Cloud Data Auditing Techniques with a Focus on Privacy and Security

Manjur Kolhar | Prince Sattam Bin Abdulaziz University
Mosleh M. Abu-Alhaj | Al-Ahliyya Amman University
Saied M. Abd El-atty | Menoufia University

An analysis of the state of the art and research in cloud data auditing techniques highlights integrity and privacy challenges, current solutions, and future research directions.

Storing large amounts of data with cloud service providers (CSPs) raises concerns about data protection. Data integrity and privacy can be lost because of the physical movement of data from one place to another by the cloud administrator, malware, dishonest cloud providers, or other malicious users who might distort the data.[1] Hence, saved data corrections must be verified at regular intervals.

Nowadays, with the help of cryptography, verification of remote (cloud) data is performed by third-party auditors (TPAs).[2] TPAs are also appropriate for public auditing, offering auditing services with more powerful computational and communication abilities than regular users.[3] In public auditing, a TPA is designated to check the correctness of cloud data without retrieving the entire dataset from the CSP. However, most auditing schemes don't protect user data from TPAs; hence, the integrity and privacy of user data are lost.[1] Our research focuses on cryptographic algorithms for cloud data auditing and the integrity and privacy issues that these algorithms face. Many approaches have been proposed in the literature to protect integrity and privacy; they're generally classified according to data's various states: static, dynamic, multiowner, multiuser, and so on.

We provide a systematic guide to the current literature regarding comprehensive methodologies. We not only identify and categorize the different approaches to cloud data integrity and privacy but also compare and analyze their relative merits. For example, our research lists the strengths and weaknesses of earlier work on cloud auditing, which will enable researchers to design new methods. Although related topics such as providing security to the cloud are beyond this article's scope, cloud data auditing requires explicit attention, which we provide below.

Background and Overview

Figure 1 depicts a cloud data auditing process that employs a TPA to achieve data integrity and privacy.

Initially, data owners convey concerns to the auditor about their data's status. The query is then forwarded to the cloud server; during this communication, the auditor and the CSP exchange the cryptographic keys and the data to be audited. As Figure 1 illustrates, the TPA is considered a centralized solution for auditing, even though it consumes critical resources such as memory and communication channels. In addition, the TPA can access the outsourced data by executing challenge algorithms for verification purposes.[1–3]

Notation and Preliminaries

Cryptographic methods are built using message authentication,[4] homomorphic linear authenticators,[1,5] and Boneh–Lynn–Shacham (BLS)-based homomorphic methods.[6] Cryptographic systems use these algorithms to establish cryptographic groups and construct security-based primitives. The cryptographic primitives specific to auditing include key generators, tag generators, challenge generators, and proof verifiers, and are called *cryptographic operations*. These interlinked algorithms are the building blocks of the auditing process. However, during the auditing process, the TPA can put the data at risk.[1] To maintain integrity and privacy, TPAs, cloud users, and CSPs employ only these algorithms; however, the approach varies with the service types provided to the cloud for auditing purposes. Basically, algorithms such as key, tag, or sign generators are executed on the client side, whereas the file to be verified (challenged) and the file's proof generation are performed on the cloud server and the auditor side, respectively, along with security parameters and secret and public keys.

A frequently used parameter in cryptography is security parameter n, which defines the public and private keys' length. This security parameter should be computationally feasible, and cryptosystem execution should be polynomial in time. If we increase the keys' size, then the time to decrypt and encrypt them will increase, and it becomes much harder for adversaries to break them in polynomial time, usually represented as 1^n. To generate keys and perform other cryptographic operations for auditing cloud storage service reliability, a security parameter representing the problem's input size—commonly 80, 128, or 160 bits—is used.[7] File F, the outsourced data, is represented as a sequence of finite sets of n blocks of memory, for example, $m_1, m_2, m_3, \ldots m_n$. These data blocks shouldn't be bigger than the security parameter because private data owners need to encrypt the data with the corresponding key.[8]

Using pairing-based tools is one cryptographic method. For example, let G_1 and G_2 be two groups with the same prime order q, where G_1 is an additive group and G_2 is a multiplicative group. A bilinear mapping can be expressed as follows: e: $G_1 \times G_2 \rightarrow GT$ with the following properties:

- bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$, for all $P \in G_1$, $Q \in G_2$ and $a, b \in Zq$, Zq is a prime order;
- nondegeneracy: If P is a generator of G_1, then e (P, P) is a generator of G_2. Hence, e (P, P) $\neq 1$; and
- e is efficiently computable.

