Data Storage Security Service in Cloud Computing: Challenges and Solutions

Alshaimaa Abo-alian, Nagwa L. Badr and Mohamed Fahmy Tolba

Abstract Cloud computing is an emerging computing paradigm that is rapidly gaining attention as an alternative to other traditional hosted application models. The cloud environment provides on-demand, elastic and scalable services, moreover, it can provide these services at lower costs. However, this new paradigm poses new security issues and threats because cloud service providers are not in the same trust domain of cloud customers. Furthermore, data owners cannot control the underlying cloud environment. Therefore, new security practices are required to guarantee the availability, integrity, privacy and confidentiality of the outsourced data. This paper highlights the main security challenges of the cloud storage service and introduces some solutions to address those challenges. The proposed solutions present a way to protect the data integrity, privacy and confidentiality by integrating data auditing and access control methods.

1 Introduction

Cloud computing can be defined as a type of computing in which dynamically scalable resources (i.e., storage, network, and computing) are provided on demand as a service over the Internet. The service delivery model of cloud computing is the set of services provided by cloud computing that is often referred to as an SPI model, i.e., Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). In a SaaS model, the cloud service providers (CSPs) install and operate application software in the cloud and the cloud users can then access the software from cloud clients. The users do not purchase software, but rather rent it for use

A. Abo-alian (✉) · N.L. Badr · M.F. Tolba
Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt
e-mail: a_alian@cis.asu.edu.eg

N.L. Badr
e-mail: dr.nagwabadr@gmail.com

M.F. Tolba
e-mail: fahmytolba@gmail.com

A.E. Hassanien et al. (eds.), *Multimedia Forensics and Security*,
Intelligent Systems Reference Library 115, DOI 10.1007/978-3-319-44270-9_2

on a subscription or pay-per-use model, e.g. Google Docs [1]. The SaaS clients do not manage the cloud infrastructure and platform on which the application is running. In a PaaS model, the CSPs deliver a computing platform which includes the operating system, programming language execution environment, web server and database. Application developers can subsequently develop and run their software solutions on a cloud platform. With PaaS, developers can often build web applications without installing any tools on their computer, and can hereafter deploy those applications without any specialized system administration skills [2]. Examples of PaaS providers are Windows Azure [3] and Google App Engine [4]. The IaaS model provides the infrastructure (i.e., computing power, network and storage resources) to run the applications. Furthermore, it offers a pay-per-use pricing model and the ability to scale the service depending on demand. Examples of IaaS providers are Amazon EC2 [5] and Terremark [6].

Cloud services can be deployed in four ways depending upon the clients' requirements. The cloud deployment models are: public cloud, private cloud, community cloud and hybrid cloud. In the public cloud (or external cloud), a cloud infrastructure is hosted, operated, and managed by a third-party vendor from one or more data centers [2]. The network, computing and storage infrastructures are shared with other organizations. Multiple enterprises can work simultaneously on the infrastructure provided. Users can dynamically provide resources through the internet from an off-site service provider [7]. In the private cloud, cloud infrastructure is dedicated to a specific organization and is managed either by the organization itself or third party service provider. This emulates the concept of virtualization and cloud computing on private networks. Infrastructure, in the community cloud, is shared by several organizations for a shared reason and may be managed by themselves or a third-party service provider. Infrastructure is located at the premises of a third party. Hybrid Cloud consists of two or more different cloud deployment models, bound together by standardized technology, which enables data portability between them. With a hybrid cloud, organizations might run non-core applications in a public cloud, while maintaining core applications and sensitive data in-house in a private cloud [2].

A cloud storage system (CSS) can be considered a network of distributed data centers which typically uses cloud computing technologies like virtualization, and offers some kind of interface for storing data [8]. Data may be redundantly stored at different locations in order to increase its availability. Examples of such basic cloud storage services are Amazon S3 [9] and Rackspace [10]. One fundamental advantage of using a CSS is the cost effectiveness, where data owners avoid the initial investment of expensive large equipment purchasing, infrastructure setup, configuration, deployment and frequent maintenance costs. Instead, data owners pay for only the resources they actually use and for only the time they require them. Elasticity is also a key advantage of using a CSS, as Storage resources could be allocated dynamically as needed, without human interaction. Scalability is another gain of adopting a CSS because Cloud storage architecture can scale horizontally or vertically, according to demand, i.e. new nodes can be added or dropped as needed. Moreover, a CSS offers more reliability and availability, as data owners can access their data from anywhere and at any time. Furthermore, Cloud service providers use several replicated sites for business continuity and disaster recovery reasons.

Despite the appealing advantages of cloud storage services, they also bring new and challenging security threats towards users outsourced data. Since cloud service providers (CSPs) are separate administrative entities, the correctness and the confidentiality of the data in the cloud is at risk due to the following reasons: First, Since cloud infrastructure is shared between organizations, it is still facing the broad range of both internal and external threats for data integrity, for example, outages of cloud services such as the breakdown of Amazon EC2 in 2010 [11]. Second, users no longer physically possess the storage of their data, i.e., data is stored and processed remotely. So, they may worry that their data could be misused or accessed by unauthorized users [12]. For example, a dishonest CSP may sell the confidential information about an enterprise to the enterprise's closest business competitors for profit. Third, there are various motivations for the CSP to behave disloyally towards the cloud users regarding their outsourced data status. For example, the CSP might reclaim storage for monetary reasons by discarding data that has not been or is rarely accessed or even hide data loss incidents to maintain a reputation [13]. In a nutshell, although outsourcing data to the cloud is economically attractive for long-term large-scale storage, the data security in cloud storage systems is a prominent problem. Cloud storage systems do not immediately offer any guarantee on data integrity, confidentiality and availability. As a result, the CSP should adopt data security practices to ensure that their (clients) data is available, correct and safe from unauthorized access and disclosure.

Downloading all the data and checking on retrieval is the traditional method for verifying the data integrity but it causes high transmission costs and heavy I/O overheads. Furthermore, checking data integrity when accessing data is not sufficient to guarantee the integrity and availability of all the stored data in the cloud because there is no assurance for the data that is rarely accessed. Thus, it is essential to have storage auditing services that verify the integrity of outsourced data and to provide proof to data owners that their data is correctly stored in the cloud.

Traditional server-based access control methods such as Access Control List (ACL) [14] cannot be directly applied to cloud storage systems because data owners do not fully trust the cloud service providers. Additionally, traditional cryptographic solutions cannot be applied directly while sharing data on cloud servers because these solutions require complicated key management and high storage overheads on the server. Moreover, the data owners have to stay online all the time to deliver the keys to new users. Therefore, it is crucial to have an access control method that restricts and manages access to data and ensure that the outsourced data is safe from unauthorized access and disclosure [15].

In this paper, an extensive survey of cloud storage auditing and access control methods is presented. Moreover, an evaluation of these methods against different performance criteria is conducted. The rest of the paper is organized as follows. Section 2 overviews various cloud storage auditing methods. Section 3 presents some literature for access control methods in cloud computing. A comparative analysis of some existing data security methods in cloud computing is provided in Sect. 4.

Section 5 discusses the limitations of different data security methods in cloud computing and provides some concluding remarks that can be used in designing new data security practices. Finally, we conclude in Sect. 6.

2 Data Storage Auditing Methods

This section first defines the system model and the security model of data storage auditing schemes within cloud computing. Then, the existing data storage auditing methods, that are classified into different categories, are presented.

Data storage auditing can be defined as a method that enables data owners to check the integrity of remote data without downloading the data or explicit knowledge of the entire data [16]. Any system model of auditing scheme consists of three entities as mentioned in [17]:

1. Data Owner: An entity which has large data files to be stored in the cloud and can be either individual consumers or organizations.
2. Cloud Storage Server (CSS): An entity which is managed by a Cloud Service Provider (CSP) and has significant storage space and computation resources to maintain clients data.
3. Third Party Auditor or Verifier (TPA): An entity which has the expertise and capabilities to check the integrity of data stored on CSS.

In the security model of most data auditing schemes, the auditor is assumed to be honest-but-curious. It performs honestly during the entire auditing protocol but it is curious about the received data. So, it is essential to keep the data confidential and invisible to the auditor during the auditing protocol, but the cloud storage server could be dishonest and may launch the following attacks [18]:

1. Replace Attack: The server may choose another valid and uncorrupted pair of data block and data tag (m_k, t_k) to replace the challenged pair of data block and data tag (m_i, t_i), when it has already discarded m_i or t_i.
2. Forge Attack: The server may forge the data tag of data block and deceive the auditor if the owners secret tag keys are reused for the different versions of data.
3. Replay Attack: The server may generate the proof from the previous proof or other information, without retrieving the actual owners data.

As illustrated in Fig. 1, a data auditing scheme should basically consist of five algorithms:

1. Key Generation: It is run by the data owner. It takes as input security parameter 1^λ and outputs a pair of private and public keys (sk, pk).
2. Tag Generation: It is run by the data owner to generate the verification metadata, i.e., data block tags. It takes as inputs a public key pk, a secret key sk and the file blocks b. It outputs the verifiable block tags T_b.

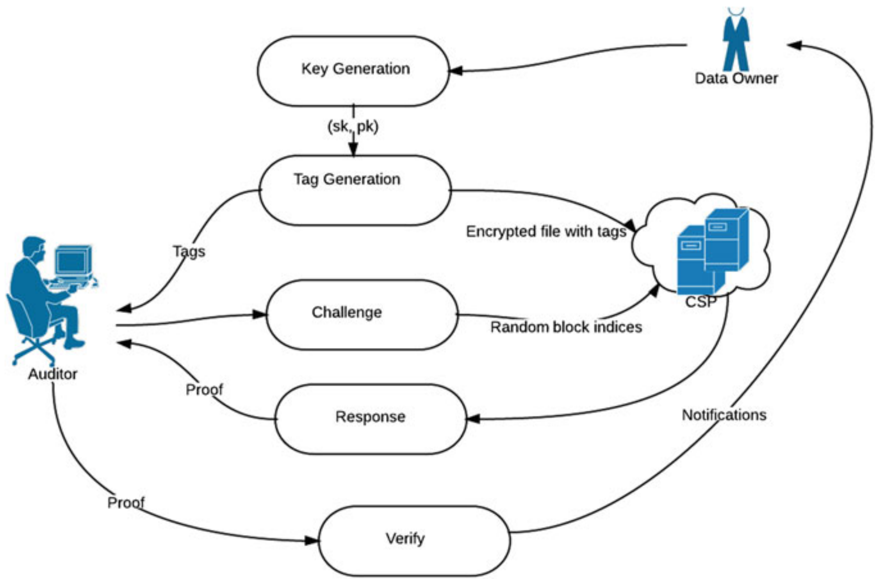

Fig. 1 Structure of an auditing scheme

3. Challenge: It is run by the auditor in order to randomly generate a challenge that indicates the specific blocks. These random blocks are used as a request for a proof of possession.
4. Response/Proof: It is run by the CSP, upon receiving the challenge, to generate a proof that is used in the verification. In this process, the CSP proves that it is still correctly storing all file blocks.
5. Verify: It is run by the auditor in order to validate a proof of possession. It outputs TRUE if the verification equation passed, or FALSE otherwise.

