. 161–165.

Bo Chen, Reza Curtmola, Giuseppe Ateniese, and Randal Burns. 2010. Remote data checking for network coding-based distributed storage systems. In *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*. ACM, Chicago, Illinois, USA, 2010, 31–42.

Henry C. H. Chen and Patrick P. C. Lee. 2013. Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation. *IEEE Transactions on Parallel and Distributed Systems* 99 (2013), 1–1.

Lanxiang Chen. 2013. Using algebraic signatures to check data possession in cloud storage. *Future Generation Computer Systems* 29, 7 (2013), 1709–1715.

Lanxiang Chen, Shuming Zhou, Xinyi Huang, and Li Xu. 2013. Data dynamics for remote data possession checking in cloud storage. *Computers & Electrical Engineering* 39, 7 (2013), 2413–2424.

Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. 2001. *Freenet: A Distributed Anonymous Information Storage and Retrieval System*. Lecture Notes in Computer Science, Vol. 2009. Book section 4, 46–66.

Reza Curtmola, Osama Khan, and Randal Burns. 2008a. Robust remote data checking. In *Proceedings of the 4th ACM International Workshop on Storage Security and Survivability*. ACM, Alexandria, Virginia, USA, 63–68.

Reza Curtmola, Osama Khan, Randal Burns, and Giuseppe Ateniese. 2008b. MR-PDP: Multiple-replica provable data possession. In *Proceedings of the 28th International Conference on Distributed Computing Systems*. Purdue Univ., West Lafayette, 411–420.

Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. 2001. Wide-area cooperative storage with CFS. *SIGOPS Operating Systems Review* 35, 5 (2001), 202–215.

Alexandros G. Dimakis, P. Brighten Godfrey, Yunnan Wu, Martin J. Wainwright, and Kannan Ramchandran. 2010. Network coding for distributed storage systems. *IEEE Transactions on Information Theory* 56, 9 (2010), 4539–4551.

Alexandros G. Dimakis, Kannan Ramchandran, Yunnan Wu, and Suh Changho. 2011. A survey on network codes for distributed storage. *Proceedings of the IEEE* 99, 3 (2011), 476–489.

Choon Hoong Ding, Sarana Nutanong, and Rajkumar Buyya. 2005. Peer-to-peer networks for content sharing. In *Peer-to-Peer Computing: The Evolution of a Disruptive Technology*. Idea Group Publishing, Hershey, PA, USA, 28–65.

Roger Dingledine, Michael J. Freedman, and David Molnar. 2001. *The Free Haven Project: Distributed Anonymous Storage Service*. Lecture Notes in Computer Science, Vol. 2009. Springer, Book section 5, 67–95.

Peter Druschel and Antony Rowstron. 2001. PAST: A large-scale, persistent peer-to-peer storage utility. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems*. IEEE, 75–80.

Chris Erway, Alptekin Küpçü, Charalampos Papamanthou, and Roberto Tamassia. 2009. Dynamic provable data possession. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*. ACM, 1653688, 213–222.

Mohammad Etemad and Alptekin Küpçü. 2013. Transparent, distributed, and replicated dynamic provable data possession. *IACR Cryptology ePrint Archive* 2013 (2013), 225.

Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. 2013. Mobile cloud computing: A survey. *Future Generation Computer Systems* 29, 1 (2013), 84–106.

Armando Fox, Rean Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica. 2009. *Above the Clouds: A Berkeley View of Cloud Computing.* Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Technical Report UCB/EECS 28 (2009).

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. *SIGOPS Operating Systems Review* 37, 5 (2003), 29–43.

Nancy Gohring. 2008. Amazon's S3 down for several hours. (2008). http://status.aws.amazon.com/s3-20080720.html.

Vânia Gonçalves and Pieter Ballon. 2011. Adding value to the network: Mobile operators experiments with Software-as-a-Service and Platform-as-a-Service models. *Telematics and Informatics* 28, 1 (2011), 12–21.

Christian Hanser and Daniel Slamanig. 2013. Efficient simultaneous privately and publicly verifiable robust provable data possession from elliptic curves. *IACR Cryptology ePrint Archive* (2013), 392–406.

John H. Hartman and John K. Ousterhout. 1993. The Zebra striped network file system. In *Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles*. Asheville, North Carolina, USA, 29–43.