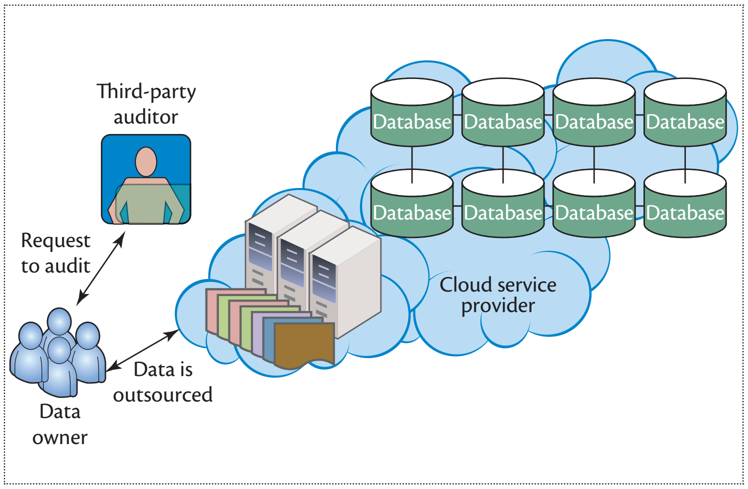

Figure 1. Auditing cloud data using a third-party auditor (TPA). A cloud user outsources data to the cloud service provider, who has resources and capabilities to provide data storage services. The TPA has the skills to audit the storage services at the cloud user's request.

Message Authentication Codes

A message authentication code (MAC) maintains the message integrity, validates the originator's identity, and provides nonrepudiation of the origin. MAC codes are generated by hash functions, which contain a hash value and the message to be authenticated. The receiver uses a security key—known to both the receiver and transmitter—to generate the message. Although a MAC usually preserves the message's integrity, the data's privacy is lost. It's also been reported that intruders can change the message[6] or share it with others, thus putting the data's integrity at risk.

It's very simple to use MACs in cloud auditing: the cloud user uploads the data block along with the MAC and sends the corresponding secret keys to the server (auditor). However, this approach requires releasing data blocks to the TPA.[1] To prevent the TPA from performing verification, the end user will be allowed to verify the data. This condition is counter to the public auditing process. The following algorithm can be used in cloud auditing:[1]

- key generator (k): $k \xleftarrow{s} k$
- code generator: $Tag \xleftarrow{s} MAC_k(M)$
- verifier: $D \leftarrow VF_k(M, Tag)$ where $D \in \{0,1\}$.

However, the key-generation algorithm is stateless and deterministic; by using a MAC, we don't need an explicit tag-verification algorithm because the receiver computes the tag using $MAC_k(M)$. If this computed tag is identical to the received tag, then the message is verified; otherwise, the receiver is unauthenticated. On the other hand, using MAC-based solutions for auditing purposes can cause serious problems, including the following:[3]

- If the cloud data is updated, then the secret keys must be regenerated and sent to the auditing entities.
- It supports only static cloud data.
- The TPA must maintain the MAC keys because the cloud data can be updated by any geographically distributed client.

Homomorphic Authentication

Homomorphic authentication (HA) lets users store the data tag α in a remote place such as a CSP; the tag is constructed from data blocks $\{m_i\}$, where $i = 1, 2, 3 \ldots n$, with a secret key. Later, the CSP uses publically available methods to compute the data blocks $\{m_i\}$ with a corresponding succinct data tag. In other words, HA allows anyone to certify the result of the complex computation performed on the authenticated datasets with data tag α. This scheme also lets users stream together bits of data from different files without exposing the data points to one another. Consider the example of supply chain management for production, sales, and retail. These transactions can be carried out without exposing each department's data.

Two types of HA are available: partially homomorphic encryption and fully homomorphic encryption. Partially homomorphic encryption can be either a multiplicative or an additive homomorphism but not both. Fully homomorphic encryption can be both additive and multiplicative.[9]

Homomorphic linear authenticators were first introduced by Giuseppe Ateniese and his colleagues, who proposed a framework that lets clients verify remotely stored files.[7] The steps for providing proof of storage are summarized as follows:

- File *f* is treated as an *N*-dimensional vector.
- Tag *t* is created for each block of file *f*.
- The client sends a random challenge vector *c*.
- The server returns proof of authentication: $\mu = \sum_i c_i, f_i$.

Homomorphic verifiable tags have been used for cloud-auditing processes. They have the properties of malleability and blockless verification. Blockless verification lets the server verify the data intact, without possessing the data and metadata for the block. For each data block or file, corresponding tags are generated, uniquely numbered, and saved as global counters. Then, the server can form a proof that lets the client verify the data by simply adding a linear combination of tag values.[10] HAs do three things: aggregate signatures, wherein *n* signatures of *n* users will have *n* messages;[11] provide a homomorphic signature;[12] and perform batch verification.[13]

HA employs four algorithms: Gen, Encode, Prove, and Vrfy.[7]

The Gen (generate) algorithm is executed by the client to set up the initial phase of tokens for proof of storage. This algorithm takes a security parameter as input, and outputs public key *pk* and private key *sk*:

$$(pk, sk) \leftarrow (Gen).$$

Encode takes two parameters as input—for example, a secret key and file *f*—and outputs encoded file f' and state information *st*:

$$(f', st) \leftarrow Encode(f).$$

Prove sends the challenge *c* along with the public key and encoded file f'; it outputs the proof:

$$\mu = Prove(pk, f', c).$$

The Vrfy (verify) algorithm verifies the algorithm by producing a 1 or 0:

$$a = Vrfy(pk, st, c, \mu).$$

With these algorithms, we conclude that the secret key isn't needed during verification. Furthermore, linear combinations of data blocks reveal adequate information for the TPA to retrieve the entire file *f*.[3,13]

The state information resulting from encoding is nothing but a security parameter; it belongs to $\{0, 1\}^K$. Ateniese and his colleagues consider file *f* to be *n* dimensional vectors. Each vector is tagged and can be identified with the help of the state information generated during encoding.[7]

HA schemes can be extended to form ring-based signatures[1] and random-masking techniques,[2] which are specially adapted for public auditability. HA-based ring signatures are used for cloud data that's shared among multiple users and are intended to provide privacy as well as blockless verification. Random masking enables the auditing process to preserve cloud data's privacy during auditing.