The existing data storage auditing schemes can be basically classified into two main categories:

1. Provable Data Possession (PDP) methods: For verifying the integrity of data without sending it to untrusted servers, the auditor verifies probabilistic proofs of possession by sampling random sets of blocks from the cloud service provider [19].
2. Proof of Retrievability (PoR) methods: A set of randomly-valued check blocks called sentinels are embedded into the encrypted file. For auditing the data storage, the auditor challenges the server by specifying the positions of a subset of sentinels and asking the server to return the associated sentinel values [20].

The existing data storage auditing methods can be further classified into several categories according to:

i. The type of the auditor/verifier: Public auditing or private auditing.
ii. The distribution of data to be audited: Single-copy or multiple-copy data.
iii. The data persistence: Static or dynamic data.

2.1 Public Auditing Versus Private Auditing

Considering the role of the auditor in the auditing model, data storage auditing schemes fall into two categories; private auditing and pubic auditing [21]. In private auditing, only data owners can challenge the CSS to verify the correctness of their outsourced data [22]. Unfortunately, private auditing schemes have two limitations: (a) They impose an online burden on the data owner to verify data integrity and (b) The data owner must have huge computational capabilities for auditing. Examples of auditing schemes that only support private auditing are [23–26]. In Public auditing or Third party auditing [27], data owners are able to delegate the auditing task to an independent third party auditor (TPA), without the devotion of their computational resources. However, pubic auditing schemes should guarantee that the TPA keeps no private information about the verified data. Several variations of PDP schemes that support public auditing such as [17, 24, 25, 28], were proposed under different cryptographic primitives.

2.2 Auditing Single-Copy Versus Multiple-Copy Data

For verifying the integrity of outsourced data in the cloud storage, various auditing schemes have been proposed which can be categorized into:

i. Auditing schemes for single-copy data.
ii. Auditing schemes for multiple-copy data.

i. Auditing Schemes for Single-Copy Data

Shacham and Waters [29] proposed two fast PoR schemes based on an homomorphic authenticator that enables the storage server to reduce the complexity of the auditing by aggregating the authentication tags of individual file blocks. The first scheme is built from BLS signatures and allows public auditing. The second scheme is built on pseudorandom functions and allows only private auditing but its response messages are shorter than those of the first scheme, i.e., 80 bits only. Both schemes use Reed-Solomon erasure encoding method [30] to support an extra feature which allows the client to recover the data outsourced in the cloud. However, their encoding and decoding are slow for large files.

Yuan and Yu [31] managed to suppress the need for the tradeoff between communication costs and storage costs. They proposed a PoR scheme with public auditing and constant communication cost by combining techniques such as constant size polynomial commitment, BLS signatures and homomorphic linear authenticators. On the other hand, their data preparation process requires $(s + 3)$ exponentiation operations where s is the block size. However, their scheme does not support data dynamics.

Xu and Chang [32] proposed a new PDP model called POS. POS requires no modular exponentiation in the setup phase and uses a smaller number, i.e., about 102 for a 1G file, of group exponentiation in the verification phase. POS only supports private auditing for static data and its communication cost is linear in relation to the number of encoded elements in the challenge query.

Ateniese et al. [33] introduced a PDP model supported with two advantages: Lightweight and robustness. Their challenge/response protocol transmits a small, and constant amount of data, which minimizes network communication. Furthermore, it incorporates mechanisms for mitigating arbitrary amounts of data corruption. On the other hand, It relies on Reed-Solomon encoding scheme in which the time required for encoding and decoding of n-block file is $O(n^2)$.

Cao et al. [34] proposed a public auditing scheme that is suitable for distributed storage systems with concurrent users access. In order to efficiently recover the exact form of any corrupted data, they utilized the exact repair method [35] where the newly generated blocks are the same as those previously stored blocks. So, no verification tags need to be generated on the fly for the repaired data. Consequently, it relieves the data owner from the online burden. However, their scheme increases the storage overheads at each server, uses an additional repair server to store the original packets besides the encoded packets, and does not support data dynamics.

ii. Auditing Schemes for Multiple-Copy Data

Barsoum and Hasan [36, 37] proposed two dynamic multi-copy PDP schemes: Tree-Based and Map-Based Dynamic Multi-Copy PDP (TB-DMCPDP and MB-DMCPDP, respectively). These schemes prevent the CSP from cheating and maintaining fewer copies, using the diffusion property of the AES encryption scheme. TB-DMCPDP scheme is based on Merkle hash tree (MHT) whereas MB-DMCPDP scheme is based on a map-version table to support outsourcing of dynamic data. The setup cost of the TB-DMCPDP scheme is higher than that of the MB-DMCPDP scheme. On the other hand, the storage overhead is independent of the number of copies for the MB-DMCPDP scheme, while the storage overhead is linear with the number of copies for the TB-DMCPDP scheme. However, the authorized users should know the replica number in order to generate the original file which may require the CSP reveal its internal structure to the users.

Zhu et al. [38] proposed a co-operative provable data possession scheme (CPDP) for multi-cloud storage integrity verification along with two fundamental techniques: Hash index hierarchy (HIH) and homomorphic verifiable response (HVR). Using the hash index hierarchy, multiple responses of the clients challenges that are computed from multiple CSPs, can be combined into a single response as the final result. Homomorphic verifiable response supports distributed cloud storage in a multi-cloud storage environment, and implements an efficient collision-resistant hash function.

Wang and Zhang [39] proved that the CPDP [38] is insecure because it does not satisfy the knowledge soundness, i.e., any malicious CSP or malicious organizer is able to pass the verification even if they have deleted all the stored data. Additionally, the CPDP does not support data dynamics.

Etemad and Kupcu [26] proposed a distributed and replicated DPDP (DR-DPDP) that provides transparent distribution and replication of the data over multiple servers where the CSP may hide its internal structure from the client. This scheme uses persistent rank-based authenticated skip lists to handle dynamic data operations more efficiently, such as insertion, deletion, and modification. On the other hand, DR-DPDP has three noteworthy disadvantages: First, it only supports private auditing. Second, it does not support recovery of corrupted data. Third, the organizer looks like a central entity that may get overloaded and may cause a bottleneck.

Mukundan et al. [24] proposed a Dynamic Multi-Replica Provable Data Possession scheme (DMR-PDP) that uses the Paillier probabilistic encryption for replica differentiation so that it prevents the CSP from cheating and maintaining fewer copies than what is paid for. DMR-PDP also supports efficient dynamic operations such as block modification, insertion and deletion on data replicas over cloud servers. However, it supports only private auditing and does not provide any security proofs.

Chen and Curtmola [25] proposed a remote data checking scheme for replication-based distributed storage systems, called RDC-SR. RDC-SR enables server-side repair and places a minimal load on the data owner who only has to act as a repair coordinator. In RDC-SR, each replica constitutes a masked/encrypted version of the original file in order to support replica differentiation. In order to overcome the replicate-on-the-fly (ROTF) attack, they make replica creation a more time-consuming process. However, RDC-SR has three remarkable limitations: First, the authorized users must know the random numbers used in the masking step in order to generate the original file. Second, it only supports private auditing. Third, it works only on static data.

2.3 Static Versus Dynamic Data Auditing

Considering the data persistence, existing auditing schemes can be categorized into: Auditing schemes that support only static archived data such as; [3, 5, 25] and auditing schemes that support data dynamics such as; insertion, deletion and modification. For enabling dynamic operations, existing data storage schemes utilize different authenticated data structures including: (i) Merkle hash tree [40], (ii) balanced update tree [41], (iii) skip-list [42–44] and (iv) map-version table (index table) [25, 45] that are illustrated in detail, as follows:

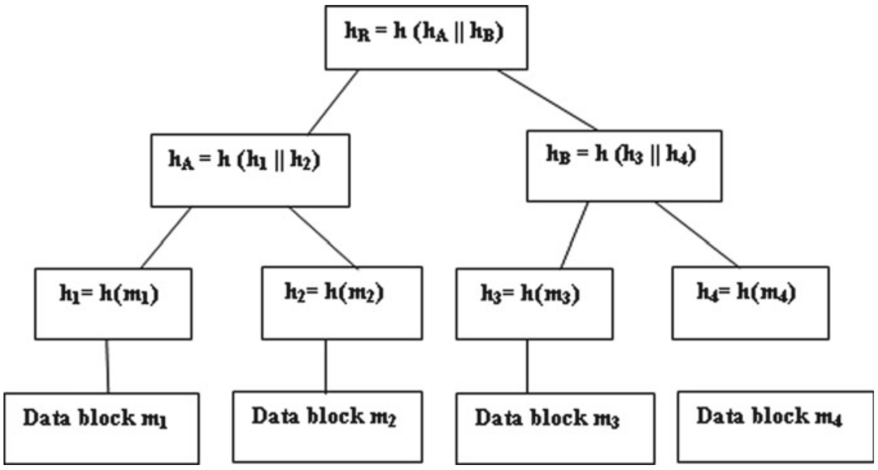

Fig. 2 Merkle hash tree [40]

i. Merkle Hash Tree

Merkle Hash Tree (MHT) [40] is a binary tree structure used to efficiently verify the integrity of the data. As illustrated in Fig. 2, the MHT is a tree of hashes where the leaves of the tree are the hashes of the data blocks. Wang et al. [46] proposed a public auditing protocol that supports fully dynamic data operations by manipulating the classic Merkle Hash Tree construction for block tag authentication in order to achieve efficient data dynamics. They could also achieve batch auditing where different users can delegate multiple auditing tasks to be performed simultaneously by the third party auditor (TPA). Unfortunately, their protocol does not maintain the privacy of the data and TPA could derive users data from the information collected during the auditing process. Additionally, their protocol does not support data recovery in case of data corruption.

Lu et al. [47] addressed the security problem of the previous auditing protocol [46] in the signature generation phase that allows the CSP to deceive by using blocks from different files during verification. They presented a secure public auditing protocol based on the homomorphic hash function and the BLS short signature scheme which is publicly verifiable and supports data dynamics, it also preserves privacy. However, their protocol suffers from massive computational and communication costs.

