A review on remote data auditing in single cloud server: Taxonomy and open issues

Mehdi Sookhak, Hamid Talebian, Ejaz Ahmed, Abdullah Gani, Muhammad Khurram Khan

Abstract

Cloud computing has emerged as a computational paradigm and an alternative to the conventional computing with the aim of providing reliable, resilient infrastructure, and with high quality of services for cloud users in both academic and business environments. However, the outsourced data in the cloud and the computation results are not always trustworthy because of the lack of physical possession and control over the data for data owners as a result of using to virtualization, replication and migration techniques. Since that the security protection the threats to outsourced data have become a very challenging and potentially formidable task in cloud computing, many researchers have focused on ameliorating this problem and enabling public auditability for cloud data storage security using remote data auditing (RDA) techniques. This paper presents a comprehensive survey on the remote data storage auditing in single cloud server domain and presents taxonomy of RDA approaches. The objective of this paper is to highlight issues and challenges to current RDA protocols in the cloud and the mobile cloud computing. We discuss the thematic taxonomy of RDA based on significant parameters such as security requirements, security metrics, security level, auditing mode, and update mode. The state-of-the-art RDA approaches that have not received much coverage in the literature are also critically analyzed and classified into three groups of provable data possession, proof of retrievability, and proof of ownership to present a taxonomy. It also investigates similarities and differences in such framework and discusses open research issues as the future directions in RDA research.

Keywords:
Cloud computing
Remote data auditing
Provable data possession
Proof of retrievability
Proof of ownership

Contents

1. Introduction .......................................................... 122
2. Background ......................................................... 123
   2.1. Cloud computing ............................................... 123
   2.2. Mobile cloud computing ...................................... 123
3. Remote data auditing technique .................................. 124
   3.1. Remote data auditing architecture ......................... 125
   3.2. Taxonomy of remote data auditing ....................... 125
4. The state-of-the-art remote data auditing approaches: taxonomy ............................................. 126
   4.1. Provable data possession based methods ................ 126
      4.1.1. Static PDP models ...................................... 126
      4.1.2. Dynamic PDP models ................................. 127
      4.1.3. Privacy-preserving models ............................ 129
      4.1.4. Robust data auditing .................................. 131
   4.2. Proofs of retrievability-based methods ................ 132
      4.2.1. Static POR models ...................................... 132
      4.2.2. Dynamic POR models .................................. 133

* Corresponding author.
E-mail addresses: m.sookhak@ieee.org (M. Sookhak), talebian@siswa.um.edu.my (H. Talebian), ejazahmed@ieee.org (E. Ahmed), abdullah@um.edu.my (A. Gani), mkhurram@ksu.edu.sa (M.K. Khan).

http://dx.doi.org/10.1016/j.jnca.2014.04.011
1084-8045/© 2014 Elsevier Ltd. All rights reserved.
1. Introduction

Cloud computing is a new model of computing in contrast to conventional desktop computing. Today’s, this new paradigm became popular and received increasing attention by researchers (academia) and industry. According to The National Institute of Standards and Technology (NIST) Cloud Computing is “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (network, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort” (Mell and Grance, 2011).

This technology allows users to outsource their data to a remote server operated by a third party called cloud service provider (CSP) (Zhibin and Dijiang, 2012). In addition, computing resources such as memory, disk storage, processor, and bandwidth are virtualized and clients are able to access them using the Internet (Kumar and Yung-Hsiang, 2010). The term cloud refers to a thousand of virtualized servers distributed over a set of data centers with different geographical locations connected together through telecommunication links. The services on the cloud are delivered to the users as pay-as-you-go pricing model. This means users are only charged for the amount of service they have used similar to water and electricity bills.

Adopting cloud computing offers various advantages to both end users and CSP. For end users the advantages include rapid elasticity, measured service, minimal capital investment, lower maintenance cost, and location-independent access to the services (Kumar and Yung-Hsiang, 2010; Wang et al., 2010). On the other hand, CSP achieves a higher level of resource utilization and thus saves energy consumption.

Despite several benefits, some security concerns inhibit users to fully adopt this new technology and shift from traditional computing to cloud computing (Zhibin and Dijiang, 2012). By storing data to a remote server, user loses his physical control over data and instead delegates management of data to an un-trusted party (Cong et al., 2010; Wei et al., 2013). Even though cloud resources are very powerful and reliable comparing to that of local machine, the data on the cloud is still vulnerable to many threats from inside or outside the cloud (Wang et al., 2010). These threats might compromise confidentiality, integrity, and availability of data. An unfaithful provider might delete less frequently accessed data to free up disk space or hide data loss to protect his reputation (Yang and Jia, 2012a). In addition, security attacks, Byzantine failure, server failure, and power outage are likely to happen. Amazon S3 breakdown (Team, 2008), Gmail email mass deletion (Arrington, 2006), Sidekick Cloud Disaster (Cellan-Jones, 2009) Breakdown of Amazon EC2 2010 (Miller, 2010) are example of such events.

Cloud users need to make sure their data remain intact after uploading to the remote server. Traditional integrity checking techniques such as hash functions and signatures require a local copy of the entire data. Unfortunately, these techniques are not well suited for the cloud environment because downloading possibly large files is impractical due to its high communication cost. This even becomes worse in case of mobile computing devices with limited power, storage capacity, and connectivity. As a result, devising a proper audit service which can remotely check the integrity of outsourced data in the cloud is deemed as a crucial need.

Remote data auditing (RDA) refers to a group of protocols to securely, frequently, and efficiently verify the correctness of the data over a cloud managed by untrustworthy provider without having to retrieve the data (Ateniese et al., 2008). The RDA protocols are able to check a small fraction of entire data, called spot checking, and give a probabilistic guarantee for the data integrity. To design a remote data audit mechanism the following important criteria must be taken into account: (1) Efficiency: audit the data with the minimum computational cost over the server and particular client. The auditing service is also reasonable for the communication overhead between client and server, (2) Public verifiability: delegate the audit task to a trusted third party auditor rather than a client in order to reduce the computation cost over the client, (3) Frequency: number of times that user is able to verify the integrity of outsourced data by generating a challenge message, (4) Probability of detection: probability by which a protocol detects data corruption, (5) Recovery: ability to recover data in case of data corruption, and (6) Dynamic update: enabling the cloud user to update the outsourced data by using insert, delete, modify, and append operation without requiring to download the whole data.

This paper reviews the state-of-the-art remote data auditing efforts that are used to check the integrity of outsourced data in a single cloud server. We study and classify the characteristics of remote data auditing approaches by thematic taxonomy into five groups, namely security requirements, security objective, performance metrics, auditing mode, and update mode. The impacts of RDA in cloud and mobile cloud computing are also presented. The main contribution of the paper is to review the state-of-the-art RDA methods, categorize current RDA mechanisms into three classes of provable data possession, proof of retrievability, and proof of ownership based on the implications, requirements, and critical characteristics. To the best of our knowledge, this is the first effort to categorize data storage strategies applied in single cloud computing. Furthermore, we identify the issues in existing solutions for data auditing and challenges to cloud based application processing and mobile device limitations. This paper lists some challenges and open issues to guide researchers to choose the appropriate domain for future research and acquire ideas for further investigations.

The rest of the paper is organized as follows. Section 2 presents the fundamental concepts of cloud computing and mobile cloud computing. It also explains data auditing and its requirements.
Section 3 discusses the concept of RDA, presents our proposed taxonomy of RDA. Section 4 presents and taxonomizes a comprehensive survey of the state-of-the-art RDA approaches. Section 5 compares current RDAs by comparing the similarities and deviations by using significant parameters presented in the taxonomy. Section 6 focuses on the issues and challenges in current RDCs. Finally, Section 7 concludes the paper with future directives.

2. Background

This section explains the concept of mobile cloud computing and data auditing, respectively. Then, it illustrates a general architecture of remote data checking protocols and the requirements of the data auditing system.

2.1. Cloud computing

Cloud computing has become an emerging technology paradigm and has been envisaged as the next generation of Information Technology (IT) because of the many benefits it brings to users such as its low-cost, on-demand services, ubiquitous resource access, rapid elasticity or expansion, and measured services (Mell and Grance, 2011). Cloud users are able to conveniently scale up/down their virtual allocated resources according to their current need with minimal management effort and service interruption (Aceto et al., 2013; Rong et al., 2013; Xu, 2012). The elasticity characteristic reduces resource waste in case of over-provisioning in which more resources are allocated than what is actually required (Armbrust et al., 2010).

Cloud services are based on pay-as-you-go pricing model in which users only pay for what they have actually used with respect to some service metrics. For instance, the service can be measured as Gigabyte per month or server per hour (Zhifeng and Yang, 2013). Location independent is an important feature of cloud that allows users to freely access services on the cloud anywhere and anytime using any computing device with internet connection. Services on the cloud can be easily migrated from one physical server to another server in different geographical locations. This flexibility enhances reliability and helps to balance load among multiple servers (Raj et al., 2012; Zhou et al., 2012).