Boneh–Lynn–Shacham

The BLS system uses bilinear pairing for verification, and signatures are grouped under an elliptic curve. It's an undeniable signature scheme that helps users verify that a signer is trusted. Furthermore, it can work with any scheme in gap Diffie–Hellman (GDH) group *G*.[14] The arrangement requires a hash function derived from the message space on *G*.

This scheme can also be utilized in cloud-auditing techniques. Let $G = <g>$ of the GDH group of prime order p, with hash function H: $\{0, 1\}^* \rightarrow G$, be considered as a random oracle. Any block of data can be encrypted using the following algorithm:[14]

- Key generation will be executed by the cloud client. Select a random variable, $x \xleftarrow{R} Zp$ and compute $v \leftarrow g^x$. The public and private keys are $v \in G$ and $x \in Zp$, respectively.
- Signing uses a private key and message $M \in \{0, 1\}^*$, determined as $h \leftarrow H(M)$, where hash value $h \in G$ and $\sigma \leftarrow h^x$.
- Verification computes $h \leftarrow H(M)$ from a public key x, a block of data, and a signature. Hence, (g, v, h, σ) is verified as a valid tuple.

These schemes are further extended to support dynamic data as well as public auditability. The Merkle hash tree is one scheme that achieves these goals. Based on the binary tree concept, its leaves are hashes of authenticated data values, as Qian Wang and his colleagues discuss.[15] In the sense that it audits dynamic data, this work extends that of Ateniese and his colleagues[10] and Ari Juels and Burton Kaliski,[16] who considered signatures with respective file indexes. Hence, once a file is updated, its previous file indexes should also be recomputed. To reduce the overhead of keeping an index of files, Qian Wang and his colleagues discarded the index information for files and created tags for each data block to support data dynamics.[15]

Current Cloud Data Storage Frameworks

Cloud computing's astonishing growth has inherently led to tremendous amounts of user data being part of these CSPs. Storing all of a user's data in a single location escalates the threat to the data. Hence, most CSPs today store cloud data in multiple locations for two main reasons. First, CSPs' reputation regarding the privacy and integrity of user data has been hurt recently.[17] Second, there are technical reasons to opt for multiple geographically distributed storage schemes.[9] When data is assumed to be permanently available, the typical approach is to distribute data control and storage to numerous geographical locations that multiple users share, as illustrated in Figure 2.

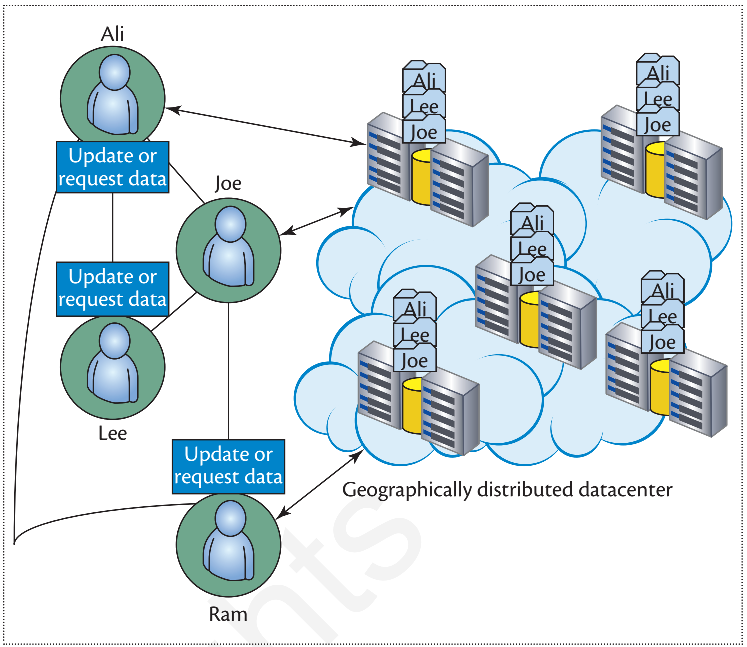

Figure 2. Storing users' data across multiple datacenters. These datacenters provide high availability of data storage services and an improved cloud user capability across broader geographical areas.

The key benefits of cloud storage with a multisite infrastructure are[9]

- an organization with multiple geographical areas,
- geolocation-sensitive data, and
- data locality and functionality close to the user.