Wang et al. [48] proposed a privacy-preserving public auditing scheme using random masking and homomorphic linear authenticators (HLAs) [49]. Their auditing scheme also supports data dynamics using Merkle Hash Tree (MHT) and it enables the auditor to perform audits for multiple users simultaneously and efficiently. Unfortunately, their scheme is vulnerable to TPA offline guessing attack.

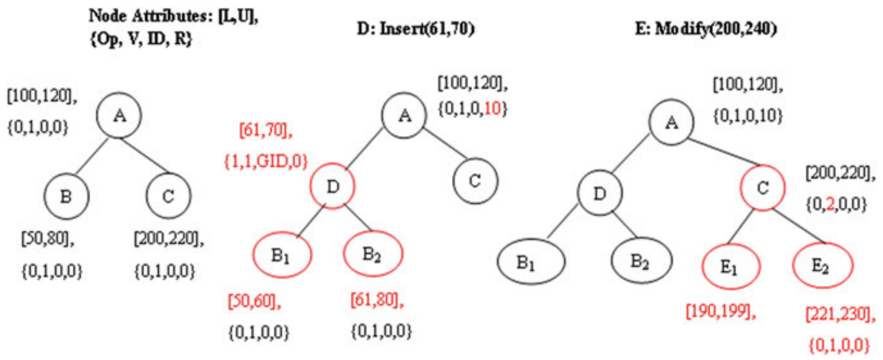

Fig. 3 Example of balanced update tree operations [41]

ii. Balanced Update Tree

Zhang and Blanton [41] proposed a new data structure called “balanced update tree,” to support dynamic operations while verifying data integrity. In the update tree, each node corresponds to a range of data blocks on which an update (i.e., insertion, deletion, or modification) has been performed. The challenge with constructing such a tree was to ensure that: (i) a range of data blocks can be efficiently located within the tree and (ii) the tree is maintained to be balanced after applying necessary updates caused by clients queries, i.e., the size of this tree is independent of the overall file size as it depends on the number of updates. However, it introduces more storage overhead on the client. Besides, the auditing scheme requires the retrieval, i.e., downloading, of data blocks which leads to high communication costs. Figure 3 illustrates an example of balanced update tree operations.

iii. Skip List

A skip list [42] is a hierarchical structure of linked lists that is used to store an ordered set of items. An authenticated skip list [43] is constructed using a collision-resistant hash function and keeps a hash value in each node. Due to the collision resistance of the hash function, the hash value of the root can be used later for validating the integrity. Figure 4 [44] illustrates an example of a rank-based authenticated skip list, where the number inside the node represents its rank, i.e., the number of nodes at the bottom level that can be reached from that node. Lu et al. [47] used the skip-list structure to support data dynamics in their PDP model that reduces the computational and communication complexity from log(n) to constant. However, the use of a skip-list creates some additional storage overheads, i.e., about 3.7 % of the original file at the CSP-side and 0.05 % at the client-side. Additionally, it only supports private auditing.

iv. Index Table

A map-version table or an index table is a small data structure created by the owner and stored on the verifier side to validate the integrity and consistency of all file

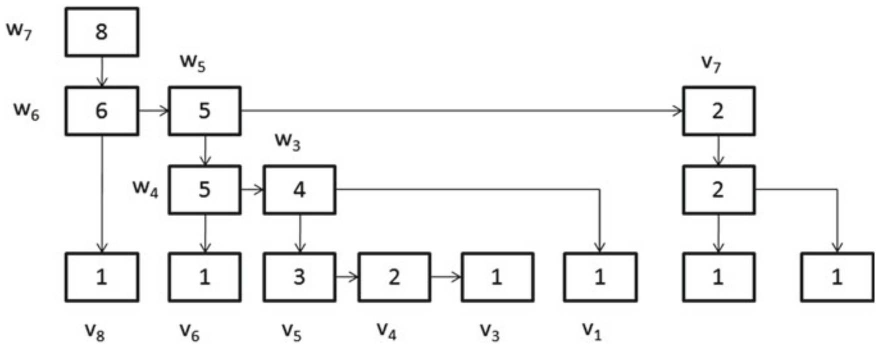

Fig. 4 Rank-based authenticated skip list [44]

copies stored by the CSP [36]. The map-version table consists of three columns: Serial number (SN), block number (BN), and version number (VN). The SN represents an index to the file blocks that indicates the physical position of a block in a data file. The BN is a counter used to make logical numbering/indexing to the file blocks. The VN indicates the current version of file blocks.

Li et al. [50, 51] proposed a full data dynamic PDP scheme that uses a map-version table to support data block updates. Then, they discussed how to extend their scheme to support other features, including public auditing, privacy preservation, fairness, and multiple-replica checking.

Recently, Yang and Xiaohua [18] presented Third-party Storage Auditing Scheme (TSAS) which is a privacy-preserving auditing protocol. It also supports data dynamic operations using Index Table (ITable). Moreover, they add a new column to the index table, Ti, that is the timestamp used for generating the data tag to prevent the replay attack. They applied batch auditing for integrity verification for multiple data owners in order to reduce the computational cost on the auditor. However, this scheme moved the computational loads of the auditing from the TPA to the CSP, i.e., pairing operation, which introduced high computational cost on the CSP side.

3 Access Control Methods

Despite of the cost effectiveness and reliability of cloud storage services, data owners may consider them as an uncertain storage pool outside the enterprises. Data owners may worry that their data could be misused or accessed by unauthorized users as a reason of sharing the cloud infrastructure. An important aspect for the cloud service provider is to have in place an access control method to ensure the confidentiality and privacy of their data, i.e., their data are safe from unauthorized access and disclosure.

Generally, access control can be defined as restricting access to resources/data to privileged/authorized entities [52]. Various access control models have been emerged, including Discretionary Access Control (DAC) [53], Mandatory Access Control (MAC) [54], and Role Based Access Control (RBAC) [55]. In these models, subjects (e.g. users) and objects (e.g. data files) are identified by unique names and access control is based on the identity of the subject, or its roles. DAC, MAC and RBAC are effective for closed and relatively unchangeable distributed systems that deal only with a set of known users who access a set of known services where the data owner and the service provider are in the same trust domain [56].

In cloud computing, the relationship between services and users is more ad hoc and dynamic, service providers and users are not in the same security domain. Users are usually identified by their characteristics or attributes rather than predefined identities [56]. Therefore, the cryptographic solution, such as data encryption before outsourcing, can be the trivial solution to keep sensitive data confidential against unauthorized users and untrusted CSPs. Unfortunately, cryptographic solutions only are inefficient to encrypt a file to multiple recipients, and fail to support fine-grained access control, i.e., granting differential access rights to a set of users and allowing flexibility in specifying the access rights of individual users. Moreover, traditional ACL-based access control methods require attaching a list of authorized users to every data object. When ACLs are enforced with cryptographic methods, the complexity of each data object in terms of its ciphertext size and/or the corresponding data encryption operation is linear to the number of users in the system, and thus makes the system less scalable [57].

The following sections overview the existing access control methods and highlight their main advantages and drawbacks.

3.1 Traditional Encryption

One method for enforcing access control and assuring data confidentiality is to store sensitive data in encrypted form. Only users authorized to access the data have the required decryption key. There are two main classes of encryption schemes: (i) Symmetric key encryption and (ii) Public key encryption [45]. There are several schemes [58–60] proposed in the area of access control of outsourced data addressing the similar issue of data access control with conventional symmetric-key cryptography or public-key cryptography. Although these schemes are suitable for conventional file systems, most of them are less suitable for fine-grained data access control in large-scale data centers which may have a large number users and data files. Obviously, neither of them should be applied directly while sharing data on cloud servers, since they are inefficient in encrypting a file to multiple recipients in terms of key size, ciphertext length and computational cost for encryption. Moreover, they fail to support fine-grained attribute-based access control and key delegation.

3.2 Broadcast Encryption

In broadcast encryption (BE), a sender encrypts a message for some subset S of users who are listening on a broadcast channel, so that only the recipients in S can use their private keys to decrypt the message. The problem of practical broadcast encryption was first formally studied by Fiat and Naor in 1994 [61]. The BE system is secure against a collusion of k users, which means that it may be insecure if more than k users collude.

Since then, several solutions have been described in the literature such as schemes presented in Halevy and Shamir [62] and Boneh et al. [63] which are the best known BE schemes. However, the efficiency of both schemes depends on the size of the authorized user set. Additionally, they also require the broadcaster/sender to refer to its database of user authorizations.

Delerable et al. [64] proposed a dynamic public-key broadcast encryption that simultaneously benefit from the following properties: Receivers are stateless; encryption is collusion secure for arbitrarily large collisions of users and security is tight in the standard model; new users can join dynamically (i.e. without modification of user decryption keys and ciphertext size).

Recently, Kim et al. [65] proposed a semi-static secure broadcast encryption scheme with constant-sized private keys and ciphertexts that improves the scheme introduced by Gentry and Waters [66]. They reduce the private key and ciphertext size by half. In addition, the sizes of the public key and the private key do not depend on the total number of users. Unfortunately, it is only secure against adaptive chosen plaintext attacks (CPA).

Apparently, a BE system achieves an one-to-many encryption with general performance. However, it may not be applied directly while sharing data on cloud servers, since it fails to support attribute-based access control and key delegation.