Figure 1 illustrates three models in cloud services: Software as Service (SaaS), Platform as Service (PaaS), and Infrastructure as Service (IaaS). In the SaaS layer, the user can access any kind of software service remotely from the mobile devices. For example, Microsoft Live Mesh allows users to share files and folder over multiple devices at the same time. Platform as a Service (PaaS) allows the user to set the runtime environment or modify the environment based on the requirement for a specific application. PaaS also provides the necessary framework programming environment, libraries and tools- to the end-users to allow them to create, manage, and deploy applications. For example, Google App Engine, Microsoft Azure, and Amazon Map are PaaS services currently available in the market. Infrastructure as a Service (IaaS) provides a more flexible environment for the end-users. It provides storage, computation, and networking infrastructure at the Virtual Machine (VM) level. For example, Amazon EC2 (Elastic Cloud Computing) and S3 (Simple Storage Service) are two IaaS services (Jing et al., 2013; Subashini and Kavitha, 2011; Takabi et al., 2010; Xu, 2012).

The cloud computing deployment model classifies cloud into four models such as public, private, hybrid and community cloud. (1) Public Cloud: the cloud services are open for public use, (2) Private Cloud: the cloud is exclusively used (or managed) by a single organization, (3) Community Cloud: a cloud for a set of organizations with shared interest or objectives, and (4) Hybrid Cloud: a combination of two or three of aforementioned clouds (Rong et al., 2013; Zhang et al., 2010; Zissis and Lekkas, 2012).

2.2. Mobile cloud computing

Mobile systems or devices like smartphone, tablets, and palm-tops have become a widely-used computational platform recently resulting from the extensive development in mobile devices computational capability and resource availability. The demands of mobile resources are increasing dramatically as more and more users are using mobile devices to perform their daily activities. However, the energy capacity is still a prime issue of mobile devices. These limitations prevent developers to design such applications for mobile devices. It is to be noted, these limitations are integral part of these devices and they are not expected to be resolved in near time according to current trends (Simoens et al., 2011; Whaiduzzaman et al., 2013).

Mobile cloud computing (MCC) has emerged as a computational paradigm for the next generation smart mobile devices, and is becoming increasingly popular because it provides a low cost, highly available rich mobile computing environment. The central idea of MCC is integrating the power of cloud with flexibility of
MCC consists of three elements: (1) portable mobile devices such as smartphone, tablets, and palmtops, (2) wireless networks, and (3) cloud computing center. Virtualization is the core technology of MCC that is used to separate the mobile application from specific access methods and device platforms, and offers faster service, greater mobile abilities and longer battery life. MCC also offers different types of services such as Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Network as a Service (NaaS) (Buyya et al., 2009). Among them, NaaS has recently been defined as the service layer for mobile cloud computing as IaaS to provide a flexible and secure virtual network (Aepona, 2010; Huang, 2011).

### 3. Remote data auditing technique

Today’s cloud users are motivated to store their data in the cloud and take advantage of the on-demand applications without the need to install them on their devices. Figure 2 shows a general comparison between the traditional systems and cloud and mobile cloud computing. It indicates that in contrast to the traditional systems, the storage service in the cloud (in SaaS, PaaS and IaaS layers) is managed by cloud service providers (CSP) and the users are unable to manage the data stored in the cloud (Ludwig, 2011). Therefore, several issues need to be resolved before storing the sensitive data in the cloud. For instance, how can the user completely put her trust in the CSP for preserving the outsourced data? Is it possible for the CSP or any inside attackers to arbitrarily change the amount of stored data without user knowledge or permission? Do the users have to download the whole outsourced data to check the integrity of them? Is there any way to update the outsourced data without having to download the entire data?

Devising a mechanism to assure cloud users that the data remain intact in the cloud storage is an essential need to trust the cloud computing paradigm. Remote data auditing (RDA) refers to a sampling of the collected data in the cloud and evaluating the data against various criteria, such as validity, accuracy, and integrity as a way to provide an assurance to the cloud users.
way to verify the reliability of the storage provider (Wang et al., 2010).

3.1. Remote data auditing architecture

RDA approaches in cloud and mobile cloud computing domain usually include four main components such as (1) Data Owner (DO): who outsources her data in the cloud through a mobile device, laptop or PC. DO is also able to update the data in the cloud by using insert, append, delete operations, (2) Cloud Data Provider (CSP): is an entity with a large amount of space to store owner’s data, (3) Third Party Auditor (TPA): since performing auditing task introduce performance overheads to data owner side the audit task is preferably delegated to an entity called TPA. TPA is a trusted entity with expertise and capability to do the auditing task on behalf of the data owner. By employing TPA, users (especially users with limited computing power, storage and connectivity) are alleviated from the burden of expensive auditing task. Although TPA is considered as trustworthy and reliable entity for data verification but it is curious at the same time. Therefore, one of the important issues in data audit service in presence of the TPA is to prevent data leakage during auditing and preserve the privacy of data, and (4) User (A/B/C): is an entity (individual or enterprise) who is registered by data owner and granted access to the outsourced data for reading the data (Koo et al., 2013; Sood, 2012; Yu et al., 2012). The architecture of RDA when TPA is involved is shown in Figure 3.

Remote data checking services follow response-challenge procedure in which (1) the DO firstly pre-processes her file and generates some metadata and hand over metadata to TPA. At this point, DO is not required to be involved in audit process anymore, (2) to check correctness of data on the cloud, TPA generates a random challenge message and sends it to the CSP (the DO is also able to generate the challenge message when the TPA is not supported by RDA service), (3) when Cloud storage receives the challenge, computes the corresponding response and send it to the TPA, (4) after receiving a response from CSP, the verification is carried out to find out whether CSP has correctly stored the file or not. It is important to mention that in order to reduce the overhead of audit process only a small fraction of whole file is queried (Xiao et al., 2012).

3.2. Taxonomy of remote data auditing

Figure 4 shows the thematic taxonomy of remote data auditing in cloud computing that is categorized based on Security Requirements, Security Metrics, Security Level, Auditing Mode and Updating Mode. The security requirements attribute indicates a number of properties which must be taken into account to propose a secure RDA method, as follows: (1) robustness equips the auditing methods with mechanisms to mitigate arbitrary amount of data corruption (Ateniese et al., 2011), (2) fairness ensures that a dishonest data owner is unable to access the data in the cloud storage and manipulate it (Zheng and Xu, 2012), (3) data deduplication ensures maximum use of available storage space by recognizing distinct chunks of data with identical content and eliminating redundant data. Considering that more than 75% of the outsourced data in the cloud are not unique, deduplication can dramatically reduce the required space to store a large data set (Gantz and Reinsel, 2010; Storer et al., 2008), (4) data recovery allows users to recover small or large fraction of file corruptions outsourced to the cloud. This requirement can be achieved by using some methods such as forward error correcting code (FEC) (Clark and Cain, 1981) or Reed–Salmon code (Lin and Costello, 2004), (5) dependability protects the stored data against Byzantine failures (Castro and Liskov, 2002), malicious data modification, and server colluding attack to augment data availability, (6) batch auditing ensures that TPA is able to quickly manage multiple auditing tasks which are received simultaneously from different users and in a cost efficient way, and (7) data privacy ensures that the auditors should not be able to learn or guess the data content or have a copy of original data. In other words, data confidentiality should be preserved against the auditors (Wei et al., 2013).

The performance metrics attribute includes a set of important measures such as computation cost, communication cost, and storage cost, and probability of detection which are needed to be kept optimized when designing a RDA method. The implemented method requires incurring the least computation, communication
and storage overhead over the client and server while the probability of detecting data corruption achieves the maximum value (Bowers et al., 2009; Oprea et al., 2005).

The security objective attributes indicate the RDA method is able to ensure which type of security components (integrity, confidentiality, and privacy). The next attribute is auditing mode including public and private auditing. In public auditing mode the integrity of outsourced data is checked by TPA while in the private mode, the data owner is only able to audit the data. Finally, current RDA methods use three different strategies for updating the stored data: (1) static model: the owner has to download the whole file and upload it after modification, which imposes high communication and computation overhead on the devices, (2) dynamic update: the owner is able to insert, append, delete or modify the file without downloading it entirely (Kan and Xiaohua, 2013), and (3) semi-dynamic model: allows the owner to make partial update operations on the outsourced data.