Therefore, when auditing algorithms are proposed, data auditing techniques should always include these as standard features. The CSP's infrastructure offers a scalable, secure, and reliable atmosphere for users on a pay-per basis. Cloud storage services are used to share data with team members, as data sharing is the most common feature in most CSPs, including major datacenters like Dropbox and Google Docs.

Cloud Data Auditing

Numerous auditing schemes have been proposed, including MAC-based,[18] homomorphic,[19] and BLS-based homomorphic methods.[20] Therefore, much of the research on cloud data auditing focuses on the verification, privacy, preservation, and integrity of the saved data using cryptography techniques.

Auditing Process Analysis

Both Ateniese and his colleagues and Juels and Kaliski have proposed proof of retrievability (POR) and provable data possession (PDP) auditing schemes.[10,16] These schemes enable the cloud storage system to produce proof of a client's data without retrieving data from the system. These models demonstrate the minimum use of I/O cycles between the client and server. However, POR methods aren't suitable for third-party auditing schemes because the file is divided into blocks of data, and each block of data is encrypted.[16] During the auditing process, the client or verifier should explicitly mention the position of the block for verification; this technique is applicable only to static cloud data.

Another method involves the privacy-preserving public auditing of stored data, proposed by Cong Wang and his colleagues, who also advised the use of a TPA to efficiently and simultaneously perform data audits for multiple users.[2]

Privacy as a service was put forth by Kui Ren and his colleagues, who proposed a security protocol that provides security and privacy feedback for the client when storing and retrieving data.[21] Data protection as a service brings data security and privacy and deals with the evidence of privacy for data owners in the presence of potential threats.[7]

Chang Liu and his colleagues formally studied and proposed a scheme that supports authorized auditing and fine-grained update requests.[6] Kan Yang and Xiaohua Jia also discussed a third-party storage auditing service that guards data privacy; the auditor mixes cryptography modules with the bilinearity property of bilinear pairing.[22] Yang and Jia extended their work by implementing a random Oracle model for batch auditing for multiple owners and multiple clouds without any third-party cloud auditing.[8] Table 1 compares recent auditing algorithms and services, with various functions, techniques, and programming libraries.

MACs, signatures, and tags are the foundations of auditing algorithms. However, they also contribute to storage overhead. For example, MAC-based solutions must store the MACs for each block of data, whereas homomorphic linear authenticators have much less storage overhead because the tags for a linear combination of multiple messages can be homomorphically unified to form a single tag.[1]

BLS-based auditing algorithms have an edge over MAC and homomorphic methods because they, with the help of a homomorphic linear authenticator, support public auditing and data dynamics.[7] Furthermore, BLS's signature size is much shorter than the RSA-based homomorphic algorithms.[15] POR and PDP methods are also built with BLS signature schemes using verifiable homomorphic linear authenticators; however, these algorithms can't maintain the auditing process's privacy.[10,16] POR methods are used to aggregate proof of small authenticator values; hence, public irretrievability is achieved only for static data.[20] Dan Boneh and his colleagues propose dynamic provable data possession as the extension of POR methods.[14] Moreover, Qian Wang and his colleagues uncover POR's and PDP's security shortcomings through a proposed verification protocol with public auditability for dynamic data support.[15]

In HA schemes, the client must pay extra attention to store the data blocks or file tags apart from the file itself. Another shortfall of HA is the uniquely generated tags, which aren't repeated at all. Eventually, these random values (tag index values) will run out. Furthermore, the tag indexed value is directly proportional to the file size; hence, CPU processing will be greater for the larger files (files are usually represented as a combination of sectors).[10] However, malleability is often undesirable because it allows an adversary to form a ciphertext into another ciphertext, which decrypts the plaintext. However, HA tags have been shown recently to help achieve nonmalleability by combining linear blocks of data so that adversaries can't produce valid signatures.[1]

On the other hand, communication costs are directly proportional to the number of parties involved in the auditing process, apart from the size of the data transferred between them. In most cases, only two parties are involved; hence, signature tag size is crucial to communication cost. Communication costs are less with BLS-based algorithms because they use smaller signature tags.[7]

Computational complexity during TPA-based auditing is based on three entities: the auditor, the server, and the client who owns the data. For the lowest computational complexity and storage overhead, the algorithm should divide the file to be audited into a combination of blocks or sectors.[7] However, finding a design that integrates uniform allocation methods and auditing schemes for data storage services is challenging in cloud computing. Therefore, the proposed schemes can't provide data privacy because the TPA doesn't retrieve data using a data generator key algorithm. The drawbacks of auditing systems with respect to MACs are that security bits are only 180 bits or 20 bytes, and 1-Gbyte data blocks will have 53,687,091 tags and the same number of network transactions between the client and the server. Figure 3 shows the number of tags for different data sizes.

Privacy and Integrity

Remote verification (integrity) of data would allow third-party verification apart from the users themselves.[10,15,16] Provable data possession allows a client machine to verify remote data without downloading it.[10] This technique employs the probabilistic possession of a random dataset from the remote server with the help of homomorphic linear authenticators. However, to achieve deterministic verification, the client must access the complete data block. Qian Wang and his colleagues extended the work on proof of storage for data dynamics by using tree datatypes for block-tag authentication.[15] They achieved public auditability for dynamic data operations and blockless verification. Previous work showed that if the server has a corrupted

Table 1. Auditing algorithms and services.