Ragib Hasan, Zahid Anwar, William Yurcik, Larry Brumbaugh, and Roy Campbell. 2005. A survey of peer-to-peer storage techniques for distributed file systems. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, Vol. 2, 205–213.

M. Azua Himmel and F. Grossman. 2014. Security on distributed systems: Cloud security versus traditional IT. *IBM Journal of Research and Development* 58, 1 (2014), 31–313.

Paul Horn. 2001. Autonomic computing: IBMś perspective on the state of information technology. Retrieved from http://www.research.ibm.com/autonomic/manifesto.

Yuchong Hu, Henry C. H. Chen, Patrick P. C. Lee, and Yang Tang. 2012. NCCloud: applying network coding for the storage repair in a cloud-of-clouds. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*. USENIX Association, San Jose, CA, 21–29.

James V. Huber Jr, Andrew A. Chien, Christopher L. Elford, David S. Blumenthal, and Daniel A. Reed. 1995. PPFS: A high performance portable parallel file system. In *Proceedings of the 9th International Conference on Supercomputing*. ACM, 385–394.

C. N. Hüfer and G. Karagiannis. 2011. Cloud computing services: Taxonomy and comparison. *Journal of Internet Services and Applications* 2, 2 (2011), 81–94.

Si-Yuan Jing, Shahzad Ali, Kun She, and Yi Zhong. 2013. State-of-the-art research study for green cloud computing. *Journal of Supercomputing* 65, 1 (2013), 445–468.

Ari Juels Jr. and Burton S. Kaliski. 2007. PORs: proofs of retrievability for large files. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*. ACM, Alexandria, Virginia, USA, 584–597.

Dongyoung Koo, Junbeom Hur, and Hyunsoo Yoon. 2013. Secure and efficient data retrieval over encrypted data using attribute-based encryption in cloud storage. *Computers & Electrical Engineering* 39, 1 (2013), 34–46.

Mads Dar Kristensen. 2009. Enabling cyber foraging for mobile devices. In *Proceedings of the 5th MiNEMA Workshop: Middleware for Network Eccentric and Mobile Applications*. Citeseer, 32–36.

John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. 2000. OceanStore: An architecture for global-scale persistent storage. *SIGPLAN Notes* 35, 11 (2000), 190–201.

Alptekin Küpçü. 2010. *Efficient Cryptography for the Next Generation Secure Cloud*. Thesis. Retrieved from home.ku.edu.tr/∼akupcu/papers/kupcu-phd.pdf.

Chang Liu, Jinjun Chen, Laurence T. Yang, Xuyun Zhang, Chi Yang, Rajiv Ranjan, and Ramamohanarao Kotagiri. 2013. Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates. *IEEE Transactions on Parallel and Distributed Systems* 99 (2013), 1–1.

Satyanarayanan Mahadev. 1990. Scalable, secure, and highly available distributed file access. *Computer* 23, 5 (1990), 9–18. DOI:http://dx.doi.org/10.1109/2.53351

Nagapramod Mandagere, Pin Zhou, Mark A Smith, and Sandeep Uttamchandani. 2008. Demystifying data deduplication. In *Proceedings of the ACM/IFIP/USENIX Middleware'08 Conference Companion*. Leuven, Belgium, 12–17.

Petros Maniatis, Mema Roussopoulos, T. J. Giuli, David S. H. Rosenthal, and Mary Baker. 2005. The LOCKSS peer-to-peer digital preservation system. *ACM Transactions on Computer Systems* 23, 1 (2005), 2–50. DOI:http://dx.doi.org/10.1145/1047915.1047917

Peter Mell and Timothy Grance. 2011. The NIST definition of cloud computing (draft). *NIST Special Publication* 800 (2011), 145.

R. C. Merkle. 1980. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy*. Oakland, CA, USA, 122–134.

Dutch T. Meyer and William J. Bolosky. 2012. A study of practical deduplication. *Transactions on Storage* 7, 4 (2012), 1–20. DOI:http://dx.doi.org/10.1145/2078861.2078864

Rich Miller. 2010. Amazon Addresses EC2 Power Outages. Retrieved from http://www.datacenterknowledge.com/archives/2010/05/10/amazon-addresses-ec2-power-outages/.

Li Mingqiang and Shu Jiwu. 2010. DACO: A high-performance disk architecture designed specially for large-scale erasure-coded storage systems. *IEEE Transations on Computers* 59, 10 (2010), 1350–1362.