3.3 Identity-Based Encryption

Identity-based encryption (IBE) was proposed by Boneh et al. in 1984 [63]. But, IBE remained an open problem for many years until a fully functional identity based encryption (IBE) scheme proposed by Halevy and Shamir [62]. It can be defined as a type of public-key cryptography (PKC), in which any arbitrary string corresponding to unique user information is a valid public key such as; an email address or a physical IP address. The corresponding private key is computed by a trusted third party (TTP) called the private key generator (PKG) as illustrated in Fig. 5 [67]. Compared with the traditional PKC, the IBE system eliminates online look-ups for the recipients authenticated public key. However, an IBE system introduces several problems: First, there is only one PKG to distribute private keys to each user which introduces the key escrow problem, i.e., the PKG knows the private keys of all users and may decrypt any message. Second, the PKG is a centralized entity; it may get overloaded and can

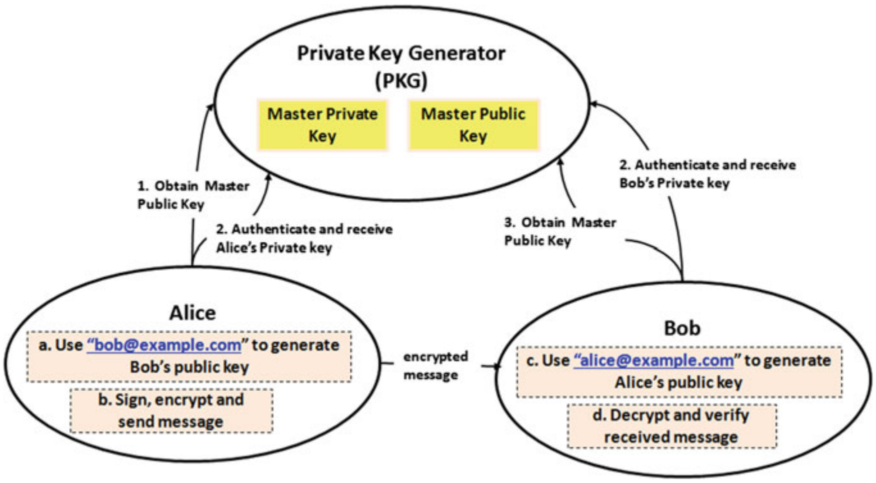
Private Key Generator (PKG)
Master Private Key
Master Public Key
1. Obtain Master Public Key
2. Authenticate and receive Alice's Private key
2. Authenticate and receive Bob's Private key
3. Obtain Master Public Key
Alice
a. Use "bob@example.com" to generate Bob's public key
b. Sign, encrypt and send message
encrypted message
Bob
c. Use "alice@example.com" to generate Alice's public key
d. Decrypt and verify received message

Fig. 5 Example of an identity-based encryption [67]

cause a bottleneck. Third, if the PKG server is compromised, all messages used by that server are also compromised. Recently, Li et al. [68] proposed a revocable IBE scheme that handle the critical issue of overhead computation at the Private Key Generator (PKG) during user revocation. They employ a hybrid private key for each user, in which an AND gate is involved to connect and bound the identity component and the time component. At first, the user is able to obtain the identity component and a default time component, i.e., PKG can issue his/her private key for a current time period. Then, unrevoked users needs to periodically request on a key-update for the time component to a newly introduced entity named Key Update Cloud Service Provider (KU-CSP) which introduces further communication costs.

3.4 Hierarchical Identity-Based Encryption

Horwitz and Lynn [69] introduced the concept of a Hierarchical identity-based encryption (HIBE) system in order to reduce the workload on the root PKG. They proposed a two-level HIBE scheme, in which a root PKG needs only to generate private keys for domain-level PKGs that, in turn generates private keys for all the users in their domains at the next level. That scheme has a chosen ciphertext security in the random oracle model. In addition, it achieves total collusion resistance at the upper levels and partial collusion resistance at the lower levels.

Gentry and Halevi [70] proposed a HIBE scheme with total collusion resistance at an arbitrary number of levels, which has chosen ciphertext security in the random oracle model under the BDH assumption and key randomization. It is noteworthy that their scheme has a valuable feature which is one-to-many encryption, i.e., an

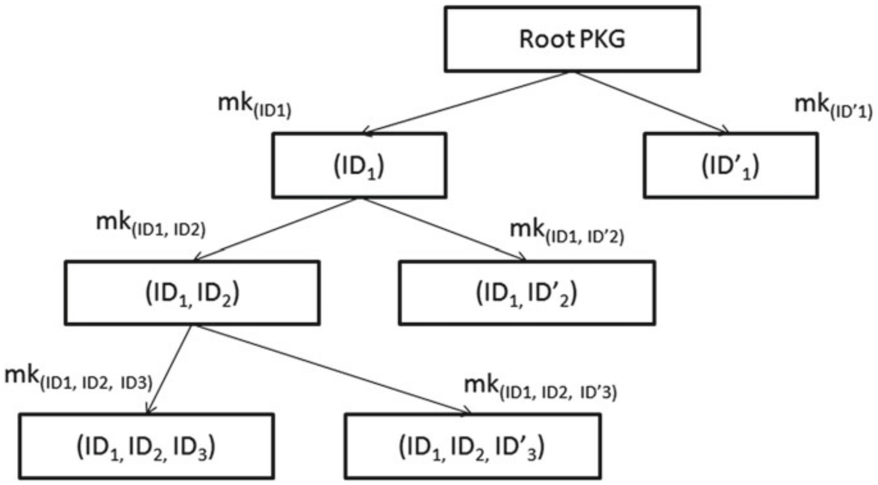

Fig. 6 Hierarchical identity-based encryption [71]

encrypted file can be decrypted by a recipient and all his ancestors, using their own secret keys, respectively. However, the length of ciphertext and private keys, as well as the time of encryption and decryption, grows linearly with the depth of a recipient in the hierarchy.

Figure 6 [71] illustrates an example of HIBE in which the root PKG generates the system parameters for the HIBE system and the secret keys for the lower-level PKGs, which, in turn generate the secret keys for the entities in their domains at the bottom level. In other words, a user public key is an ID-tuple, which consists of the users identity (ID) and the IDs of the users ancestors. Each PKG uses its secret keys (including a master key and a private key) and a user public key to generate secret keys for each user in its domain. Liu et al. [72] utilized the 'one-to-many' encryption feature of [70] and proposed an efficient sharing of the secure cloud storage services scheme. In their scheme, a sender can specify several users as the recipients for an encrypted file by taking the number and public keys of the recipients as inputs of a HIBE system. Using their scheme, the sender needs to encrypt a file only once, and store only one copy of the corresponding ciphertext regardless of the number of intended recipients. The limitation of their scheme is that, the length of ciphertexts grows linearly with the number of recipients, so that it can only be used in the case of a confidential file involving a small set of recipients.

Recently, Mao et al. [73] presented a new HIBE system where the ciphertext sizes as well as the decryption costs are independent of the hierarchy depth, i.e., constant length of ciphertext and a constant number of bilinear map operations in decryption. Moreover, their scheme is fully secure in the standard model. The HIBE system obviously achieves key delegation and some HIBE schemes achieve one-to-many encryption with adequate performance. However, it may not be applied directly while sharing data on cloud servers because it fails to efficiently support fine-grained access control.

3.5 *Attribute-Based Encryption*

An Attribute-Based Encryption (ABE) scheme is a generalization of the IBE scheme. In an IBE system, a user is identified by only one attribute, i.e., the ID. While, in an ABE scheme, a user is identified by a set of attributes, e.g. specialty, department, location, etc. Sahai and Waters [74] first introduced the concept of the ABE schemes, in which a sender encrypted a message, specifying an attribute set and a number d, so that only a recipient who has at least d attributes of the given attributes can decrypt the message. Although their scheme, which is referred to as a threshold encryption, is collusion resistant and has selective-ID security, it has three drawbacks: First, it is difficult to define the threshold, i.e., the minimum number of attributes that a recipient must have to decrypt the ciphertext. Second, revoking a user requires redefining the attribute set. Third, it lacks expressibility which limits its applicability to larger systems.

As an extension of the ABE scheme, two variants are proposed in the literature: (i) the Key-Policy based ABE (KP-ABE) scheme and (ii) the Ciphertext-Policy based ABE (CP-ABE) scheme.

i. Key-Policy Attribute-Based Encryption

A key-policy attribute-based encryption (KP-ABE) scheme was first proposed by Goyal et al. [75] which supports any monotonic access formula consisting of AND, OR, or threshold gates. Their scheme is considered a fine-grained and expressive access control. KP-ABE is a scheme in which the access structure or policy is specified in the users' private keys, while ciphertexts are associated with sets of descriptive attributes.

As stated in [76], any monotonic access structure can be represented as an access tree over data attributes. For example, Fig. 7 presents an access structure and attribute sets that can be generated in a healthcare application [77]. The data owner encrypts the data file using a selected set of attributes (e.g., diabetes, A, asian, etc.) before

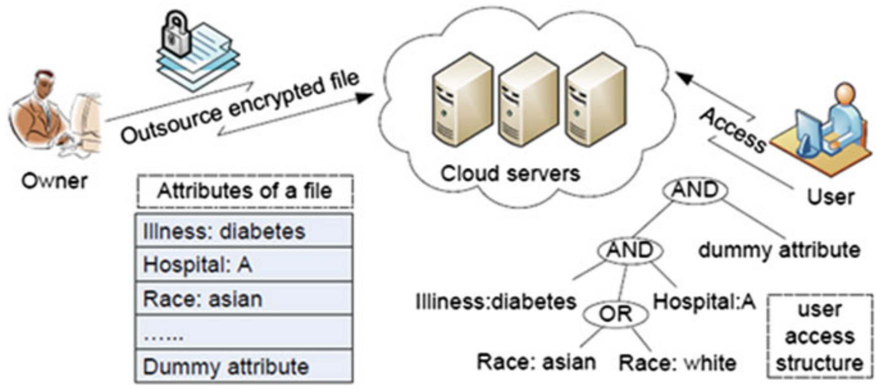

Fig. 7 Key policy attribute-based encryption in a healthcare system [77]

uploading it to the cloud. Only users, whose access structure specified in their private keys matching the file attributes can decrypt the file, In other words, user with access structure as follows: diabetes (asian V white) A can decrypt the data file encrypted under the attributes diabetes, A, Asian. In recent times, [77, 78] proposed a user private key revocable Key Policy Attribute Based Encryption (KP-ABE) scheme with non-monotonic access structure, which can be combined with the XACML policy [79] to address the issue of moving the complex access control process to the cloud and constructing a security provable public verifiable cloud access control scheme.

All of the prior work described above assume the use of a single trusted authority (TA) that manages, e.g., add, issue, revoke, etc., all attributes in the system domain. This assumption not only may create a load bottleneck, but also suffers from the key escrow problem as the TA can decrypt all the files, causing a privacy disclosure. Thus, Chase [80] provided a construction for a multi-authority ABE scheme which supports many different authorities operating simultaneously, and each administering a different set of domain attributes, i.e., handing out secret keys for a different set of attributes [81]. However, his scheme is still not ideal because there are three main problems: first, there is a central authority that can decrypt all ciphertexts because it masters the system secret keys and thus a key-escrow problem is aroused; second, it is very easy for colluding authorities to build a complete profile of all of the attributes corresponding to each global identifier (GID); third, the set of authorities is pre-determined.

Chase and Chow [82] proposed a more practical multi-authority KP-ABE system, which removes the trusted central authority to preserve the user's privacy. Their scheme allows the users to communicate with AAs via pseudonyms instead of having to provide their GIDs. Moreover, they prevent the AAs from pooling their data and linking multiple attribute sets belonging to the same user.