Function	Technique	Service type	Cloud service	Programming library
Proofs of retrievability[16]	Prior to archiving a file, the auditor computes and stores a hash value	Verify	No	No
Homomorphic encryption[10]	Local file can be deleted, provided its metadata is locally saved	Verify	No	No
Homomorphic linear authenticator with random masking technique[2]	Third party; allows batch auditing of remote data	Privacy-preserving public auditing	No	Yes
Boneh–Lynn–Shacham signature and Merkle hash tree[7]	Fine-grained dynamic data updates	Verifiable fine-grained dynamic data operations	Yes	No
Homomorphic authenticator[15]	Performs multiple auditing tasks simultaneously; data dynamics for remote data integrity	Verifiable dynamic data operations	No	Yes
Homomorphic message authentication code (MAC; Li Chen et al.)[26]	Auditing of shared data	Privacy preserving	No	Yes
Bilinear map (Boyang Wang et al.)[24]	Batch auditing for multiword and multicloud	Privacy preserving	No	Yes
Homomorphic authenticable ring signatures[8]	Signature maker on each block in shared data is kept private from a third-party auditor	Privacy preserving	No	Yes
Homomorphic MAC (Cong Wang et al.)[27]	Auditing of shared data	Integrity	No	Yes
Homomorphic authenticable proxy (Boyang Wang et al.)[28]	Batch auditing for the shared data	Integrity	No	Yes
Multicloud[29]	Signature maker on each block in shared data is kept private from third-party auditor	Privacy preserving	No	No

file, then the verifying phase of the cloud-auditing algorithm would detect this misbehavior with a probability of $(O)1$.[10,15,16]

Furthermore, these integrity schemes have faced difficulties when verifying small data updates.[15] One of the most well-known techniques for data integrity is the ranked Merkle hash tree, extended for the cloud-auditing scheme.[7] This technique maintains data integrity based on the signature scheme[23] and provides authorized auditing (which avoids a malicious user posing as a TPA). This scheme is similar to the binary tree, wherein each node N will have a maximum of two child nodes. Each node is represented as $\{H, rN\}$, where H is the hash value and rN is the rank of the node. A leaf node LN contains the message or data block m_i, and its H value is calculated as $H(m_i)$, rLN. This tag generation scheme σ uses the following equation:

$$\sigma = \left(H(m_i) \prod_{j=1}^{s_i} u_j m_{ij} \right),$$

where $u_j \in U$, $U = \{u_k \in Zp\}$, $K \in [1, s_{max}]$ file blocks of sector s. Segment F is divided into $\{m_{ij}\}$, where i is the block length and j is the set of sectors s.

An important feature of this scheme is that during the challenge phase, the TPA should receive an authentication tag from the cloud client for auditing; an adversary can't challenge TPA without this authentication tag. Furthermore, data auditing protocols, especially those designed for integrity, are unable to protect data privacy against the TPA.[3,10,15,16] It's already been reported that a TPA might obtain the data information by recovering the data blocks from the data proof phase of the auditing process.[17]

All these proposed protocols have missed the importance of data privacy. Hence, new privacy-preserving protocols are critical for maintaining data integrity and privacy.[3] For example, Cong Wang and his colleagues have used a public key–based homomorphic linear authenticator.[2] This empowers the TPA to perform auditing without downloading the data; hence, the proposed algorithm reduces the communication overhead compared to general auditing techniques. Furthermore,

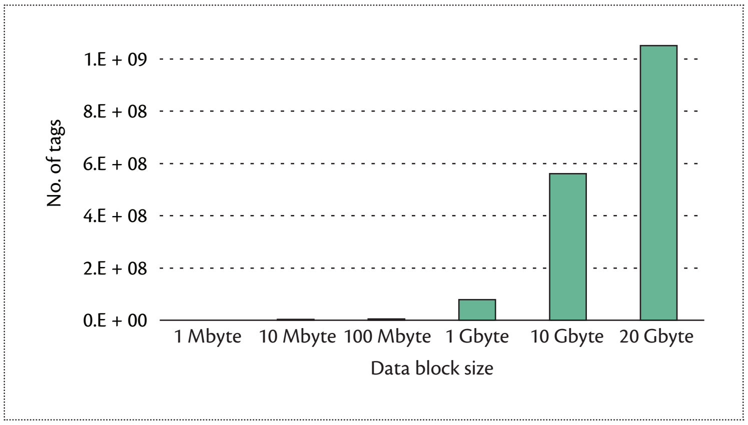

Figure 3. Transaction between cloud server and auditor: generation of security tags between TPA and cloud storage during auditing.

it also random-masks the auditing process to prevent the TPA from learning about the data. However, some cloud service features have become issues for cloud data; for instance, multitenancy, which means that the cloud platform (VM concepts) is shared and used by various geographically distributed users. Hence, the data dynamic feature should be equipped with auditing algorithms.