4. The state-of-the-art remote data auditing approaches: taxonomy

RDA is a crucial technique that concerns data integrity verification and public or private auditing services in the cloud and mobile cloud computing. According to identified security requirements in the previous Section, we analyze and taxonomize the state-of-the-art remote data auditing approaches into three models, namely provable data possession-based (PDP-based), proof of retrievability-based (POR-based), and proof of ownership-based (POW-based) which are depicted in Figure 5. We describe few remote data auditing methods for each group and tabulate the comparison results in Tables 1 and 2.

4.1. Provable data possession based methods

The first group of remote data auditing schemes in cloud and mobile cloud computing that is only responsible to preserve the integrity of outsourced data, is called provable data possession (PDP). This type of methods usually includes four main steps – setup, challenge, proof, and verification. (1) Setup phase: in this phase, the input data is divided into \( n \) blocks and the unique tag (metadata) for each block is computed using the distinctive formula. Finally, the input file and tags are sent to the cloud, (2) challenge phase: in order to audit the cloud and check the correctness of the stored data, a verifier requires selecting some data blocks randomly as a challenge by using pseudo-random permutation, (3) proof phase: upon receiving the challenge message, the prover generates a short integrity check over the received challenge message as a proof message – that usually includes the aggregation of the blocks and the tags – and sends it to verifier, and (4) verification phase: in the verification phase, the verifier validates the proof message regarding to the proof and challenge messages. The structure of PDP-based methods is shown in Figure 6.

In the rest of this section some PDP-based methods are critically reviewed along with their advantages and disadvantages.

4.1.1. Static PDP models

Ateniese et al. (2007) were the first to propose two provably-secure schemes by using the RSA-based Homomorphic Verifiable Tag (HVT) to verify the integrity of data storage in the cloud without having to retrieve the data. HVT permits the client to check whether the server has certain blocks based on a proof that is constructed by the server, even when the client has no access to the blocks. Secure PDP (S-PDP) is the first model with strong guarantee on data possession by adding the robust feature to PDP based on spot-checking mechanism. Since that incur computation cost on the client, they suggested an Efficient PDP (E-PDP) to reduce the computation cost by assuring the possession of the combined blocks (Ateniese et al., 2011). However, these two schemes have several drawbacks such as: (1) incur expensive server computation and communication cost over the whole file because of using RSA numbering, (2) have linear storage for the user, and (3) fail to provide secure data possession completely when the prover has malicious intent.

Hanser and Slamanig (2013) offered a provable data possession method based on elliptic curves cryptosystem in which a data owner or third party is simultaneously able to audit outsourced data remotely. The main idea behind this method is generating the
same tag for simultaneous private and public verifiability by identifying a vector for each data block. Therefore, the input file including \( n \) data blocks, is represented by \( nt \) consecutive vectors where \( t \) is the length of each vector.

\[
F = \begin{bmatrix}
F_1 \\
F_2 \\
\vdots \\
F_t
\end{bmatrix} = 
\begin{bmatrix}
F_{1,1} & \cdots & F_{1,t} \\
\vdots & \ddots & \vdots \\
F_{t,1} & \cdots & F_{t,t}
\end{bmatrix}
\]

Finally, the data owner calculates a tag \( (T_i) \) for each vector \( (F_i) \) by using hash function which maps to an elliptic curve group (Icart, 2009).

Schwarz and Miller (2006) were the first to propose a challenge–response scheme to check remotely administered storage based on the algebraic signature properties in the peer-to-peer networks. Algebraic signature is a type of hash functions with algebraic properties that takes a signature of the sum of some random blocks gives the same result as taking the sum of the signatures of the corresponding blocks. Chen (2013) derived algebraic signatures to check data possession in cloud storage efficiently because the algebraic signature imposes less computation overhead on client side and cloud side rather than Homomorphic cryptosystem. In this method, the algebraic properties are used to calculate the algebraic signature of the sum of some file blocks (block tag) without the need for the challenger. To compute an algebraic signature of the sum as the block tag, the algebraic signature of each data block is generated and then \( c \) file blocks are randomly selected. Data owner generates \( t \) different tags to audit the cloud by \( t \) random challenge. Therefore, the data owner needs to refresh the block tags after \( t \) time verifications. The main drawback of this method is that it relies on probabilistic guarantee of possession because of checking a random subset of file blocks rather than checking all blocks in deterministic security.

The data owner or TPA is always in charge of auditing the outsourced data in all PDP methods. In some situation, however, the client will be restricted due to unavailability of internet connection, such as on the ocean-going vessel, in the jail or battlefield. On the other hand, the TPA is not able to perform remote data checking independently, when the data owner is not capable of passing the verification step. The TPA does not have permission to take the further countermeasures without informing the owner. Wang (2012) overcame this issue by proposing a Proxy PDP (PPDP) method by using the bilinear pairing technique in which a remote data integrity checking task is delegated to a proxy according to a warrant. As it is shown in Figure 7, the PPDP scheme consists of four steps: (1) the system parameters and the public keys are generated by TTP, (2) the warrant and corresponding sign are generated by the data owner to delegate the tasks to proxy, (3) the proxy verifies the received warrant and the corresponding Signature, (4) the client divides the input into \( n \) blocks, generates the corresponding tag for each block, and stores them on the cloud, (5) the CSP validates the tags to resist the malicious clients, and (6) the proxy is able to audit the stored data in the cloud by using the challenge–response method.

4.1.2. Dynamic PDP models

As mentioned in Section 2, dynamic data update is a useful feature for data auditing methods that allows data owners to update their outsourced data whenever necessary without the need to download the data. The dynamic data update includes update, insert, append and delete operations.

Ateniese et al. (2008) considered the problem of static PDP methods for updating data and developed a new PDP protocol called Scalable PDP based on symmetric-key cryptography to solve the scalability, efficiency, and dynamic update issues in the original PDP method. The distinction between Scalable PDP and original PDP is that a certain number of short possession verification tokens are pre-computed by the data owner before uploading data on the cloud. Each token is generated by using a random challenge and a random set of data blocks. Although the scalable PDP supports dynamic data, the update operations are limited to modify, delete, and append. To update a data block in the cloud, the data owner needs to retrieve all tokens and replace the hash of the block’s old version with the new one by using XOR operation (Bellare et al., 1995). However, once an update operation is asked by user, it is required to re-compute all remaining tokens which are computationally expensive and thus impractical for large files. In addition, even though the scalable PDP enjoys more efficiency than original PDP, but the number of updates and challenges is restricted and it does not support public verifiability in which...
other parties rather than data owner also can check the integrity of outsourced data.

One of the effective ways to add dynamic data support to the current RDA protocols is making use of authenticated data structures such as Merkle Hash Tree (MHT) (Merkle, 1980), skip list (Pugh, 1990). The first fully dynamic PDP method is designed by Erway et al. (2009) by combining the original skip list (Pugh, 1990) with an authentication dictionary (Goodrich et al., 2001) and rank-based information to enable efficient authentication of clients’ updates. In rank-based authentication skip list structure, each node stores the homomorphic block tag of the data block \( (T(b[i])) \), level of node, the number of leaf nodes which are reachable from that node as a rank of node, searching variables, and a label of node. Erway et al. (2009) also proposed another dynamic PDP method by using rank-based RSA trees to enhance the probability of detecting a tampered block in the dynamic PDP method. The main difference of these two methods is storing the rank information trees on the internal nodes in the rank-based RSA scheme. To update a data block in Dynamic PDP, the client requests to retrieve the homomorphic block tag of this data block \( (T(b[i])) \) and its proof. In delete operation, the client needs to access the homomorphic block tag of previous block \( (T(b[i−1])) \) and its proof. In insert operation, the height of the tower of the skip list associated with the new block is also re-computed by the client. After verifying the proof, the client requires calculating the label of the start node of the skip list after the update by Papamanthou and Tamassia (2007). In the last step of update operations, the server updates the skip list on the basis of the received parameters.

DPDP employs rank-based Authenticated Skip List to sufficiently support dynamic data update with \( O(\log n) \) complexity (Erway et al., 2009). However, the variable size of updated file incurs more overhead on other blocks with \( O(n) \) complexity because the indices of the blocks are used in the skip list. For example, Figuer 8 shows the outsourced file that is divided into some blocks. If the data owner decides to change the “white” to “red”, it needs to balance the list by deleting some blocks and insert them in the new place because the size of blocks in Rank-based Authenticated Skip List is fixed.

Esiner et al. (2013) overcome this issue by implementing a Flexible Length-based Authenticated Skip List method in which the indices of the bytes of the file are used to facilitate inserting, updating, deleting, or challenging a specific block consisting of the bytes at specific indices of the file. The significant advantage of the FlexList method is its compatibility with the variable size of data block and data update. In other words, each node indicates how many bytes can be reached from this node instead of the number of accessible blocks. As a result, FlexList scheme is faster dynamic PDD in terms of data update with \( O(u) \) complexity where \( u \) is the size of update. Figure 9 shows that the data owner only needs to update the 3rd leaf of the list and the considering fathers.