Yu et al. [77] exploited the uniquely combined techniques of ABE, proxy re-encryption (PRE) [60], and lazy re-encryption (LRE) [83] to allow the data owner to delegate most of the computation tasks involved in user revocation to untrusted CSPs without disclosing the underlying data contents. PRE eliminates the need of direct interaction between the data owner and the users for decryption key distribution, whereas LRE allows the CSP to aggregate the computation tasks of multiple user revocation operations. For example, once a user is revoked, the CSP just record that. If only there is a file data access request from a user, the CSP then re-encrypts the requested files and updates the requesting users secret key.

Recently, Li et al. [84] propose an expressive decentralizing KP-ABE scheme with a constant ciphertext size, i.e., the ciphertext size is independent of the number of attributes used in the scheme. In their construction, there is no trusted central authority to conduct the system setup and the access policy can be expressed as any non-monotonic access structure. In addition, their scheme is semantically secure in so-called Selective-Set model based on the n-DBDHE assumption.

Hohenberger and Waters [85] proposed a KP-ABE scheme in which ciphertexts can be decrypted with a constant number of pairings without any restriction on the number of attributes. However, the size of the user's private key is increased by a factor of number of distinct attributes in the access policy. Furthermore, there is a trusted single authority that generates private keys for users which violates the user privacy and causes a key-escrow problem.

Unfortunately, in all KP-ABE schemes, the data owners have no control over who has access to the data they encrypt, except by their choice of the set of descriptive attributes for the data. Rather, they must trust that the key-issuer issues the appropriate keys to grant or deny access to the appropriate users. Additionally, the size of the user's private key and the computation costs in encryption and decryption operations depend linearly on the number of attributes involved in the access policy.

ii. Ciphertext Policy Attribute-Based Encryption

In the ciphertext policy ABE (CP-ABE) introduced by Bethencourt et al. [86], the roles of the ciphertexts and keys are reversed in contrast with the KP-ABE scheme. The data owner determines the policy under which the data can be decrypted, while the secret key is associated with a set of attributes.

Most proposed CP-ABE schemes incur large ciphertext sizes and computation costs in the encryption and decryption operations which depend at least linearly on the number of attributes involved in the access policy. Therefore, Chen et al. [87] proposed two CP-ABE schemes, both have constant-size ciphertext and constant computation costs for an access policy containing AND-Gate with wildcard. The first scheme is provably CPA-secure in standard model under the decision n-BDHE assumption while the second scheme is provably CCA-secure in a standard model under the decision n-BDHE assumption and the existence of collision-resistant hash functions.

Zhu et al. [88, 89] proposed a comparison-based encryption scheme that support a complete comparison relation, e.g., $<$, $>$, $\leq$, $\geq$, in policy specification to implement various range-constraints on integer attributes, such as temporal and level attributes. They combined proxy re-encryption (with CP-ABE to support key delegation and reduce computational overheads on lightweight devices by outsourcing the majority of decryption operations to the CSP. Their scheme provides O(1) size of private-key and ciphertext for each range attribute. Additionally, it is also provably secure under the RSA and CDH assumption. However, their scheme depends on a central single authority to conduct the system setup and manage all attributes. And, they do not provide a mechanism for efficient user revocation.

Zhang and Chen [90] proposed the idea of "Access control as a service" for public cloud storage, where the data owner controls the authorization, and the PDP (Policy Decision Point) and PEP (Policy Enforcement Point) can be securely delegated to the CSP by utilizing CP-ABE and proxy re-encryption. However, it incurs high communication and setup costs.

The main limitations of most traditional CP-ABE schemes are: First, compromising the users privacy since the access structure is embedded in the ciphertext that may reveal the scope of data file and the authorized users who have access. The obvious solution to this problem as proposed by Nishide et al. [91], is to hide ciphertext policy, i.e., hidden access structure. Subsequently, there are various efforts to improve the traditional CP-ABE scheme and to support privacy-preserving access policy such as in [92–94]. The other limitation of the traditional CP-ABE schemes is depending on a single central authority for monitoring and issuing user's secret keys. Recently, many CP-ABE schemes consider multi-authority environments [94–96].

To achieve fine-grained and scalable data access control for personal health records (PHRs), Li et al. [95] utilized CP-ABE techniques to encrypt each patients PHR file while focusing on the multiple data owner scenario. Moreover, they adopted proxy encryption and lazy-revocation to efficiently support attribute and user revocation. However, the time costs of the key generation, encryption, and decryption processes are all linear with the number of attributes.

3.6 Hierarchical Attribute-Based Encryption

The Hierarchical Attribute-Based Encryption (HABE) model, as described in [97, 98], integrates properties in both a HIBE model [71] and an ABE model [74]. As illustrated in Fig. 8, it consists of a root master (RM) and multiple domains, where the RM functions as the TTP, and the domains are enterprise users. More precisely, a domain consists of many domain masters (DMs) corresponding to the internal trusted parties (ITPs) and numerous users corresponding to end users. The RM, whose role closely follows the root PKG in a HIBE system, is responsible for generation and distribution of system parameters and domain keys. The DM, whose role integrates both the properties of the domain PKG in a HIBE system and the AA in an ABE system, is responsible for delegating keys to the DMs at the next level and distributing secret keys to users.

Wang et al. [99] proposed fuzzy and precise identity-based encryption (FPIBE) scheme that supports the full key delegation and requires only a constant number of bilinear map operations during decryption. The FPIBE scheme is able to efficiently

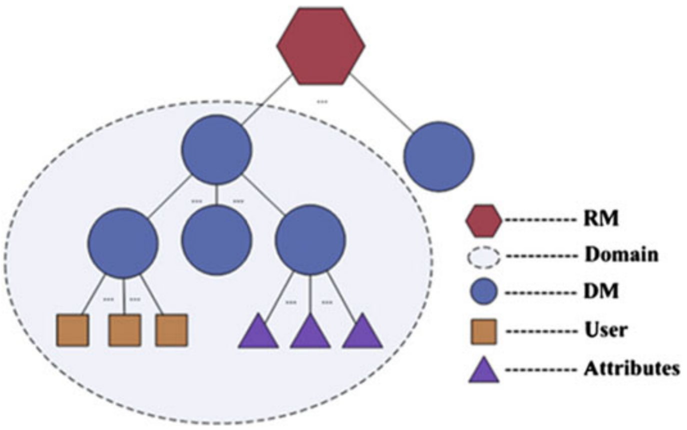

Fig. 8 Hierarchical attribute-based encryption model [98]

achieve a flexible access control by combining the HIBE system and the CP-ABE system. Using the FPIBE scheme, a user can encrypt data by specifying a recipient ID set, or an access control policy over attributes, so that only the user whose ID belongs to the ID set or attributes satisfying the access control policy can decrypt the corresponding data. However, the ciphertext length is proportional to the number of authorized users and encryption time. As well as, the size of a user's secret key is proportional to the depth of the user in the hierarchy.

For supporting compound and multi-valued attributes in a scalable, flexible, and fine-grained access control, Wan et al. [100] proposed hierarchical attribute-set-based encryption (HASBE) by extending the ciphertext policy attribute-set-based encryption (CP-ASBE) with a hierarchical structure of users. HASBE employs multiple value assignments for access expiration time to deal with user revocation more efficiently. However, the granting access operation is proportional to the number of attributes in the key structure.

Chen et al. [101] proposed a new hierarchical key assignment scheme, called CloudHKA, that addresses a cryptographic key assignment problem for enforcing a hierarchical access control policy over cloud data. CloudHKA possesses many advantages: (1) Each user only needs to store one secret key, (2) Each user can be flexibly authorized the access rights of Write or Read, or both, (3) It supports a dynamic user set and access hierarchy, and (4) It is provably-secure against collusive attacks. However, the re-key cost in the case of user revocation is linear in the number of users in the same security class. CloudHKA does not consider the expressive user attributes, so it can be considered coarse-grained access control scheme.

Wang et al. [102] recently extended the hierarchical CP-ABE scheme [97, 98] by incorporating the concept of time to perform automatic proxy re-encryption. More specifically, they proposed a time-based proxy re-encryption (TimePRE) scheme to allow a user's access right to expire automatically after a predetermined period of time. In this case, the data owner can be offline in the process of user revocations. Unfortunately, TimePRE scheme has two drawbacks: first, It assumes that there is a global time among all entities. Second, the user secret key size is O(mn), where m is the number of nodes in the time tree corresponding to the user's effective time period and n is the number of user attributes.

3.7 Role-Based Access Control

In a Role-Based Access Control (RBAC) system, access permissions are assigned to roles and roles are assigned to users/subjects [103]. Roles can be created, modified or disabled according to the system requirements. Role-permission assignments are relatively stable, while user-role assignments change quite frequently (e.g., personnel moving across departments, reassignment of duties, etc.). So, managing the user-role permissions is significantly easier than managing user rights individually [104].

Zhou et al. [105] proposed a role-based encryption (RBE) scheme for secure cloud storage. This scheme specifies a set of roles assigned to the users, each role having a set of permissions. Roles can be defined in a hierarchy, which means a role can have sub-roles (successor roles). The owner of the data can encrypt the private data to a specific role. Only the users in the specified role or predecessor roles are able to decrypt that data. The decryption key size still remains constant regardless of the number of roles that the user has been assigned to. However, the decryption cost is proportional to the number of authorized users in the same role.

4 Comparative Analysis of Security Methods on Cloud Data Storage

This section evaluates the performance of auditing and access control methods, that were presented in the previous sections, against different performance criteria.

4.1 Performance Analysis of Data Storage Auditing Methods

This section assesses the different characteristics of some existing data storage auditing schemes such as auditor type, supporting dynamic data, replication/multiple-copy and data recovery, as illustrated in Table 1. In addition, it evaluates their performance in terms of computational complexity at the CSP and the auditor, storage overheads at the CSP and the auditor, communication complexity between the auditor and the CSP, as illustrated in Table 3.

As shown in Table 1, most auditing schemes focus on a single copy of the file and provide no proof that the CSP stores multiple copies of the data owners file. Although data owners many need their critical data to be replicated on multiple servers across multiple data centers to guarantee the availability of their data. only few schemes [26, 37, 38] that support auditing for multiple replicas of data owners file.

Table 2 gives more notations for cryptographic operations used in the different auditing methods. Let r, n, k denote the number of replicas, the number of blocks per replica and the number of sectors per block (in case of block fragmentation), respectively. s denotes the block size. c denotes the number of challenged blocks. Let λ be the security parameter which is usually the size of key. Let p denote the order of the groups and $\phi(N)$ denotes the Euler Function on the RSA modulus N.