Cong Wang and his colleagues also implemented an authenticator for each block of data;[3] cloud users should attach metadata during the auditing process's setup. In response, the server will encapsulate the corresponding auxiliary authentication information during the audit phase and calculate the aggregated authenticators σ as follows:

$$\sigma = \prod_{i \in I} \sigma_i^{vi} \in G_1 .$$

For each element $i \in I$, the TPA chooses a random value vi. Then, it transmits the proof of correctness to the TPA as $\{\mu, \sigma, R\}$, where μ is the combination of the sampled blocks specified during the challenge phase and R is the random element belonging to the multiplicative cyclic group. Then, the TPA calculates the following as the verification equation:

$$R \cdot e(\sigma^{\gamma}, g) = e\left[\left(\prod_{i=s_1}^{Sc} H(W_i) v_i\right)^{\gamma} \cdot \mu^{\mu}, v\right],$$

where H is a secure map-to-point hash function, W_i is the name of the file, and $i\{s_1 \ldots s_c\}$ is the set of challenge elements. The public parameters are $\{v, g, e(u, v)\}$. Boyang Wang and his colleagues proposed a privacy-preserving protocol for shared data in the cloud environment.[24] They utilized ring-based signatures to construct a homomorphic authenticator. With such an approach, the TPA can't determine the block's signer. Batch auditing for the shared data is also achieved with the help of bilinear maps; auditing shared data for different users is a single auditing job assigned to a TPA.

Ring-based signing includes public keys, $(\mathrm{pk}_1 \ldots \mathrm{pk}_d) = (\omega_1 \ldots \omega_d)$, for all d users, and a block of data with its identifier, $m \in Z_p$ and id, respectively. The ring signature σ of the block is given by the following equation:

$$\sigma_s = \left(\frac{\beta}{\varphi\left(\prod_{i \neq s} \omega_\sigma\right)^{ai}}\right)^{1/x_s} \in G_1 .$$

To produce ring signature for s users, we need to have computable isomorphism φ, a_i and x_i randomly picked prime ordered Zp, and public key ω_i and private key sk, for identifier $i \in [1, \mathrm{d}]$. The verifier first computes challenge β as follows:

$$\beta = H_1(id) g_1^m \in G_1 ,$$

where $H_1 \{0, 1\}^* \rightarrow G_1$ is a public map to point to the hash function. G_1 is the cyclic group of order g_1.

Yang and Jia suggested using the bilinearity property of bilinear pairing during the proof phase of auditing.[8] However, the auditor can still check the proof's correctness without having to decrypt the proof. Furthermore, this is the first proposal to have batch auditing on multiclouds and multiowners along with data dynamics. In this scheme, the key generation algorithm is a combination of a secret public key $(sk_t\text{–}pk_t)$ and a hash secret key $(sk_{h,kl}) \in S$, such that there exists a different hash key for each server. Each data component is denoted as M_{kl}, where M is the data component owned by owner O_k, and maintained by server S_l. $W_{kl,i}$ is the data block of owner k and server l, and i represents the data block of m_i. J is the sector number and is represented as $j \in [1, s]$. For each sector, data tags are calculated as

$$u_j = g_1^{X_j} \in G_1 .$$

To generate the batch proof, after receiving a challenge, each server (site) generates a tag proof as follows:

$$TP_l = \prod_{k \in O_{chal}} \prod_{i \in Q_{kl}} t_{kl,i}^{Vkl,i} ,$$

where Q is the challenge set of data blocks and k is the owner of the data block for which the server generates the O_{chal}. The challenge phase takes information, such as a set of owners and a set of cloud servers. This information is used to generate a random number $V_{kl,i}$ for each chosen data block, as tag $t_{kl,i}$.

It's been proven that cryptography isn't a suitable solution for cloud privacy.[25] The abovementioned studies have done tremendous work on data dynamics and batching auditing; yet, many obstacles to maintaining data privacy remain. Most of the proposed auditing techniques are built on the assumption that TPAs are trustworthy—an unscientific assumption that might lead to privacy issues. TPAs will possess data that might be encrypted or random-masked during the proofing phase of the auditing process. Thus, during this crucial phase of auditing, researchers must very carefully utilize cryptographic systems to secure the data from the TPAs. With random masking, TPA can't retrieve the data using linear equations.[3] Hence, we can conclude that randomization of data blocks and tags is the most suitable technique for preventing data leakage during auditing's proofing phase.

Another layer of information can also be implemented to protect data privacy: encrypting the data before passing it to the cloud owner. The auditing process consists of the exchange of cryptographic keys between the servers and a TPA. A typical challenge sent to the server consists of the position of the data block and a random value stored in some variable. According to Minqi Zhou and his colleagues,[5] this variable can be static or private to the customer C_1, and preserved at the server S_1. Because of multitenancy, the private area is shared with the other customers of S_1. Adversaries could benefit from this multitenancy and exploit the preserved data.