Since prior dynamic data possession methods require verifying the cloud for each data block update operation, they incur high computation overhead on cloud and data owner. On the other hand, today, many web-based service providers include the web services, blogs, and other web-based application providers tend to outsource their data to the cloud or remote servers. In this way, users of such services are able to get access to the data anytime and from anywhere, and perform functions such as deleting, modifying, or inserting new data to the stored data, simultaneously. For instance, most of the popular blogs which are hosted by a cloud-based server permit their subscribers to delete, append, or remove blog content, freely. In this context, the data auditing methods should be able to manage multi-user access to the shared data on the cloud without leaking or losing data. Sometimes, the clients also need to retrieve previous versions of their data or they need to verify the integrity of the data without concerning about computation and storage overhead.

Zhang and Blanton (2012) designed an efficient dynamic provable possession based on a new data structure – that is called block update tree – to address these issues. The data owner and server need to store the block update tree in order to overcome the requirement of verification for each dynamic operation. The main characteristic of this tree is that it is always balanced irrespective of the number and order of dynamic operations on the storage. Moreover, the size of the maintained update tree is independent of the outsourced data size because when the owner restores some data blocks in the cloud, all previous copies of the considering data blocks are deleted. The block update tree is a binary tree in which each node consists of some attributes such as (1) node type (op) indicates the type of operation that is performed on the node (delete = −1, modify = 0, and insert = 1). (2) data block range \((LU)\) represent the range of data blocks in which the index of left child is always lower indices than \( L \) and the index of right child is always higher than \( U \). As a result, some standard algorithms such as AVL tree (Adelson-Velskii and Landis, 1963) are able to be used to efficiently balance the tree. (3) offset \((R)\) is used to identify the
indices of data blocks after insert and delete operations, and (4) version number \( (V) \) represents the number of data block modification. For example, Figure 10 shows the node balancing function when a new node \( (D) \) is added to the tree. Since that the range of this node is \([111,120]\), it is inserted as a left child of \( A \). \( B \) also needs to be increased because of the range overlap between \( B \) and \( D \).

Merkle Hash Tree (MHT) is a simple and effective model of the authentication structure which is presented as a binary hash tree. This data structure is used to detect any tampering and to prove that a set of elements remains unaltered. The leaves of the MHT are the hashes of authentic data values and the other nodes are computed by hashing the combination of the hash of left and right children \( (h(\text{left child}) \| h(\text{right child}))\). However, MHT has a node balancing drawback which occurs after inserting or deleting a series of requests. As a result, this technique is not directly applicable in provable data possession schemes. Wang et al. (2011) proposed a public and dynamic data auditing method by combining the MHT data structure (Merkle, 1980) and bilinear aggregate signature (Boneh et al., 2003) to address the node balancing issue in MHT. When data owner decides to update a data block, she sends the new data block along with its authentication tag to the cloud. Upon receiving the update request, the cloud performs the update operation, re-generates the root of the MHT, updates aggregation block tags and returns the signed root \( (\text{sign}_{pr}(\text{root of MHT})) \). Finally, the owner validates the signed root to ensure the performance of update operation. Figure 11 illustrates the effect of insert and delete operations on MHT in the Public PDP method.

4.1.3. Privacy-preserving models

Cloud-based collaborative authoring is a fledgling service that helps the clients to share private documents with others anywhere and anytime. The cloud-based structure of collaborative authoring service augments usability and fault tolerance; and achieves efficient resource allocation and global accessibility (Sheng-Cheng et al., 2012). However, by storing data to a remote server, the clients lose the physical control over data and delegate management of data to an untrusted cloud service provider. As a result, to protect the privacy of data, the clients need to encrypt the data using cryptographic techniques before outsourcing those data to the cloud (Kamara and Lauter, 2010). Since that data owner and the co-authors only have the encryption key in collaborative text editing services, unauthorized users are unable to access the shared document.

When the co-authors access to an outsourced document, the collaborative services are in charge of downloading the last version of the document and decrypt it using the appropriated key. On the other hand, by modifying a small part of the file, the owner and the co-authors need to encrypt and decrypt the whole file to obtain the last version of the file. Then, by increasing the size of documents or the number of authors, the required time to encrypt and decrypt the document is also increasing.

Yeh et al. (2013) addressed this issue by proposing an efficient and secure cloud-based collaborative editing approach using the Red-Black tree to reduce the number of data that need to be encrypted. In other words, when an authorized user (the data owner or one of the co-authors) modifies a block of the file, instead of a whole file, the modified block only requires to be encrypted and updated. The Red-Black tree is a type of binary search tree that was developed by Bayer (1972) as a symmetric
binary B-tree to organize a part of text fragments or numbers. The main property of this data structure is that the computation order of search, insert, and delete operations is $O(\log n)$, when $n$ is the number of data blocks (nodes).

Data auditing methods usually assume TPA is a trustworthy agent and they neglect the privacy of data when TPA is involved. However, such assumption is illogical and leads to data leakage. Wang et al. (2012) considered this issue and proposed lightweight TPA protocol on the basis of public key under Homomorphic Linear Authenticator (HIA). The main idea behind of this method is to integrate HLA with random masking technique to protect both data integrity and data privacy. In other words, before transferring the proof message to TPA, the aggregated blocks ($\mu$) under challenge message needs to be blinded by using a random mask as follows:

$$\mu' = \mu + r \cdot h(u^r)^f$$  \hspace{1cm} (2)

where, $\mu'$ is the blinded blocks aggregation, $\mu$ is the aggregated blocks, $r$ is a random mask, $u$ is a cloud public key, $x$ is a cloud private key and $h$ indicates the hash function.

Yang and Jia (2012b) desinged another privacy-preserving auditing for data storage security in cloud computing to address the storage overhead issue in PP-POR (Wang et al., 2012). To this end, they use the Bilinearity property of the bilinear pairing and to generate an encrypted proof the challenge stamp such that the auditor is only able to verify the proof. They also improve the performance of this scheme by using the data fragment technique and HVT to reduce number of data tags, as follows: (1) the input data is divided into $n$ blocks and each data block is split to $s$ sectors by using the data fragment technique, and (2) since that the number of sectors in all block must be same, the number of these sectors in the data blocks that have less sectors must be reached to $s$ by appending the additional sectors with zero content. As a result, the number of data blocks in the input file ($F$) is calculated by:

$$n = \frac{\text{size of } (F)}{s \cdot \log p}$$ \hspace{1cm} (3)

where $p$ is the size of each sector and one data tag is generated for $s$ sectors. For example, when the size of each block is 20 byte, 50 kbyte input file is divided to 2560 data blocks. Therefore, 2560 tags is needed to be generated for this file which incur 50 kbyte storage overhead while by using the data fragment technique, the storage overhead is reduced to 50/s kbyte.

Zhu et al. (2011) introduced a zero knowledge proof model to PDP in order to hinder data leakage during verification step. Their model also supports soundness property based on computation Diffie–Hellman assumption and the rewindable black-box knowledge extractor. The soundness property indicates that the cloud is not able to deceive the verifier to accept false statements. The principal idea behind this scheme is to randomize data blocks and their tags in order to prevent data and tag leakage during verification step.

Since that mobile computing devices have limited processing power, small storage capacity and short battery lifetime, the audit services are required to be efficiently designed for these devices. As a result, Zhu et al. (2012) improved the performance of audit service in two ways: utilizes probabilistic query and periodic verification which helps to balance the computation and communication overhead. They also reduced the size of required storage using an algorithm which selects a number of sectors for each block in the input file.

Wei et al. (2013) was the first to propose a privacy preserving and computation auditing by using Commitment-Based Sampling (CBS) technique (Du et al., 2010) and designated verifier signature (Huang et al., 2011, Zhang and Mao, 2008) to achieve privacy cheating discouragement and minimize the computation cost in cloud computing. To store the input file on the cloud securely, the data owner splits the input file into $n$ blocks ($b_1, b_2, ..., b_n$) and $n$ storage space are allocated to the blocks by CSP. Before transferring the data blocks to the cloud, each data block needs to be signed to enable the data auditing. After generating a secure communication tunnel by using a session key, the data blocks, and corresponding signature are transmitted to the cloud. When the data blocks are received, the CSP decrypts data blocks by using a session key and verifies the signature by using its secret key.