As shown in Table 3, MAC-based schemes [29] require storage overheads of metadata, i.e., block tags, as long as each data block size. On the other hand, they have efficient computational complexity at the CSP and the auditor. The homomorphic tags based on BLS signatures [19, 29] are much shorter than the RSA-based homomorphic tags [29]. While the verification cost, i.e., the computational cost at the auditor in the BLS-based auditing schemes is higher than that of RSA-based auditing schemes due to the bilinear pairing operations which consume more time than other cryptographic operations.

Table 1 Characteristics of data storage auditing schemes

Scheme	Auditor type	Dynamic data	Replication	Data recovery
Liu et al. [17]	Public	Yes	No	No
Yang and Xiaohua [18]	Public	Yes	No	No
Ateniese et al. [19]	Public	No	No	No
Wang et al. [48]	Public	Yes	No	No
Shacham and Waters [29] (BLS)	Public	No	No	Yes
Shacham and Waters [29] (RSA)	Public	No	No	Yes
Shacham and Waters [29] (MAC)	Private	No	No	Yes
Yuan and Yu [31]	Public	No	No	Yes
Xu and Chang [32]	Public	No	No	Yes
Barsoum and Hasan [37]	Public	Yes	Yes	No
Zhu et al. [38]	Public	No	Yes	No
Etemad and Kupcu [26]	Private	Yes	Yes	No

Table 2 Notation of cryptographic operations

Notation	Cryptographic operation
MUL	Multiplication in group G
ADD	Addition in group G
EXP	Exponentiation in group G
H	Hashing into group G
Pairing	Bilinear pairing; e(u, v)
SEncr	Stream Encryption
MOD	Modular operation in z_N

To reduce the storage overheads, the data owner can store the tags together with the data blocks only on the CSP. Upon challenging the CSP, the CSP generates the data proof and the tag to the auditor instead of only the data proof. But this solution will increase the communication cost between the CSP and the auditor. In fact, there is a tradeoff between the storage overheads at the auditor and the communication costs between the CSP and the auditor.

The scheme of Yuan and Yu [31] omits the tradeoff between communication costs and storage costs. The message exchanged between the CSP and the auditor during the auditing procedure consists of a constant number of group elements. However, it requires 4 Bilinear pairing operations for proof verification that result in high computational cost at the auditor. The communication cost can be reduced by using short homomorphic tags such as BLS tags [6, 24, 26] that enable the CSP to reduce the communication complexity of the auditing by aggregating the authentication tags of individual file blocks into a single tag. Batch auditing [18, 48] can further reduce the communication cost by allowing the CSP to send the linear combination of all the challenged data blocks whose size is equal to one data block instead of sending them sequentially.

Auditing schemes that support dynamic data [17, 18, 26, 37, 48] add more storage overheads at the auditor. MHT-based schemes [48] keep the metadata at the auditor side (i.e. the root of the MHT) smaller than schemes that utilize the index-table [18, 37], as the total number of index table entries is equal to the number of file blocks. On the contrary, the computational and communication costs of the MHT-based schemes are higher than those of the table-based schemes. During the dynamic operations of the MHT-based schemes, the data owner sends a modification request to the CSP and receives the authentication paths. The CSP updates the MHT according to the required dynamic operations, regenerates a new directory root and sends it to the auditor. On the other hand, for the table-based schemes, the data owner only sends a request to the CSP and updates the index table without usage of any cryptographic operations. However, auditing schemes based on the index table [18, 37] suffer a performance penalty during insertion or deletion operations, i.e. O(n) in the worst case, because the indexes of all the blocks after the insertion/deletion point are changed, and all the tags of these blocks should be recalculated. While the complexity of insertion or deletion operations when using skip lists [26] is O(log n).

4.2 Performance Analysis of Access Control Methods

This section presents an evaluation of different access control methods in terms of the ciphertext size, user's secret key size, decryption cost, user revocation cost and the existence of multiple authorities. Table 4 evaluates the performance of different access control methods.

As illustrated in Table 4, user revocation is a very challenging issue in access control methods that requires re-encryption of data files accessible to the revoked user, and may need updates of secret keys for all the non-revoked users. So the user revocation requires a heavy computation overhead on the data owner and may also require him/her to be always online. Access control methods that utilize the proxy encryption can efficiently perform the user revocation operation such as [90, 95, 96, 101]. To further improvement on the complexity of user revocation, hierarchical access control methods such as [100, 105] use the concept of key delegation. Ciphertext and secret key sizes are other challenging issues in current access control methods that may cause high storage overhead and communication cost. Ciphertext and secret key sizes usually grow linearly with the number of attributes in the system domain. Methods of [90, 92, 100, 101, 105] have constant ciphertext sizes while only the methods of [92, 105] achieve constant secret key size.

In most of the access control methods, decryption cost scales with the complexity of the access policy or the number of attributes which is infeasible as in case of

Table 3 Performance analysis of different auditing schemes

Scheme	Computational complexity		Storage overhead		Communication complexity
	CSP	Auditor	CSP	Auditor	
Liu et al. [17]	1 SEncr + O(c) [MUL + EXP + ADD]	O(c) [MUL + H + EXP] + 2 *pairing*	(1 + l/k)n	O(1)	O(1)
Yang and Xiaohua [18]	O(c) [ADD + MUL + EXP] + O(k) [EXP + *pairing*]	O(c) [H + EXP + MUL] + 2 *pairing*	s · n/k	O(1)	O(c)
Ateniese et al. [19]	O(c) [H + MUL + EXP]	O(c) [H + MUL + EXP]	n.|N|	O(λ)	3|N|
Wang et al. [48]	O(c) [MUL + EXP + ADD] + H + 1 *pairing*	O(c) [MUL + H + EXP] + 2 *pairing*	(1 + l/k)n	O(1 + 1/k)	O(kc)
Shacham and Waters [29] **(BLS)**	O(c) [ADD + MUL + EXP]	O(k) [MUL + EXP] + O(c) [H + EXP] + 2 *pairing*	n – |p|	O(λ)	O(c) + |p|
Shacham and Waters [29] **(RSA)**	O(c) [ADD + MUL + EXP]	O(k) [MUL + EXP] + O(c) [H + EXP]	n – |N|	O(λ)	O(c) + |N| + s
Shacham and Waters [29] **(MAC)**	O(c)	O(c)H	n – |p|	N/A	O(c) + |p| + s
Yuan and Yu [31]	O(c + s) [MUL + EXP)	O(c) [MUL + EXP] + 4 *pairing*	(1 + l/s)n	O(1)	O(1)
Xu and Chang [32]	O(s) EXP	3 EXP + O(c) [ADD + MUL + PRF]	(1 + l/s)n	O(1)	O(λ)
Barsoum and Hasan [37]	O(c) [EXP + MUL + ADD]	O(c) [H + MUL + EXP] + O(k) [MUL + EXP] + O(r)ADD + 2 *pairing*	2n	O(2n)	O(kr)
Zhu et al. [38]	O(c) *pairing* + O(ck)EXP	3 *pairing* + O(k) EXP	n – |p|	O(λ)	O(c) + O(k)
Etemad and Kupcu [26]	O(1 + log nr) [EXP + MUL]	O(1 + log nr) [EXP + MUL]	log nr	O(1)	O(log nr)

Table 4 Performance analysis of different access control methods

Access control method	Approach	Ciphertext size	User key size	Decryption cost	User revocation cost	Multiple authority
Hohenberger and Waters [85]	KP-ABE	Linear with no. of attributes	Linear with no. of attributes	Constant no. of pairings	N/A	No
Doshi and Jinwala [92]	CP-ABE	Constant	Constant	Constant no. of pairings	N/A	No
Qian et al. [93]	CP-ABE fully hidden access structure	Linear with no. of attributes	Linear with total no. of attributes	Linear with no. of attribute authorities	N/A	Yes
Li et al. [95]	CP-ABE with proxy encryption and lazy re-encryption	Linear with no. of attributes and no. of AA	Linear with no. of attributes in the secret key	Linear with total no. of attributes	Linear with no. of attributes in the secret key	Yes
Yang et al. [96]	CP-ABE with proxy encryption	Linear with no. of attributes in the policy	Linear with the total no. of attributes	Constant	Linear with no. of non-revoked users who hold the revoked attribute	Yes
Wang et al. [99]	HIBE + CP-ABE	Linear with no. of users and the max. depth of hierarchy	Linear with no. of attributes of a user	Constant	N/A	Yes
Wan et al. [100]	CP-ABE + HIBE	Constant	Linear with no. of attributes of a user	Linear with no. of attributes in the key	Constant	Yes
Chen et al. [101]	HIBE with proxy encryption	Constant	Read and write keys independent of number of ciphertexts	Linear with the depth of the user in hierarchy	Linear with no. of authorities and no. of ciphertext accessed by revoked user	Yes
Zhou et al. [105]	Hierarchical RBAC	Constant	Constant	Linear with no. of users in the same role	Linear with no. of roles	Yes
Zhang and Chen [90]	CP-ABE + proxy encryption	Constant	Linear with no. of attributes	Constant no. of pairings	Constant	No

lightweight devices such as mobiles. Some access control methods such as [85, 90, 92, 96, 99] have constant decryption costs. Some access control methods, e.g. [85, 90, 92], assume the use of a single trusted authority (TA) that manages (e.g., add, issue, revoke, etc.,) all attributes in the system domain. This assumption not only may create a load bottleneck, but also suffers from the key escrow problem as the TA can decrypt all the files, causing a privacy disclosure. Instead, recent access control methods, such as [99–101], consider the existence of different entities, called attribute authorities (AAs), responsible for managing different attributes of a person, e.g. the Department of Motor Vehicle Tests whether you can drive, or a university can certify that you are a student, etc. Each AA manages a disjoint subset of attributes, while none of them alone is able to control the security of the whole system.

5 Discussions and Concluding Remarks

From the extensive survey in the previous sections, we conclude that data security in cloud is one of the major issues that is considered as a barrier in the adoption of cloud storage services. The most critical security concern is about data integrity, availability, privacy and confidentiality.

For validating data integrity and availability in cloud computing, many auditing schemes have been proposed under different security levels and cryptographic assumptions. Most auditing schemes are provably secure in the random oracle model. The main limitations of existing auditing schemes can be summarized as follows: (1) Dealing only with archival static data files and do not consider dynamic operations such as insert, delete and update, (2) Relying on spot checking that can detect if a long fraction of the data stored at the CSP has been corrupted but it cannot detect corruption of small parts of the data, (3) Relying on Reed-Solomon encoding scheme to support data recovery feature in case the data is lost or corrupted. It leads to inefficient encoding and decoding for large files, (4) Supporting only private auditing that impose computational and online burdens on the data owners for periodically auditing their data, (5) verifying only single copy data files and not considering replicated data files, and (6) Incurring high computational costs and storage overheads.