Cloud-Auditing Architecture Analysis

The research we've discussed thus far never mentions auditing architecture. The rapidly increasing use of cloud service applications and services on devices has led CSPs to evolve their technological approach to dynamic requirements. Cloud computing, which is elastic in nature, can fulfill these dynamic requirements by providing a suitable architecture for the services chosen by the client. Cloud architecture isn't monolithic—it can contain several modules stored in different machines in the cloud architecture. These machines, or nodes, have dedicated tasks, some of which are dynamic, and each act according to the demand presented. Furthermore, most of the nodes must be programmed to provide services according to the CSPs' service offerings. The cloud architecture will vary according to the services offered. Provisioning the services is a major contribution of the cloud.

CSPs usually provide clients with dynamic resource allocation such that the CSP doesn't over- or underprovide resources. Numerous CSPs are commercially available; thus, the first questions clients should ask before opting for CSP services (such as information as a service, platform as a service, and software as a service) are whether the CSP provides elasticity of services, will meet the agreed service level of the agreement, and has the required architecture to run the desired services. An architecture for auditing should also be provided, where all the modules responsible for the auditing process are programmed and assigned a duty for their role in the process. The cloud service architecture is the cloud's backbone—a group of components or modules that work together to achieve certain tasks. This group of modules is loosely coupled together to achieve elasticity and varies with the CSPs' service offerings. To gain the faith of cloud users, CSPs must have flexible services.

Future Research

There are many avenues for future research in this particular area of cloud computing. For instance, unified data storage space allocation according to users' requirements and facility reservations could reduce the cost and time involved in the auditing process. Haar Wavelet matrix operations have only addition, and many of its elements are zero. By compressing data before transferring it to the client, data integrity is maintained and costs decrease. Furthermore, clients or end users can schedule auditing based on data usage. If the most recently used data is scheduled for later use, it will be less of a burden on the server hosting the data. This will also benefit data with multiple owners.

Another potential research area involves giving cloud providers responsibility for maintaining client files' metadata. Client metadata can be placed in the blocks that are protected by software-based memory locks. Hence, this process can be used for data dynamics. Furthermore, it can reduce communication costs and computation for carrying secured tags between the TPA and the server. This process is also helpful for avoiding data retrieval from the tags by the TPA because it resides only in the cloud.

Studying the framework of an interaction-based system using a graphical dynamic system would also be useful. Because the communication path between the TPA and the server can't be predicted, data integrity and privacy remain secure.

From the data auditing perspective, the technical challenges of auditing services can be addressed by employing a separate architecture for auditing purposes. Data stored on the cloud comes from devices with different backhaul networks, such as 2G, 3G, LTE, and 4G. These architectures have different network delivery systems and must be synchronized to provide seamless connections. Apart from these differences, heterogeneous data also exists; although this data runs only on IP networks, it contains different types, such as audio, video, image, or text message. Hence, the data also has different provisioning requirements that must be harmonized to provide rational knowledge to the cloud client.

The core network of cloud architecture attempts to differentiate between victims and intruders with random early detection (RED) and weighted RED. Hence, when auditing is scheduled, these routers must be able to differentiate between valid traffic and intruders. Thus, another potential area for research is the introduction of explicit congestion notification (ECN), so that the application-level quality of service (QoS) can be met. Most of the techniques applied in service-level agreement (SLA) verification analyze QoS metrics at the domain gateways to discover abnormal activities. The SLA verifier agent can be used to recognize machine/end users with an ECN echo notification; it can further probe the traffic for the packet transmission rate. RED always monitors the average queue size of the network edge router to avoid the threshold limits.

We hope this analysis will help the research community develop more secure methods of auditing cloud data. ■

Acknowledgments

We greatly appreciate the deanship of research at Prince Sattam University for allowing us to perform this study. We also express our thanks and appreciation for the editors' and reviewer's efforts in improving the manuscript.