The main contribution of this method is to use the CBS technique based on Merkle Hash Tree (MHT) to provide computation security in two steps: (1) computation request step: in which the positions index of data blocks ($I={i_1, i_2, ..., i_n}$) and a computation service request including a set of some basic functions ($F={f_1, f_2, ..., f_n}$) such as data sum, average, and other complicated computations are submitted to the cloud. (2) commitment generation step: when the computation request is received by the cloud, the requested data blocks based on their position index set are retrieved and considering functions are computed on them ($y_i = f_i(b_i)$). Then, the CSP constructs a MHT with $n$-leaves in which the value of leaves are computed by $V_i = H(V_{leftchild} || H(V_{rightchild}))$. The root of MHT ($R$), its signature and the set of computation results are transferred to the user through transmission tunnel. As a result, the data owner or TPA is able to verify the storage correctness and the computation correctness by using the challenge-response
method and re-building the MHT. Figure 12 shows the steps of this method to provide the security and privacy for storage and computation in cloud computing.

### 4.1.4. Robust data auditing

The most of data auditing methods rely on selecting small portions of the data randomly as a challenge and checking their integrity – is called spot checking. However, this technique is only able to detect a fraction of the data corruption in the server and the client cannot find corruption of small parts of the data. Ateniese et al. (2011) was the first to empower the PDP protocols to mitigate the arbitrary amount of data corruption – is called robust feature – by integrating Forward Error Checking (FEC) with PDP methods. In other words, the input file firstly needs to be encoded by using the FEC technique and then the encoded file is used as an input file to the PDP methods. There are different ways to encode the input file that causes the auditing schemes to include different properties and performance characteristics. The main distinction between these encoding techniques is related to the way of encryption or permutation the data blocks in each constraint group.

#### 1. Simple Reed–Solomon:

To achieve this goal, the input file is divided into $k$-symbol chunks and then each chunk is expanded it to $n$-symbol codeword by applying a $(n, k)$ Reed–Solomon code. The first $k$ symbol in the output is the original file and the remaining $(n - k = d)$ symbols are parity blocks which are used to recover $d$ erased blocks. The constraint group is defined concept as a group of blocks from the same encoding symbols (the original $k$ blocks and their corresponding $d$ parity). However, the attacker is able to manipulate the data by deleting a fixed number of blocks (any $d + 1$ blocks of encrypted file from the same constraint group) due to the fixed number of $k$ and $d$.

#### 2. Permutation all $(\pi A)$:

To overcome the first model, it requires hiding the constraints among blocks. As a result, all blocks of the encoded file should be permuted randomly. However, the resource-intensive nature of this method is slow because of performing permutation on all blocks. Furthermore, the robustness of the file is compromised by allowing sequential access to the original file.

#### 3. Permute-redundancy $(\pi R)$:

In spite of the Permutation all $(\pi A)$, only the parity symbols needs to be permuted and encrypted. The comparison of $(6,4)$ code permutation by the $(\pi R)$ and $(\pi A)$ methods are illustrated in Figure 13.

---

**Fig. 11.** The effect of insert and delete operations on Merkle Hash Tree in the Public PDP method.

**Fig. 12.** Security and privacy for storage and computation in cloud computing.
As mentioned earlier, the Reed–Solomon code is able to provide error correction in a static setting and it needs to hide the relationship between the symbols and the constraint groups by using permutation functions. On the contrary, since that dynamic data update affects on the parity symbols of the corresponding group, the relationship between the symbols and the constraint groups has to be revealed in dynamic methods. Considering a contradiction between dynamic data update and robustness feature, an important question comes to mind: how is it possible to add the robust feature to dynamic provable data possession?

Chen and Curtmola (2012) solved this issue and introduced a robust dynamic provable data possession (RD-PDP) by presenting two new permutation functions such as Permute-redundancy (πR – D), and Variable Length Constrain Group (VLGG).

1. Permute-Redundancy (πR – D): in this scheme, the πR technique is adopted to add the robustness feature to the dynamic PDP method. Since that the content of constraint group depends on the index of data symbol, to insert/delete a data block, the client has to download the whole file and re-compute the parity based on a new set of constrain blocks. However, to modify a data block, the client only needs to download the requested block and the considering parity. After updating the data block and computing the new parity symbol, the parity symbols have to permuted and re-encrypted to preserve the privacy of data against server.

2. Variable Length Constrain Group (VLGG): to overcome the drawback of πR–D technique in insert/delete a data block, symbols are assigned to constraint groups on the basis of the content of symbols instead of the position of symbols in πR – D. As a result, the data owner is able to update – insert and delete operations – the symbol, by only downloading the affected parity and updating the considering parity symbols of the considering constraint group.

They also combined Reed–Solomon codes (Reed and Solomon, 1960) with Cauchy Matrices (Plank and Xu, 2006) in order to reduce communication overhead of RS and support efficient dynamic updates. In addition, the Cauchy Reed Solomon codes are two times faster than classical form of Reed Solomon.

4.2. Proofs of retrievability-based methods

Proof of retrievability (POR) is a type of cryptographic Proof Of Knowledge (POK) to ensure the privacy and integrity of outsourced data in the untrusted cloud without having to download the files. It also provides data recovery and mitigates data corruption by performing Forward Error-correcting Codes (FECs) in which the verifier has capability to recover the file when a considerable fraction of the file is uncorrupted, as proved by spot-checks. The main difference between POR and PDP is the security features which they provide, because in the POR approach the client’s data are completely stored on the server, while the PDP-based methods only guarantee that most of the client’s data are kept in the server and a small portion of the data may be lost by the server. In addition, the POR method stores a redundant encoding of the client data on the server (Cash et al., 2012; Kůpčík, 2010).

4.2.1. Static PDP models

The first POR method was proposed by Juels et al. (2007) on the basis of sentinel blocks – called sentinel – which are concealed among other data blocks before transferring to the cloud. The sentinel blocks are computed by using a one-way function (f) as follows:

\[ \text{sentinels} : \{s_1, s_2, ..., s_m\} \rightarrow s_i = f(\text{key}, i) \]  

Since modifying part of data affects sentinel blocks with a certain probability, the verifier only requires checking whether a random set of sentinel blocks is intact. The main disadvantage of POR scheme is that the number of challenges is limited by the number of embedded sentinel blocks in to file.

Shacham and Waters (2008) designed another POR scheme to improve the efficiency and security of POR protocol and overcome its limitation in terms of number of challenge. The Compact POR method relies on applying the BLS homomorphic signature (Boneh et al., 2004) to aggregate the tags and generate a single short tag as a proof to minimize the network computation overhead – O(t) for t challenges – during checking the integrity of blocks. The authors used the Reed–Solomon code (Plank and Xu, 2006) to support the error recovery in two ways – public verifiability, and private verifiability. The main difference between these two methods is that in private verifiability model the verifier needs to know the DO’s private key in order to validate the proof message rather than the DO’s public key in public verifiability model.

Non-trivial (linear or quadratic) communication complexity is another crucial issue of the existing POR schemes which are designed based on the homomorphic commitment schemes because of the linear sizes of their proves. Yuan and Yu (2013a) proposed a new POR method with constant communication cost by using a constant size polynomial commitment technique (Kate et al., 2010). A polynomial commitment scheme helps the prover to generate a polynomial with a short string as a proof which can be used to audit the cloud. Since the polynomial commitments have a constant size and the overhead of opening a proof is constant, this scheme decreases the communication cost in POR scheme. In order to reduce the complexity of the challenge message of the Public POR method, the authors follow the challenge technique in (Dodis et al., 2009) in which a Hitter

![Fig. 13. The (6,4) code permutation under (πR) and (πA) methods.](image-url)
sampler technique (Goldreich, 1997) is used to randomly select the indices of input file as a challenge.

4.2.2. Dynamic POR models

The main limitation of most POR method is fail to support dynamic data update efficiently as the server is not able to distinguish the relation between the data blocks and encrypted codewords. Cash et al. (2012) overcome this difficulty and implement a dynamic proof of retrievability by combining the POR scheme and ORAM technique (Goldreich and Ostrovsky, 1996; Goodrich et al., 2012). ORAM is a hierarchical data structure which allows the client to read and write from/to the outsource data in a private way by hiding the location of the codewords. This data structure includes several levels of hash tables to hold encrypted address-value pairs while the lower tables have more capabilities. The top tables store the most recently accessed data and the bottom tables keep the least recently used data. When the client wants to read a data block, the address of data is hashed to checks random positions in the remaining tables are checked to hide the proper location in the top table. If the data is found, some addresses of bytes within or between files – known as chunks – and allow the storing of a single instance of each chunk, irrespective of the number of repetitions. In typical storage system that support data deduplication, a client needs to convince the server of having a copy of the outsourced file by sending a hash of the file to the server. If the server finds this hash in the database, accepts the client’s claim and marks the client as the owner of that file; otherwise ask him to upload the entire file. However, this method is vulnerable against some security attacks, because anyone who gets the hash value is permitted to access the file (Halevi et al., 2011; Harnik et al., 2010).