Raghul Mukundan, Sanjay Madria, Mark Linderman, and N. Y. Rome. 2012. Replicated data integrity verification in cloud. *Bulletin of the Technical Committee on Data Engineering* (2012), 55–65.

Athicha Muthitacharoen, Robert Morris, Thomer M. Gil, and Benjie Chen. 2002. Ivy: A read/write peer-to-peer file system. *SIGOPS Operating Systems Review* 36, SI (2002), 31–44.

Erica Naone. 2010. What Twitter Learns from All Those Tweets. Retrieved from http://www.technologyreview.com/view/420968/what-twitter-learns-from-all-those-tweets/.

Paulo F. Oliveira, Luísa Lima, Tiago T. V. Vinhoza, João Barros, and Muriel Médard. 2012. Coding for trusted storage in untrusted networks. *IEEE Transactions on Information Forensics and Security* 7, 6 (2012), 1890–1899.

Andrew Oram. 2001. *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technologies*. O'Reilly Media, CA.

Ivan Osipkov, Peng Wang, Nicholas Hopper, and Yongdae Kim. 2006. Robust accounting in decentralized P2P storage systems. In *IEEE International Conference on Distributed Computing Systems*. 14–14.

Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques.* Lecture Notes in Computer Science, Vol. 1592. Springer, 223–238.

James S. Plank. 2005. T1: erasure codes for storage applications. In *Proceedings of the 4th USENIX Conference on File and Storage Technologies*. San Francisco, 1–74.

James S. Plank and Ying Ding. 2005. Note: Correction to the 1997 tutorial on Reed-Solomon coding. *Software: Practice and Experience* 35, 2 (2005), 189–194. DOI:http://dx.doi.org/10.1002/spe.631

Michael O. Rabin. 1989. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM* 36, 2 (1989), 335–348.

Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. 1978. On data banks and privacy homomorphisms. *Foundations of Secure Computation* 32, 4 (1978), 169–178.

Chunming Rong, Son T. Nguyen, and Martin Gilje Jaatun. 2013. Beyond lightning: A survey on security challenges in cloud computing. *Computers & Electrical Engineering* 39, 1 (2013), 47–54.

Robert B. Ross and Rajeev Thakur. 2000. PVFS: A parallel file system for Linux clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*. USENIX Association, 391–430.

Ahmad-Reza Sadeghi, Thomas Schneider, and Marcel Winandy. 2010. *Token-Based Cloud Computing*. Lecture Notes in Computer Science, Vol. 6101. Springer, Book section 30, 417–429.

Sherif Sakr, Anna Liu, and Ayman G. Fayoumi. 2013. The family of Mapreduce and large-scale data processing systems. *ACM Computer Surveys* 46, 1 (2013), 1–44.

Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. 1985. Design and implementation of the Sun network filesystem. In *Proceedings of the Summer USENIX Conference*. 119–130.

Frank B. Schmuck and Roger L. Haskin. 2002. GPFS: A shared-disk file system for large computing clusters. In *Proceedings of the 1st Conference on File and Storage Technologies (FAST'02)*, Vol. 2. 19.

Mathew J. Schwartz. 2012. 6 Worst Data Breaches of 2011. Retrieved from http://www.informationweek.com/news/security/attacks/232301079.

S. J. Thomas Schwarz and Ethan L. Miller. 2006. Store, forget, and check: Using algebraic signatures to check remotely administered storage. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*. 12–12.

Hovav Shacham and Brent Waters. 2008. *Compact Proofs of Retrievability*. Lecture Notes in Computer Science, Vol. 5350. Springer, Book section 7, 90–107.

Elaine Shi, Emil Stefanov, and Charalampos Papamanthou. 2013. Practical dynamic proofs of retrievability. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM, 325–336.

Y. Shin, J. Hur, and K. Kim. 2012. Security weakness in the proof of storage with deduplication. *IACR Cryptology ePrint Archive* (2012), 554. http://eprint.iacr.org.

Sandeep K. Sood. 2012. A combined approach to ensure data security in cloud computing. *Journal of Network and Computer Applications* 35, 6 (2012), 1831–1838.