References

1. B. Wang, B. Li, and H. Li, "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," *Proc. IEEE 5th Int'l Conf. Cloud Computing* (CLOUD 12), vol. 2, no. 1, 2012, pp. 295–302.
2. C. Wang et al., "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," *Proc. 29th Conf. Information Communications* (INFOCOM 10), 2010, pp. 525–533.
3. C. Wang et al., "Privacy-Preserving Public Auditing for Secure Cloud Storage," *IEEE Trans. Computers*, vol. 62, no. 2, 2013, pp. 362–375.
4. M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," *Proc. 16th Ann. Int'l Cryptology Conf. Advances in Cryptology* (CRYPTO 96), 1996, pp. 1–15.
5. M. Zhou et al., "Security and Privacy in Cloud Computing: A Survey," *Proc. 6th Int'l Conf. Semantics Knowledge and Grid* (SKG 10), 2010, pp. 105–112.
6. C. Liu et al., "Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-Grained Updates," *IEEE Trans. Parallel and Distributed Systems*, 2014; doi:10.1109/TPDS.2013.191.
7. G. Ateniese, S. Kamara, and J. Katz, "Proofs of Storage from Homomorphic Identification Protocols," *Proc. 15th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology* (ASIACRYPT 09), 2009, pp. 319–333.
8. K. Yang and X. Jia, "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing," *IEEE Trans. Parallel and Distributed Systems*, vol. 24, no. 9, 2013, pp. 1717–1726.
9. C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," *Proc. 41st Ann. ACM Symp. Theory of Computing* (STOC 09), 2009, pp. 169–178.
10. G. Ateniese et al., "Provable Data Possession at Untrusted Stores," *Proc. 14th ACM Conf. Computer and Communications Security* (CCS 07), 2007, pp. 598–609.
11. L.A. Bastião Silva, C. Costa, and J.L. Oliveira, "A Common API for Delivering Services over Multi-vendor Cloud Resources," *J. Systems and Software*, vol. 86, no. 9, 2013, pp. 2309–2317.
12. D. Boneh and D.M. Freeman, "Homomorphic Signatures for Polynomial Functions," *Proc. 30th Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques: Advances in Cryptology* (EUROCRYPT 11), 2011, pp. 149–168.
13. A.L. Ferrara et al., "Practical Short Signature Batch Verification," *Proc. Cryptographers' Track at the RSA Conf. 2009 Topics in Cryptology* (CT-RSA 09), 2009, pp. 309–324.
14. D. Boneh et al., "A Survey of Two Signature Aggregation Techniques," *RSA CryptoBytes*, vol. 6, no. 2, 2003, pp. 1–10.
15. Q. Wang et al., "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," *IEEE Trans. Parallel and Distributed Systems*, vol. 22, no. 5, 2011, pp. 847–859.
16. A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," *Proc. 14th ACM Conf. Computer and Communications Security*, 2007, pp. 584–597.
17. D. Zissis and D. Lekkas, "Addressing Cloud Computing Security Issues," *J. Future Generation Computer Systems*, vol. 28, no. 3, 2012, pp. 583–592.
18. B. Kaliski and M. Robshaw, "Message Authentication with MD5," *RSA CryptoBytes*, vol. 1, no. 1, 1995.
19. D. Boneh and D.M. Freeman, "Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures," *Proc. 14th Int'l Conf. Practice and Theory in Public Key Cryptography* (PKC 11), 2011, pp. 1–16.
20. D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *Proc. 7th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology* (ASIACRYPT 01), 2001, pp. 514–532.
21. R. Kui, C. Wang, and Q. Wang, "Security Challenges for the Public Cloud," *IEEE Internet Computing*, vol. 16, no. 1, 2012, pp. 69–73.
22. K. Yang and X. Jia, *Security for Cloud Storage Systems*, Springer, 2014.

23. R.C. Merkle, "A Certified Digital Signature," *Proc. Advances in Cryptology* (CRYPTO 89), 1989, pp. 218–238.
24. B. Wang, B. Li, and H. Li, "Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud," *Proc. 10th Int'l Conf. Applied Cryptography and Network Security* (ACNS 12), 2012, pp. 507–525.
25. M. Van Dijk and A. Juels, "On the Impossibility of Cryptography Alone for Privacy-Preserving Cloud Computing," *Proc. 5th USENIX Conf. Hot Topics in Security* (HotSec 10), 2010, pp. 1–8.
26. L. Chen et al., "An Efficient and Privacy-Preserving Semantic Multi-keyword Ranked Search over Encrypted Cloud Data," *Int'l J. Security and Its Applications*, vol. 8, no. 2, 2014, pp. 323–332.
27. C. Wang et al., "Toward Secure and Dependable Storage Services in Cloud Computing," *IEEE Trans. Services Computing*, vol. 5, no. 2, 2012, pp. 220–232.
28. B. Wang, B. Li, and H. Li, "Panda: Public Auditing for Shared Data with Efficient User Revocation in the Cloud," *IEEE Trans. Services Computing*, vol. 8, no. 1, 2015, pp. 92–106.
29. W. Itani, A. Kayssi, and A. Chehab, "Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures," *Proc. 8th IEEE Int'l Conf. Dependable, Autonomic and Secure Computing* (DASC 09), 2009; doi:10.1109/DASC.2009.139.

Manjur Kolhar is an assistant professor of computer science and information at Prince Sattam Bin Abdulaziz University. His research interests include computer science, computer networks, cloud computing, and resource management. Kolhar received a PhD in heterogenous IP communication systems from the National Advanced IPv6 Centre of Excellence, Universiti Sains Malaysia. Contact him at manjur.kolhar@gmail.com.

Mosleh M. Abu-Alhaj is a senior lecturer at Al-Ahliyya Amman University. His research interests include VoIP, multimedia networking, congestion control, and cloud computing. Abu-Alhaj received a PhD in multimedia network protocols from Universiti Sains Malaysia. Contact him at m.abualhaj@ammanu.edu.jo.

Saied M. Abd El-atty is an associate professor in the department of Electronics and Electrical Communication Engineering of Faculty of Electronic Engineering at Menoufia University. His research interests include design, analysis, and optimization of wireless mobile communication networks and vehicular networks; nanonetworking; and molecular communications. Abd El-atty received a PhD in wireless communication networks from the University of the Aegean. Contact him at sabdelatty@gmail.com.