Fairness is another inherent security issue in dynamic POR methods in which a dishonest data owner legally accuses the truthful cloud service provider for tampering its outsourced data. Zheng and Xu (2011) addressed this issue by proposing a fair and dynamic proof of the retrievability method (FD-POR) on the basis of 2–3 range based tree (rb23Tree) and Hash-Compress-and-Sign (HCS). Rb23Tree is an authenticated data structure based on a 2–3 tree in which all intermediate nodes have two or three children and the leaves should be located on the same height. This tree is applied for authenticating a specific value that is stored at a specific leaf. The main feature of this tree is that leaf removal and leaf insertion involve logarithmic complexity. However, balancing rb23Tree is an expensive task and imposes a huge computational burden on the server.

The core idea underlying the FD-POR approach is securing the block index when computing the authentication tag in the compact POR scheme to prevent data leakage. As it is shown in Figure 15, the FD-POR scheme consists of four steps. In the first step, an input file is divided into n blocks \( F = (f_1, f_2, \ldots, f_n) \) and then each block is encrypted by using ECC (\( F = (f_1, f_2, \ldots, f_n, f_{n+1}) \)). In the next step, the encrypted blocks are hashed \( (H_1 = h(f_1)) \) to construct the rb23Tree with leaves \( H_1, H_2, ..., H_n, H_{n+1} \). Finally, a flat tree is constructed to achieve the fairness.

4.3. Proof of ownership

Data deduplication is a type of single-instance data storage (data compression) techniques, which is used to remove data redundancy and duplicate copies of data to provide a cost-efficient storage (Mandagere et al., 2008; Meyer and Bolosky, 2012). Deduplication techniques are responsible to recognize a common set of bytes within or between files – known as chunks – and allow the storing of a single instance of each chunk, irrespective of the number of repetitions. In typical storage system that support data deduplication, a client needs to convince the server of having a copy of the outsourced file by sending a hash of the file to the server. If the server finds this hash in the database, accepts the client’s claim and marks the client as the owner of that file; otherwise ask him to upload the entire file. However, this method is vulnerable against some security attacks, because anyone who gets the hash value is permitted to access the file (Halevi et al., 2011; Harnik et al., 2010).

Halevi et al. (2011) considered these security issues and proposed a deduplication scheme in which the data owner is able to convince the server without transferring the file. This method – that is called proof of ownership – is constructed on the basis of Merkle Hash Tree (MHT) and Collision-Resistant Hash (CRH) functions. Since in POW scheme the client needs to persuade the cloud, the role of prover and verifier is reversed. When the server receives the input file \( F \), he maps it to \( L \) bits using Pairwise Independent Hashing \( X = NH(F) \), and then constructs a Merkle tree based on it \( R = MHT(X) \). The verifier returns the root of MHT and the number of its leaves \( R \text{ MHT, } n_l \) as the verification
information. During the Proof step, the client computes the sibling-paths of all the leaves as a proof of deduplication and sends it to the cloud. Finally, the verifier validates the sibling-paths with regard to MHT(X). Figure 16 shows the core idea behind the proof of the ownership method.

Design a secure data deduplication method to provide simultaneously data auditing and deduplication in cloud computing is contradictory because security and efficiency aspects seem to be two conflicting goals. In other words, deduplication eliminates the identical contents, while data security attempts to encrypt all the contents, and the same contents can then be encrypted by two different keys to create different ciphertexts (Marques and Costa, 2011; Storer et al., 2008).

Zheng and Xu (2012) introduced the first proof of storage with deduplication (POSD) method by integrating two different concepts of Proof of Data Possession (PDP) and proof of ownership (POW) with the purpose of providing both security and efficiency. This method consists of four steps – key generation, uploading, auditing and deduplication. Before uploading the file to the cloud, the honest data owner is responsible to generate two pairs of public and private keys for checking data integrity and deduplication. The data owner divides the file into n block with m bits and then computes the block tags to upload to the cloud. The POSD method includes two challenge–response algorithms such as data auditing, which the data owner validate the correctness of data by sending challenge to the cloud as a prover, and deduplication, which the cloud verifies the claim of a client for having a copy of outsourced data.

The POSD scheme needs to meet the following security requirements: (1) server unforgeability in which the server should provide a valid response to the client challenges with non-negligible probability, and (2) (k, ε) uncheatability in which dishonest clients are not able to deceive the server with non-negligible probability. The validity of the POSD scheme depends on the reliability of the clients in terms of performing key generation, while this assumption is not reasonable in the cross-multiple users and the cross-domain environment of cloud computing.

Shin et al. (2012) shows that dishonest client is able to manipulate the key generation step to create weak keys. As a result, this method fails to fulfill the unforgeability and uncheatability features as security requirements and it is vulnerable to some attack scenarios such as (1) Malware Distribution: a malicious client uploads a file in the cloud and modifies it by attaching a malware to this file. This malware is distributed among clients when they execute the deduplication step to take the ownership of the original file, and (2) Unintended content distribution network: an adversary has capability to transfer the data to the other malicious client through the cloud by using the weak key. Shin et al. (2012) improved the security of POSD scheme by minimizing the client capability to control the key generation step. The core idea behind this method is to blind the keys with random values, when the server receive the data and corresponding tags.

Another disadvantage of the POSD scheme (Zheng and Xu, 2012) is the linear communication and computational cost on the client side regarding to the number of elements in each data block and the number of checking blocks during data auditing step. As a result, by increasing the number of mobile users, the communication and computational cost incur a huge overhead on the client who utilizes the resource constrained devices (e.g. smart mobile phones) to access the cloud storage. Yuan and Yu (2013b) proposed a new data storage auditing with deduplication capability to address the linear communication and computational cost on POSD scheme based on polynomial-based authentication tags and Homomorphic Linear Authenticators. The communication complexity of auditing step in PCAD scheme depends on transferring a challenge message, the aggregation tags of data blocks and the proof information which impose O(1) as a total communication complexity on the client. However, the total communication complexity in POSD method is O(s + k) because the CSP needs to transfer k authentication tags of the challenging blocks and s aggregated data blocks to the verifier, where s is the size of data block.

5. Comparison of remote data auditing protocols

This section compares the current remote data auditing protocols on the basis of the taxonomy presented in Figure 4. The commonalities and differences in such protocols are compared based on the presented parameters in such taxonomy. The comparison parameters considered are: Scheme Nature (SN), Protocol Type (PT), security pattern, cryptography model, batch auditing, public auditing, dependability, and data recovery. Table 1 shows a comparison of remote data auditing protocols based on such parameters and the assumptions and drawbacks of each method, as well.

The attribute of the Scheme Nature indicates the main mechanism used to audit the outsourced data auditing in the single cloud server. The attribute of the security pattern indicates the cryptographic algorithms which are used in the client or the cloud to audit or store data dynamically. The following security patterns are practiced to audit the outsourced data in the current data storage security schemes:

(1) Homomorphic encryption: Data encryption is a crucial method to store and access data securely in the cloud. However, the main issue is how to perform computation on encrypted data received by the cloud server without having to decrypt it and to obtain the same result as performing on the original data. Rivest et al. (1978) was the first to achieve this goal by proposing a homomorphic encryption. In single cloud server, the homomorphic encryption mechanisms are categorized into the following two types: (i) Homomorphic verification tag (HVT) allows the client to combine the computed tags for multiple blocks of each file into a single value (e.g., Ateniese et al., 2007, 2011; Hanser and Slamanig, 2013), (ii) Homomorphic Linear Authentication (HLA) utilizes a linear combination of the individual data block to generate a single value. Since HLA uses a relatively small-sized BLS signature, it incurs less computation overhead than HVT (e.g., Shacham and Waters, 2008; Wang et al., 2011, 2012).

(2) Pairing-based cryptography: The main idea behind this method is to construct a mapping between the elements of two cryptographic groups for building a cryptosystem on the basis of the reduction of one problem in one group. It can be included Cryptographic Bilinear Pairings and Diffie–Hellman assumption (Dutta et al., 2004).

(3) Symmetric Key Cryptography: In this type of cryptography, a shared secret key is used to encrypt the plaintext and decrypt the ciphertext (Delfs and Knebl, 2007).

(4) Polynomial commitment scheme: Helps a committer to commit to a polynomial with a short string that can be utilized by a client to verify claimed evaluations of the committed polynomial.