Mehdi Sookhak, Adnan Akhunzada, Abdullah Gani, Muhammad Khurram Khan, and Nor Badrul Anuar. 2014a. Towards dynamic remote data auditing in computational clouds. *Scientific World Journal* 2014 (2014), 12. DOI:http://dx.doi.org/10.1155/2014/269357

Mehdi Sookhak, Hamid Talebian, Ejaz Ahmed, Abdullah Gani, and Muhammad Khurram Khan. 2014b. A review on remote data auditing in single cloud server: Taxonomy and open issues. *Journal of Network and Computer Applications* 43 (2014), 121–141. DOI:http://dx.doi.org/10.1016/j.jnca.2014.04.011

Emil Stefanov, Marten van Dijk, Ari Juels, and Alina Oprea. 2012. Iris: A scalable cloud file system with efficient integrity checks. In *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 229–238.

Darlene Storm. 2011. Epsilon Breach: Hack of the Century? Retrieved from http://blogs.computerworld.com/18079/epsilon_breach_hack_of_the_century.

S. Subashini and V. Kavitha. 2011. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications* 34, 1 (2011), 1–11. DOI:http://dx.doi.org/10.1016/j.jnca.2010.07.006

Chandramohan A. Thekkath, Timothy Mann, and Edward K. Lee. 1997. Frangipani: A scalable distributed file system. *SIGOPS Operating Systems Review* 31, 5 (1997), 224–237.

Kurt Tutschku. 2004. *A Measurement-Based Traffic Profile of the eDonkey Filesharing Service*. Lecture Notes in Computer Science, Vol. 3015. Springer, Book section 2, 12–21. DOI:http://dx.doi.org/10.1007/978-3-540-24668-8_2

Luism Vaquero, Luis Rodero-Merino, and Daniel Morán. 2011. Locking the sky: A survey on IaaS cloud security. *Computing* 91, 1 (2011), 93–118.

Sudharshan S. Vazhkudai, Xiaosong Ma, Vincent W. Freeh, Jonathan W. Strickland, Nandan Tammineedi, and Stephen L. Scott. 2005. FreeLoader: Scavenging desktop storage resources for scientific data. In *Proceedings of the ACM/IEEE Supercomputing Conference*. Washington, DC, USA, 56–56.

Srikumar Venugopal, Rajkumar Buyya, and Kotagiri Ramamohanarao. 2006. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computer Surveys* 38, 1 (2006), 3.

Michael Vrable, Stefan Savage, and Geoffrey M. Voelker. 2009. Cumulus: Filesystem backup to the cloud. *Transactions on Storage* 5, 4 (2009), 1–28.

Marc Waldman, Aviel D. Rubin, and Lorrie Faith Cranor. 2001. Publius: A robust, tamper-evident censorship-resistant web publishing system. In *Proceedings of the 9th USENIX Security Symposium*. 59–72.

Cong Wang, S. S. M. Chow, Qian Wang, Kui Ren, and Wenjing Lou. 2013. Privacy-preserving public auditing for secure cloud storage. *IEEE Transactions on Computers* 62, 2 (2013), 362–375.

Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. 2010. Toward publicly auditable secure cloud data storage services. *IEEE Network* 24, 4 (2010), 19–24.

Cong Wang, Qian Wang, Kui Ren, Ning Cao, and Wenjing Lou. 2012. Toward secure and dependable storage services in cloud computing. *IEEE Transactions on Services Computing,* 5, 2 (2012), 220–232.

Huaqun Wang. 2012. Proxy provable data possession in public clouds. *IEEE Transactions on Services Computing,* 99 (2012), 1–1.

Qian Wang, Cong Wang, Jin Li, Kui Ren, and Wenjing Lou. 2009. Enabling public verifiability and data dynamics for storage security in cloud computing. *Proceedings of Computer Security* 5789 (2009), 355–370.

Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. 2011. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE Transactions on Parallel and Distributed Systems* 22, 5 (2011), 847–859.

Weichao Wang, Zhiwei Li, Rodney Owens, and Bharat Bhargava. 2009. Secure and efficient access to outsourced data. In *Proceedings of the ACM Workshop on Cloud Computing Security*. ACM, 1655016, 55–66.

Hakim Weatherspoon and John D. Kubiatowicz. 2002. *Erasure Coding Vs. Replication: A Quantitative Comparison*. Lecture Notes in Computer Science, Vol. 2429. Springer, Book section 31, 328–337.