Therefore, to overcome the prior limitations, an ideal auditing scheme should have the following features:

1. **Public auditing**: To enable the data owners to delegate the auditing process to a TPA in order to verify the correctness of the outsourced data on demand.
2. **Privacy-preserving assurance**: To prevent the leakage of the verified data during the auditing process.
3. **Data dynamics**: To efficiently allow the clients to perform block-level operations on the data files such as: insertion, deletion and modification while maintaining the same level of data correctness assuring and guaranteeing data freshness.
4. **Robustness**: To efficiently recover an arbitrary amount of data corruptions.
5. **Availability and reliability**: To support auditing for distinguishable multi-replica data files to make sure that the CSP is storing all the data replicas agreed upon.

6. **Blockless verification**: To allow TPA to verify the correctness of the cloud data on demand without possessing or retrieving a copy of challenged data blocks.
7. **Stateless verification**: Where the TPA is not needed to maintain a state between audits.
8. **Efficiency**: To achieve the following aspects:
   (a) Minimum computation complexity at the CSP and the TPA.
   (b) Minimum communication complexity between the CSP and the TPA.
   (c) Minimum storage overheads at the CSP and the TPA.

For ensuring data confidentiality in the cloud computing, various access control methods were proposed in order to restrict access to data and guarantee that the outsourced data is safe from unauthorized access. However, these methods suffer many penalties such as: (1) Heavy computation overheads and a cumbersome online burden towards the data owner because of the operation of user revocation, (2) Large ciphertext and secret key sizes as well as high computation costs in the encryption and decryption operations which depend at least linearly on the number of attributes involved in the access policy, (3) Compromising the users privacy by revealing some information about the data file and the authorized users in the access policy, and (4) Relying on single authority for managing different attributes in the access policy which may cause a bottleneck and a key escrow problem.

Consequently, we propose that the ideal access control methods should satisfy the following requirements:

1. **Fine-grained**: Granting different access rights to a set of users and allowing flexibility and expressibility in specifying the access rights of individual users.
2. **Privacy-preserving**: Access policy does not reveal any information to the CSP about the scope of data file and the kind or the attributes of users authorized to access.
3. **Scalability**: the number of authorized users cannot affect the performance of the system.
4. **Efficiency**: The method should be efficient in terms of: ciphertext size, user's secret key size, and the costs of encryption, decryption, and user revocation.
5. **Forward and backward security**: The revoked user should not be able to decrypt new ciphertexts encrypted with the new public key. And, the newly joined users should be able to decrypt the previously published ciphertexts encrypted with the previous public key if they satisfy the access policy.
6. **Collusion resistant**: Different users cannot collude each other and combine their attributes to decrypt the encrypted data.
7. **Multiple authority**: To overcome the problems of load bottleneck and key escrow problems, there should be multiple authorities that manage the user attributes and issue the secret keys, rather than a central trusted authority.

6 Conclusion

Cloud storage services have become extremely promising due to its cost effectiveness and reliability. However, this service also brings many new challenges for data security and privacy. Therefore, cloud service providers have to adopt security practices to ensure that the clients' data is safe. In this paper, the different security challenges within a cloud storage service are introduced and a survey of the different methods to assure data integrity, availability and confidentiality is presented. Comparative evaluations of these methods are also conducted according to various pre-defined performance criteria. Finally, we suggest the following research directions for data security within cloud computing environments: (1) Integrate attribute-based encryption with role-based access control models such that the user-role and role-permission assignments can be separately constructed using access policies applied on the attributes of users, roles, the objects and the environment. (2) Develop a context-aware role-based control model and incorporate it to the Policy Enforcement Point of a cloud, and enable/activate the role only when the user is located within the logical positions time intervals and under certain platforms in order to prevent the malicious insiders from disclosing the authorized user's identity. (3) Integrate efficient access control and auditing methods with new hardware architectural and virtualization features that can help protect the confidentiality and integrity of the data and resources. (4) Incorporate the relationship between auditing and access control for guaranteeing secure cloud storage services. (5) Extend current auditing methods with data recovery features.

References

1. Attebury, R., George, J., Judd, C., Marcum, B.: Google docs: a review. Against Grain **20**(2), 14–17 (2008)
2. Tim, M., Subra, K., Shahed, L.: Cloud Security and Privacy. O'Reilly and Associates, USA (2009)
3. Chambers, J.: Windows Azure Web Sites. Wiley (2013)
4. Pandey, U.S., Anjali, J.: Google app engine and performance of the web application. Int. J. **2**(2) (2013)
5. Gonzalez, C., Border, C., Oh, T.: Teaching in amazon EC2. In: The 13th Annual ACM SIGITE Conference on Information Technology Education. ACM (2013)
6. Srinivasan, S.: Cloud computing providers. In: Cloud Computing Basics. Springer, New York (2014)
7. Bhadauria, R., Sanyal, S.: Survey on security issues in cloud computing and associated mitigation techniques. Int. J. Comput. Appl. **47**(18), 47–66 (2012)
8. Borgmann, M., Hahn, T., Herfert, M., Kunz, T., Richter M., Viebeg, U., Vowe, S.: On the Security of Cloud Storage Services. Fraunhofer-Verlag (2012)
9. Berriman, G.B., Deelman, E., Good, J., Juve, G., Kinney, J., Merrihew, A., Rynge, M.: Creating A Galactic Plane Atlas With Amazon Web Services (2013). arXiv:1312.6723
10. Garg, S.K., Versteeg, S., Buyya, R.: A framework for ranking of cloud computing services. Future Gener. Comput. Syst. **29**(4), 1012–1023 (2013)

11. Miller, R.: Amazon Addresses EC2 Power Outages. Data Center Knowledge (2010). http://www.datacenterknowledge.com/archives/2010/05/10/amazon-addresses-ec2-power-outages/
12. Aboalian, A., Badr, N.L., Tolba, M.F.: Keystroke dynamics based user authentication service for cloud computing. In: Practice and Experience: Concurrency and Computation (2015)
13. Cong, W., Ren, K., Lou, W., Li, J.: Toward publicly auditable secure cloud data storage services. IEEE Netw. **24**(4), 19–24 (2010)
14. Shalabi, S.M., Doll, C.L., Reilly, J.D., Shore, M.: Access Control List. U.S. Patent Application 13/311, 278 (2011)
15. Abo-alian, A., Badr, N.L., Tolba, M.F.: Hierarchical attribute-role based access control for cloud computing. In: The 1st International Conference on Advanced Intelligent System and Informatics (AISI2015). Springer (2016)
16. Blum, M., Evans, W., Gemmell, P., Kannan, S., Naor, M.: Checking the correctness of memories. In: The 32nd Annual Symposium on Foundations of Computer Science. IEEE Computer Society, Washington, DC, USA (1991)
17. Liu, H., Zhang, P., Lun, J.: Public data integrity verification for secure cloud storage. J. Netw. **8**(2), 373–380 (2013)
18. Yang, K., Xiaohua, J.: TSAS: third-party storage auditing service. In: Security for Cloud Storage Systems. Springer Briefs in Computer Science (2014)
19. Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.X.: Provable data possession at untrusted stores. In: The 2007 ACM Conference on Computer and Communications Security. ACM (2007)
20. Juels, A., Kaliski, B.S.: Pors: proofs of retrievability for large files. In: The 2007 ACM Conference on Computer and Communications Security. ACM (2007)
21. Zheng, Q., Xu, S.: Secure and efficient proof of storage with deduplication. In: The Second ACM Conference on Data and Application Security and Privacy. ACM (2012)
22. Yang, K., Jia, X.: Data storage auditing service in cloud computing: challenges, methods and opportunities. World Wide Web **15**(4), 409–428 (2012)
23. Chen, B., Curtmola, R.: Robust dynamic provable data possession. In: The 32nd International IEEE Conference on Distributed Computing Systems Workshops. IEEE (2012)
24. Mukundan, R., Madria, S., Linderman, M.: Replicated data integrity verification in cloud. IEEE Data Eng. Bull. **35**(4), 55–64 (2012)
25. Chen, B., Curtmola, R.: Towards self-repairing replication-based storage systems using untrusted clouds. In: The 3rd ACM Conference on Data and Application Security and Privacy (CODASPY '13). ACM (2013)
26. Etemad, M., Kupcu, A.: Transparent distributed and replicated dynamic provable data possession. In: The 11th International Conference on Applied Cryptography and Network. Springer, Berlin (2013)
27. Zhu, Y., Ahn, G., Hu, H., Yau, S.S., An, H.G., Hu, C.: Dynamic audit services for outsourced storages in clouds. IEEE Trans. Serv. Comput. **6**(2), 227–238 (2013)
28. Abo-alian, A., Badr, N.L., Tolba, M.F.: Auditing-as-a-service for cloud storage. In: Intelligent Systems' 2014. Springer (2015)
29. Shacham, H., Waters, B.: Compact proofs of retrievability. J. Cryptol. **26**(3), 442–483 (2013)
30. Plank, J.S.: A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. Softw. Pract. Exp. **27**(9), 995–1012 (1997)
31. Yuan, J., Yu, S.: Proof of retrievability with public verifiability and constant communication cost in cloud. In: The 2013 International ACM Workshop on Security in Cloud Computing. ACM (2013)
32. Xu, J., Chang, E.C.: Towards efficient provable data possession. In: IACR Cryptology ePrint Archive 574. ASIACCS (2011)
33. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Khan, O., Kissner, L., Peterson, Z., Song, D.: Remote data checking using provable data possession. ACM Trans. Inf. Syst. Secur. **14**(1), 121–155 (2011)