(5) Computational Diffie–Hellman (CDH) is a valuable assumption for cryptographic purposes and relates to the difficulty of computing the discrete logarithm problem within a cyclic group (Bao et al., 2003). This means that CDH is concerned with the mathematical operations that are completed quickly, but difficult to reverse.
<table>
<thead>
<tr>
<th>SN</th>
<th>PT</th>
<th>Protocols</th>
<th>Security pattern</th>
<th>Cryptography model</th>
<th>Batch auditing</th>
<th>Public auditing</th>
<th>Dependability</th>
<th>Data recovery</th>
<th>Assumption and drawback</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dynamic structure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|    | Static | PDP (Ateniese et al., 2007) | RSA based HVT | Not support | Random oracle | No | Yes | No | No | 1. High computation and communication overhead on the server  
|    |     |           |                  |                |              |               |              |              | 2. Fail to provide secure data possession completely  
|    |     |           |                  |                |              |               |              |              | 3. Leak of information |
|    | ES-PDP (Hanser and Slamanig, 2013) | Elliptic curves cryptography, HVT | Not support | Random oracle | No | Yes | No | No | Only provides a probabilistic guarantee of possession |
|    |     |           |                  |                |              |               |              |              | 1. Lies on the probabilistic security  
|    |     |           |                  |                |              |               |              |              | 2. only provides a probabilistic guarantee of possession |
|    | PPDP (Wang, 2012) | Bilinear pairing technique | Not support | Random oracle | No | No | No | No | Leak of information |
|    | Dynamic | Scalable PDP (Ateniese et al., 2008) | Symmetric key cryptography | XMACC | Random oracle | No | No | No | No | 1. The number of queries is restricted  
|    |     |           |                  |                |              |               |              |              | 2. Cannot provide a dynamic update fully.  
|    |     |           |                  |                |              |               |              |              | 3. Only provides a probabilistic guarantee of possession |
|    | DDPD (I) (Erway et al., 2009) | Homomorphic block tag | Rank-based authentication skip lists | Standard | No | No | Yes | No | 1. Cannot support privacy  
|    |     |           |                  |                |              |               |              |              | 2. Only provides a probabilistic guarantee of possession |
|    | DDPD (II) (Erway et al., 2009) | Homomorphic block tag | Rank-based RSA Tree | Standard | No | No | Yes | No | 1. Higher probability of detection but incurs more computation overhead than D-PDP (I)  
|    |     |           |                  |                |              |               |              |              | 2. Lacks flexibility on the data updates |
|    | Flex-DPDP (Esiner and Kachkeev, 2013) | Homomorphic block tag | Flexible length-based authentication skip list | Standard | No | No | Yes | No | Cannot support privacy |
|    |     |           |                  |                |              |               |              |              |                        |
|    | E-DPDP (Zhang and Blanton, 2012) | Homomorphic block tag | Block update tree | Standard | No | No | Yes | No | Cannot support privacy |
|    |     |           |                  |                |              |               |              |              |                        |
|    | Public PDP (Wang et al., 2011) | HLA | Merkle Hash Tree | Random oracle | Yes | Yes | Yes | No | 1. Leaks the data content to the auditor  
|    |     |           |                  |                |              |               |              |              | 2. incurs heavy computation cost of the auditor |
|    | PP-PDP (Wang et al., 2012) | HLA with random masking | Merkle Hash Tree | Random oracle | Yes | Yes | Yes | No | Incurs a heavy storage overhead on the server because of the large number of data tags  
|    |     |           |                  |                |              |               |              |              | The Index Table incurs huge storage overhead on auditor when the number of file increasing. |
|    | EPP-PDP (Yang and Jia, 2012b) | Bilinear pairing technique, HVT | Index Table | Random oracle | Yes | Yes | No | No |                        |
|    | Privacy-preserving | I-PDP (Zhu et al., 2012) | HVT Polynominal-time rewindable, CDH | Not support | No | No | Yes | No | 1. Support static update  
|    |     |           |                  |                |              |               |              |              | 2. TPA is reliable |
|    | Sec-PDP (Wei et al., 2013) | Commitment-based Sampling (CBS), verifier signature | Not support | Random oracle | Yes | Yes | No | No | Support static update |
|    | Robust | S-PDP (Ateniese et al., 2011) | RSA base HVT | Not support | Random oracle | No | Yes | No | Yes | High computation overhead |
|    |     |           |                  |                |              |               |              |              |                        |
|    | E-PDP (Ateniese et al., 2011) | RSA base HVT | Not support | Random oracle | No | Yes | No | Yes | High computation overhead |
|    |     |           |                  |                |              |               |              |              |                        |
|    | RD-PDP (π−D) (Chen and Curtmola, 2012) | RS codes-based Cauchy matrices | Not support | Random oracle | No | Yes | No | Yes | Higher computation overhead than VLGC |
|    |     |           |                  |                |              |               |              |              |                        |

Table 1: Comparison of remote data auditing protocols on the basis of the basic parameters.
<table>
<thead>
<tr>
<th>SN</th>
<th>PT</th>
<th>Protocols</th>
<th>Security pattern</th>
<th>Cryptography model</th>
<th>Batch auditing</th>
<th>Public auditing</th>
<th>Dependability</th>
<th>Data recovery</th>
<th>Assumption and drawback</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>RD-PDP (VLCG) (Chen and Curtmola, 2012)</td>
<td>RS codes-based Cauchy Matrices, Pf</td>
<td>Rank-based Authentication Skip Lists</td>
<td>Standard</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>POR-based</td>
<td>POR (Juels et al., 2007)</td>
<td>Sentinel-based, RS codes</td>
<td>Not support</td>
<td>Standard</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>Compact POR(I) (Shacham and Waters, 2008)</td>
<td>HLA, pseudo-random function</td>
<td>Not support</td>
<td>Standard</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>Compact POR(II) (Shacham and Waters, 2008)</td>
<td>HLA based BLS signature, CDH</td>
<td>Not support</td>
<td>Random oracle</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>Public POR (Yuan and Yu, 2013a)</td>
<td>Polynomial commitment scheme, CDH Sentinel-based</td>
<td>Not support</td>
<td>Standard</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>DPOR (Cash et al., 2012)</td>
<td>Hash-compress-and-sign Rb23Tree</td>
<td>Not support</td>
<td>Random oracle</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>FD-POR (Zheng and Xu, 2011)</td>
<td>MHT, CRH</td>
<td>Not support</td>
<td>Standard</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>POW-based</td>
<td>POW (Halevi and Harink, 2011)</td>
<td>CDH</td>
<td>Not support</td>
<td>Random oracle</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>POW-based</td>
<td>POSD (Zheng and Xu, 2012)</td>
<td>CDH, random key blender</td>
<td>Not support</td>
<td>Random oracle</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
The second part of security pattern includes some data structures that are used to provide dynamic data update such as:

1. Merkle Hash Tree or Commitment-Based Sampling (MHT) is a very simple and effective model of the authentication structure that is presented as a binary hash tree to detect any tampering (Wang et al., 2011).

2. Range-based 2–3 tree (rb23Tree) is an authenticated data structure based on a 2–3 tree in which all nodes have two or three children, except the leaves which are located on the same height. This tree is applied for authenticating a specific value that is stored at a specific leaf (Zheng and Xu, 2011).

3. Rank-based authentication based on a skip list and RSA tree are two authentication models which are used to check the integrity of file block and to support dynamic data update (Erway et al., 2009).

4. Update Trees: This hash tree has three main features which make it different from the other authenticated data structures such as (i) it is always balanced, (ii) its size is independent of the number and order of dynamic operations or outsourced data blocks on the storage, and (iii) each node in such tree is consist of a range of block indices instead of a certain index in MHT (Zhang and Blanton, 2012).

The attribute of the cryptography model indicates a common methodology to design the cryptographic protocols, including:

1. Standard model: In this model of computation, the complex assumptions such as hash functions (MD5, SHA) are used to prove the security of the scheme, but meantime achieving security proofs in this model is very difficult, (2) Random oracle model: The hash functions in such mode are replaced by a set of truly random functions to prove the security of the ideal system. It is important to mention that whenever a method is secure under the random oracle model, the implementation of this system is also secure in the standard model (Ganetti et al., 2004). The other attributes such as Batch Auditing, Public Auditing, Dependability, and Data Recovery are described in the third section.

To analyze the efficiency of the remote data auditing approaches, there are several metrics that should be considered: (1) Computational cost: Data auditing approaches impose different computation overhead on the client as a verifier and cloud service provider as a prover on the basis of their cryptographic algorithms. Client computation indicates the computational resources that are used by the client to generate the challenge and verify the proof message while server computation denotes the computation resources that the server uses to process an update step or compute a proof for a block. (2) Communication complexity shows the size of the challenge message sent to the prover and the size of proof message received by the verifier. There are two techniques to reduce computation and communication complexity in remote data auditing methods – Sampling and batch auditing. In the sampling technique, the input file is divided into several blocks and a random number of blocks is used to perform batch processing (Erway et al., 2009). The batch auditing technique decreases the size of the proof message by sending a linear combination of random blocks to decrease the communication overhead (Wang et al., 2009). (3) The probability of detection is the last factor which represents the probability of detecting a cloud server’s misbehavior. The comparison of the efficiency between some protocols based on the computation cost, communication cost, and the probability of misbehavior detection are represented in Table 2, where $n$ is the number of blocks of each file, $s$ is the number of sectors of a block, $m$ indicates the number of symbols of a block, $t$ shows the number of blocks that will be changed, $c$ is the number of cloud service providers in multi-cloud, $\rho$ and $\rho_k$ are the probability of block corruption in a cloud server and $k$th server in the multi-cloud, and $\Omega(.)$ is the proof size in the hash function.