Lifei Wei, Haojin Zhu, Zhenfu Cao, Xiaolei Dong, Weiwei Jia, Yunlu Chen, and Athanasios V. Vasilakos. 2014. Security and privacy for storage and computation in cloud computing. *Information Sciences* 258 (2014), 371–386.

Chun Wesley. 2011. What Is Google App Engine? Retrieved from https://ep2012.europython.eu/conference/talks/google-app-engine-best-practices-latest-features.

Md Whaiduzzaman, Mehdi Sookhak, Abdullah Gani, and Rajkumar Buyya. 2014. A survey on vehicular cloud computing. *Journal of Network and Computer Applications* 40 (2014), 325–344.

Zack Whittaker. 2012. Amazon web services suffers partial outage. Retrieved from http://www.zdnet.com/blog/btl/amazon-web-services-suffers-partial-outage/79981.

Da Xiao, Yan Yang, Wenbin Yao, Chunhua Wu, Jianyi Liu, and Yixian Yang. 2012. Multiple-file remote data checking for cloud storage. *Computers & Security* 31, 2 (2012), 192–205. DOI:http://dx.doi.org/10.1016/j.cose.2011.12.005

Min Xie, Haixun Wang, Jian Yin, and Xiaofeng Meng. 2007. Integrity auditing of outsourced data. In *Proceedings of the 33rd International Conference on Very Large Data Bases*. VLDB Endowment, 1325940, 782–793.

Zhu Yan, Hu Hongxin, Ahn Gail-Joon, and Yu Mengyang. 2012. Cooperative provable data possession for integrity verification in multicloud storage. *IEEE Transactions on Parallel and Distributed Systems* 23, 12 (2012), 2231–2244.

Kan Yang and Xiaohua Jia. 2012. An efficient and secure dynamic auditing protocol for data storage in cloud computing. *IEEE Transactions on Parallel and Distributed Systems* PP, 99 (2012), 1717–1726.

Liu Ying and V. Vlassov. 2013. Replication in distributed storage systems: State of the art, possible directions, and open issues. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. IEEE, 225–232.

Shucheng Yu, Wnjing Lou, and Kui Ren. 2012. *Data Security in Cloud Computing*. Morgan Kaufmann/Elsevier, Book section 15, 389–410.

Jiawei Yuan and Shucheng Yu. 2013a. Proofs of retrievability with public verifiability and constant communication cost in cloud. In *Proceedings of the International Workshop on Security in Cloud Computing*. ACM, Hangzhou, China, 19–26.

Jiawei Yuan and Shucheng Yu. 2013b. Secure and constant cost public cloud storage auditing with deduplication. *IACR Cryptology ePrint Archive* 2013 (2013), 149.

Qi Zhang, Lu Cheng, and Raouf Boutaba. 2010. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications* 1, 1 (2010), 7–18.

Xiaolan Zhang, Giovanni Neglia, and Jim Kurose. 2012. *Network coding in disruption tolerant networks.* Academic Press, Boston, 267–308.

Qingji Zheng and Shouhuai Xu. 2011. Fair and dynamic proofs of retrievability. In *Proceedings of the First ACM Conference on Data and Application Security and Privacy*. 237–248.

Qingji Zheng and Shouhuai Xu. 2012. Secure and efficient proof of storage with deduplication. In *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*. ACM, San Antonio, Texas, USA, 1–12. DOI:http://dx.doi.org/10.1145/2133601.2133603

Zhou Zhibin and Huang Dijiang. 2012. Efficient and secure data storage operations for mobile cloud computing. In *8th International Conference and Workshop on Systems Virtualiztion Management Network and Service Management*. 37–45.

Xiao Zhifeng and Xiao Yang. 2013. Security and privacy in cloud computing. *IEEE Communications Surveys & Tutorials* 15, 2 (2013), 843–859.

Yan Zhu, Huaixi Wang, Zexing Hu, Gail-Joon Ahn, Hongxin Hu, and Stephen S. Yau. 2010. Efficient provable data possession for hybrid clouds. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*. ACM, New York, NY, 756–758.

Dimitrios Zissis and Dimitrios Lekkas. 2012. Addressing cloud computing security issues. *Future Generation Computer Systems* 28, 3 (2012), 583–592.

Received August 2014; revised January 2015; accepted February 2015