34. Cao, N., Yu, S., Yang, Z., Lou, W., Hou, Y.T.: LT codes-based secure and reliable cloud storage service. In: The 2012 INFOCOM. IEEE (2012)
35. Rashmi, K.V., Shah, N.B., Kumar, P.V., Ramchandran, K.: Exact regenerating codes for distributed storage. In: Allerton Conference on Control Computing and Communication (2009)
36. Barsoum, A.F., Hasan, M.A.: On verifying dynamic multiple data copies over cloud servers. IACR Cryptol. ePrint Arch. **447** (2011)
37. Barsoum, A.F., Hasan, M.A.: Integrity verification of multiple data copies over untrusted cloud servers. In: The 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (2012)
38. Zhu, Y., Hu, H., Ahn, G.J., Yu, M.: Cooperative provable data possession for integrity verification in multicloud storage. IEEE Trans. Parallel Distrib. Syst. **23**(12), 2231–2244 (2012)
39. Wang, H., Zhang, Y.: On the knowledge soundness of a cooperative provable data possession scheme in multicloud storage. IEEE Trans. Parallel Distrib. Syst. **25**(1), 264–267 (2014)
40. Merkle, R.C.: Protocols for public key cryptosystems. In: IEEE Symposium on Security and Privacy. IEEE Computer Society (1980)
41. Zhang, Y., Blanton, M.: Efficient dynamic provable possession of remote data via balanced update trees. In: The 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (2013)
42. Pugh, W.: Skip lists: a probabilistic alternative to balanced trees. Commun. ACM **33**(6), 668–676 (1990)
43. Goodrich, M.T., Tamassia, R., Schwerin, A.: Implementation of an authenticated dictionary with skip lists and commutative hashing. In: DARPA Information Survivability Conference (2001)
44. Erway, C., Kp, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. In: The 16th ACM Conference on Computer and Communications Security. ACM (2009)
45. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Advances in Cryptology CRYPTO99. Springer, Heidelberg (1999)
46. Wang, Q., Wang, C., Ren, K., Lou, W., Li, J.: Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parallel Distrib. Syst. **22**(5), 847–859 (2011)
47. Liu, F., Gu, D., Lu, H.: An improved dynamic provable data possession model. In: The IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS). IEEE (2011)
48. Wang, C., Chow, S.S., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for secure cloud storage. IEEE Trans. Comput. **62**(2), 362–375 (2013)
49. Ateniese, G., Kamara, S., Katz, J.: Proofs of Storage from homomorphic identification protocols. In: The 15th International Conference on Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT). Springer, Heidelberg (2009)
50. Li, C., Chen, Y., Tan, P., Yang, G.: An efficient provable data possession scheme with data dynamics. In: Tthe International Conference on Computer Science and Service System (CSSS). IEEE (2012)
51. Li, C., Chen, Y., Tan, P., Yang, G.: Towards comprehensive provable data possession in cloud computing. Wuhan Univ. J. Nat. Sci. **18**(3), 265–271 (2013)
52. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996)
53. Li, N.: Discretionary access control. In: Encyclopedia of Cryptography and Security. Springer (2011)
54. Lindqvist, H.: Mandatory access control. Master's Thesis in Computing Science, Umea University, Department of Computing Science (2006)
55. Ferraiolo, D., Kuhn, D.R., Chandramouli, R.: Role-Based Access Control. Artech House (2003)
56. Cha, B., Seo, J., Kim, J.: Design of attribute-based access control in cloud computing environment. In: The International Conference on IT Convergence and Security. Springer, Netherlands (2012)

57. Yu, S.: Data sharing on untrusted storage with attribute-based encryption. PhD diss, Worcester Polytechnic Institute (2010)
58. Kallahalla, M., Riedel, E., Swaminathan, R., Wang, Q., Fu, K.: Plutus: scalable secure file sharing on untrusted storage. In: FAST03 Berkeley, California, USA (2003)
59. Vimercati, S.D.C. di, Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Over-encryption: management of access control evolution on outsourced data. In: The 33rd International Conference on Very Large Data Bases, VLDB Endowment (2007)
60. Goh, E., Shacham, H., Modadugu, N., Boneh, D.: Sirius: securing remote untrusted storage. In: NDSS 03, San Diego, CA, USA (2003)
61. Fiat, A., Naor, M.: Broadcast encryption. In: CRYPTO 93 (Lecture Notes in Computer Science), Santa Barbara, CA, USA (1993)
62. Halevy, D., Shamir, A.: The LSD broadcast encryption scheme. In: CRYPTO 02 (Lecture Notes in Computer Science), Santa Barbara, CA, USA (2002)
63. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Proceedings of CRYPTO 05 (Lecture Notes in Computer Science), Santa Barbara, CA, USA (2005)
64. Delerable, C., Paillier, P., Pointcheval, D.: Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In: Pairing-Based Cryptography Pairing 2007. Springer, Heidelberg (2007)
65. Kim, J., Susilo, W., Au, M.H., Seberry, J.: Efficient semi-static secure broadcast encryption scheme. In: Pairing-Based Cryptography Pairing 2013. Springer (2014)
66. Gentry, C., Waters, B.: Adaptive security in broadcast encryption systems (with short ciphertexts). In: Advances in Cryptology-EUROCRYPT 2009. Springer, Heidelberg (2009)
67. Wikipedia: ID-based encryption (2014). http://en.wikipedia.org/wiki/ID-based_encryption
68. Li, J., Chen, X., Jia, C., Lou, W.: Identity-based encryption with outsourced revocation in cloud computing. IEEE Trans. Comput. 1–12 (2013)
69. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: EUROCRYPT 02 (Lecture Notes in Computer Science), Amsterdam, The Netherlands (2002)
70. Gentry, C., Halevi, S.: Hierarchical identity based encryption with polynomially many levels. In: TCC 09 (Lecture Notes in Computer Science), San Francisco, CA, USA (2009)
71. Gagn, M.: Identity-based encryption. In: Encyclopedia of Cryptography and Security. Springer Science Business Media, LLC (2011)
72. Liu, Q., Wang, G., Wu, J.: Efficient sharing of secure cloud storage services. In: IEEE TSP 10 in Conjunction with IEEE CIT 10, Bradford, UK (2010)
73. Mao, Y., Zhang, X., Chen, M., Zhan, Y.: Constant size hierarchical identity-based encryption tightly secure in the full model without random oracles. In: The 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies (EIDWT). IEEE (2013)
74. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: EUROCRYPT 05 (Lecture Notes in Computer Science), Aarhus, Denmark (2005)
75. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCS 06, Alexandria, VA, USA (2006)
76. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Public Key Cryptography|PKC, LNCS. Springer (2011)
77. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: The 2010 IEEE INFOCOM. IEEE (2010)
78. Si, X., Wang, P., Zhang, L.: KP-ABE based verifiable cloud access control scheme. In: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE (2013)
79. Moses, T.: Extensible access control markup language (xacml) version 2.0. Oasis Standard 200502 (2005)
80. Chase, M.: Multi-authority attribute based encryption. In: TCC 07 (Lecture Notes in Computer Science), Amsterdam, The Netherlands (2007)
81. Li, M., Yu, S., Ren, K., Lou, W.: Securing personal health records in cloud computing: patient-centric and fine-grained data access control in multi-owner settings. In: Proceedings of Security and Privacy in Communication Networks. Springer, Heidelberg (2010)

82. Chase, M., Chow, S.: Improving privacy and security in multi-authority attribute-based encryption. In: ACM CCS 09, Chicago, IL, USA (2009)
83. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: EUROCRYPT 98, Espoo, Finland (1998)
84. Li, Q., Xiong, H., Zhang, F., Zeng, S.: An expressive decentralizing kp-abe scheme with constant-size ciphertext. Int. J. Netw. Secur. **15**(3), 161–170 (2013)
85. Hohenberger, S., Waters, B.: Attribute-based encryption with fast decryption. In: Public-Key Cryptography PKC 2013. Springer, Heidelberg (2013)
86. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy. IEEE Computer Society (2007)
87. Chen, C., Zhang, Z., Feng, D.: Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In: Proceedings of Provable Security. Springer, Heidelberg (2011)
88. Zhu, Y., Hu, H., Ahn, G., Huang, D., Wang, S.: Towards temporal access control in cloud computing. In: The 2012 IEEE INFOCOM. IEEE (2012)
89. Zhu, Y., Hu, H., Ahn, G., Yu, M., Zhao, H.: Comparison-based encryption for fine-grained access control in clouds. In: The Second ACM Conference on Data and Application Security and Privacy. ACM (2012)
90. Zhang, Y., Chen, J.: Access control as a service for public cloud storage. In: Distributed Computing Systems Workshops (ICDCSW). IEEE (2012)
91. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden encryptor-specified access structures. In: Applied Cryptography and Network Security. Springer, Heidelberg (2008)
92. Doshi, N., Jinwala, D.: Hidden access structure ciphertext policy attribute based encryption with constant length ciphertext. In: Advanced Computing, Networking and Security. Springer, Heidelberg (2012)
93. Qian, H., Li, J., Zhang, Y.: Privacy-preserving decentralized ciphertext-policy attribute-based encryption with fully hidden access structure. In: Information and Communications Security. Springer (2013)
94. Jung, T., Li, X., Wan, Z., Wan, M.: Privacy preserving cloud data access with multi-authorities. In: The 2013 IEEE INFOCOM. IEEE (2013)
95. Li, M., Yu, S., Zheng, Y., Ren, K., Lou, W.: Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. IEEE Trans. Parallel Distrib. Syst. **24**(1), 131–143 (2013)
96. Yang, K., Jia, X., Ren, K., Zhang, B.: Dac-macs: effective data access control for multi-authority cloud storage systems. In: The 2013 IEEE INFOCOM. IEEE (2013)
97. Wang, G., Liu, Q., Wu, J.: Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In: The 17th ACM Conference on Computer and Communications Security. ACM (2010)
98. Wang, G., Liu, Q., Wu, J., Guo, M.: Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. Comput. Secur. **30**(5), 320–331 (2011)
99. Wang, G., Liu, Q., Wu, J.: Achieving finegrained access control for secure data sharing on cloud servers. Concurr. Comput. Pract. Exp. **23**(12), 1443–1464 (2011)
100. Wan, Z., Liu, J., Deng, R.H.: HASBE: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing. IEEE Trans. Inf. Forensics Secur. **7**(2), 743–754 (2012)
101. Chen, Y., Chu, C., Tzeng, W., Zhou, J.: Cloudhka: A cryptographic approach for hierarchical access control in cloud computing. In: Applied Cryptography and Network Security. Springer, Heidelberg (2013)
102. Wang, G., Liu, Q., Wu, J.: Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. Inf. Sci. **258**, 355–370 (2014)
103. Wikipedia: Role-based access control (2014). http://en.wikipedia.org/wiki/Role-based_access_control

104. Ferrara, A.L., Madhusudan, P., Parlato, G.: Policy analysis for self-administrated role-based access control. In: Tools and Algorithms for the Construction and Analysis of Systems. Springer, Heidelberg (2013)
105. Zhou, L., Varadharajan, V., Hitchens, M.: Enforcing role-based access control for secure data storage in the cloud. Comput. J. **54**(10), 1675–1687 (2011)