### 6. Open issues and challenges

In this section, we highlight some of the most important issues and challenges in deploying and utilizing the remote data storage auditing approaches as the future research directions.

### 6.1. Lightweight data auditing approach for mobile cloud computing

Developing lightweight remote data auditing approaches to improve the security of mobile users without any further limitation and requirement is a significant challenge in mobile cloud computing environment. Dividing the huge files into some blocks, generating the specific tag for each block, computing a challenge, and verifying the proof message are particular tasks in data auditing mechanisms that noticeably increase overall execution time and decrease the lifetime of resource constrained devices such as smartphones and tablets.

A feasible approach to decrease the side effect of remote data auditing approach on mobile devices is to utilize the efficient public verification approach. As a result, the mobile user delegates the challenge and verification steps to the trusted third party to release the mobile devices from the communication and computation overhead of these steps. However, some of remote data auditing approaches (e.g. DPOR (Cash et al., 2012)) are unable to support the public verification technique. On the other hand,

### Table 2

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Client computation</th>
<th>Server computation</th>
<th>Communication complexity</th>
<th>Probability of detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDP (Ateniese et al., 2007)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>O(mn)</td>
<td>1 – (1 – $\rho$)</td>
</tr>
<tr>
<td>Scalable PDP (Ateniese et al., 2008)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>O(t log n)</td>
<td>1 – (1 – $\rho$)^m</td>
</tr>
<tr>
<td>DPDP(I) (Erway et al., 2009)</td>
<td>O(t log n)</td>
<td>O(t log n)</td>
<td>O(t log n)</td>
<td>1 – (1 – $\rho$)</td>
</tr>
<tr>
<td>DPDP(II) (Erway et al., 2009)</td>
<td>O(t log n)</td>
<td>O(t log n)</td>
<td>O(t log n)</td>
<td>1 – (1 – $\rho_k$ log n)</td>
</tr>
<tr>
<td>Public PDP (Wang et al., 2011)</td>
<td>O(t log n)</td>
<td>O(t log n)</td>
<td>O(t log n)</td>
<td>N/A</td>
</tr>
<tr>
<td>PP-PDP (Wang et al., 2012)</td>
<td>O(t log n)</td>
<td>O(t log n)</td>
<td>O(t log n)</td>
<td>1 – (1 – $\rho$)</td>
</tr>
<tr>
<td>EPF-PDP (Yang and Jia, 2012b)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>1 – (1 – $\rho$)</td>
</tr>
<tr>
<td>I-PDP (Zhu et al., 2012)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>1 – (1 – $\rho_k$)</td>
</tr>
<tr>
<td>RDPD (Chen, 2013)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>N/A</td>
</tr>
<tr>
<td>POR (Juels et al., 2007)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>O(mn)</td>
<td>N/A</td>
</tr>
<tr>
<td>Compact POR(I) (Shacham and Waters, 2008)</td>
<td>O(t+ s)</td>
<td>O(t+ s)</td>
<td>O(s)</td>
<td>1 – (1 – $\rho$)</td>
</tr>
<tr>
<td>Compact POR(II) (Shacham and Waters, 2008)</td>
<td>O(t+ s)</td>
<td>O(t+ s)</td>
<td>O(s)</td>
<td>1 – (1 – $\rho$)^m</td>
</tr>
<tr>
<td>DPOR (Cash et al., 2012)</td>
<td>O(t log n)</td>
<td>O(t log n)</td>
<td>O(t log n)</td>
<td>N/A</td>
</tr>
<tr>
<td>POSD (Zheng and Xu, 2012)</td>
<td>O(t)</td>
<td>O(t)</td>
<td>O(m t+n)</td>
<td>N/A</td>
</tr>
</tbody>
</table>
when the security of data is very important, data owners prefer to use the private verification methods. Therefore, implementing lightweight remote data auditing methods demands lightweight computation and communication techniques to be applicable on mobile devices.

6.2. Dynamic data update

To update a single bit of data or changes a bit location of the outsourced data in static data auditing methods, the data owner has to download the whole data, change the location of more than half bits of files, and upload it again to the cloud which incurs high communication overhead on the data owner. Therefore, dynamic data update is a vital feature to almost all remote data auditing approaches in the cloud and mobile cloud computing due to the dynamic nature of the data involved, such as electronic documents and log files. However, one of the main limitations of POR-based and POW-based methods is the need to support dynamic data update (Erway et al., 2009). Although Cash et al. (2012) proposed a first POR-based method to overcome the dynamic data update issue in cloud computing, the lack of public verification feature makes this method impractical for mobile cloud computing.

On the other hand, current dynamic data update methods also impose high storage and computation overhead on data owner – especially on resource restriction devices – resulting from rebalancing the stored tree, increasing the size of the tree. The main approach to address this issue is to develop an update tree with independent size of the outsourced data and without having to balancing procedure. As a result, enabling mobile users to efficiently and dynamically update their outsourced data requires future research and developments.

6.3. Data access control over shared data

Today, many web-based service providers including the web services, blogs, and other web-based application providers tend to outsource their data to the cloud or remote servers. It is clear that users of such services require gaining access to their data anytime, anywhere, and simultaneously performing updating operations such as deleting, modifying or inserting new data to the stored data. For instance, most of the popular blogs which are hosted by a cloud-based server permit their subscribers to delete, append, or remove blog content, freely. However, most of the methods which are designed to ensure data integrity are unable to fulfill these requirements completely or will incur extra computation and high storage overhead on the users.

Data deduplication is a necessary feature of remote data checking mechanisms which has a noticeable effect on data communication and communication cost over cloud user. Though the POW-based approaches permit multi users to get access to the shared data, the users are unable to write, delete or modify the data in the same time. In this context, the remote data auditing methods need to manage multi-user access to the shared data on the cloud without leaking or losing data.

6.4. Data computational integrity

Recently, many data owners tend to outsource an arbitrary computation service to a cloud service provider (Cachin, 2011). Ensuring the integrity outsourced computations enable a client to transfer a computation step of remote data checking in PDP-based, POR-based and POW-based method to another computer and then, without executing the computation, merely checks the integrity of
computation in the new computer (Setty et al., 2011). In other words, the practical and unconditional verification is capable of contributing significantly to the development of remote data auditing approaches by reducing the computational overhead. However, current data integrity methods are unable to support data computation integrity as well. Migrating computational functions along with data into the cloud and using challenge-response approach to verify computation and data integrity can be a possible way to address this issue. This technique, which is useful for resource constraint devices to reduce the computation cost still needs certain degree of attention.

7. Conclusion

Auditing outsourced data in cloud computing is an emerging research area, which has been getting more attention in recent years. Current RDA approaches accomplish data checking process in diverse modes. Several approaches only audit the integrity of outsourced data, while a number of these approaches focus on error recovery and the rest of approaches are able to check the integrity of data instead of the TPA in the verification mode. The ultimate goal of RDA is to preserve the integrity and privacy of outsourced data and computation in single and distributed cloud servers regardless of underlying resource restrictions.

In this paper, we explained the concept of cloud computing, mobile cloud computing and discussed the different techniques used to ensure data integrity and privacy in the cloud and mobile cloud computing. A comprehensive survey on data storage security in cloud computing was presented and several RDA approaches are reviewed with the aim of highlighting the similarities and differences in the thematic taxonomy based on various parameters. We also discussed the issues in the state-of-the-art RDA approaches and focused on the challenges pertaining to the security requirements to provide optimal and lightweight security frameworks. We analyzed all methods in detail, with emphasis on their advantages and disadvantages as well as their significance and requirements. Several open challenges particularly, lightweight data auditing, dynamic data update, data integrity can be a possible way to address this issue. This technique, which is useful for resource constraint devices to reduce the computation cost still needs certain degree of attention.

Acknowledgments

This work was carried out as part of the mobile cloud computing research project funded by the Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant, Reference no. U.M/CIR/MOHE/FS/C03/03.

This work was partly supported by the National Natural Science Foundation of China under Grant no. 61300220.

References

ABI-Research. Mobile cloud applications: Widgets and the apps dilemma for smartphones, netbooks, media tablets, and connected mobile devices. ABI Research; 67.
Arrington M. Gmail disaster: reports of mass email deletions; 2006.
Cellan-Jones. The sidekick cloud disaster; 2009.
Clark GC, Cain JB. Error-correction coding for digital communications. New York: Springer; 11.

