2014

Survey on securing data storage in the cloud

Lei Huang
Loyola Marymount University

SIP (2014), vol. 3, e7, page 1 of 17 © The Authors, 2014.

doi:10.1017/ATSIP.2014.6

OVERVIEW PAPER

Survey on securing data storage in the cloud

CHUN-TING HUANG[1], LEI HUANG[2], ZHONGYUAN QIN[3], HANG YUAN[1], LAN ZHOU[4], VIJAY VARADHARAJAN[4] AND C.-C. JAY KUO[1]

Cloud Computing has become a well-known primitive nowadays; many researchers and companies are embracing this fascinating technology with feverish haste. In the meantime, security and privacy challenges are brought forward while the number of cloud storage user increases expeditiously. In this work, we conduct an in-depth survey on recent research activities of cloud storage security in association with cloud computing. After an overview of the cloud storage system and its security problem, we focus on the key security requirement triad, i.e., data integrity, data confidentiality, and availability. For each of the three security objectives, we discuss the new unique challenges faced by the cloud storage services, summarize key issues discussed in the current literature, examine, and compare the existing and emerging approaches proposed to meet those new challenges, and point out possible extensions and futuristic research opportunities. The goal of our paper is to provide a state-of-the-art knowledge to new researchers who would like to join this exciting new field.

Keywords: Cloud computing, Data security, Data integrity, Confidentiality, Access control, Searchable encryptions, Data availability

Received 16 September 2013; Revised 20 March 2014; Accepted 24 March 2014

I. INTRODUCTION

Rapid advances in broadband communication and high-speed packet switching networks have made large file sharing much more effective during the past two decades. Consequently, the demand for rich media applications, such as multimedia mails, orchestrated presentations, high-quality audio and video sharing, and collaborative documents, has grown tremendously. The amount of data and computing resources being used by those applications have also grown exponentially. As a result, the costs of IT service and support, such as investment in new hardware and software, staffing for installation and maintenance are rising consistently for both enterprises and individual users. Therefore, cloud computing has become an appealing new model of IT service provisioning and support driven by economic and productivity advantages. Instead of investing in new hardware and software, as well as maintaining those resources, users can use applications, infrastructures, servers, storage, network, and other computing resources that are available in the 'cloud', which is a shared pool of computing resources that can be easily accessed through broadband network connections. This new IT service provisioning model offers users seemingly unlimited computing resources without up-front acquisition and/or sustaining maintenance costs. Moreover, it offers on-demand elasticity and flexibility in using computing resources. The utility pricing model allows users to pay for their actual usage only.

Storage, as one of the most influential and demanding computing resources in current digital era, is among the first being moved into the cloud. This type of cloud computing services, known as cloud storage, represents a business model in which the service provider rent spaces in their large-scale storage infrastructure to organizations and individuals. It has always been one of the most prevalent services in cloud computing industry. As an extension of traditional data center or file hosting service into cloud, cloud storage has distinct characteristics including on-demand self-service, broadband network access, resource multiplexing, rapid elasticity and measured usage for utility billing. Besides the key advantages of cost saving, cloud storage can facilitate information sharing and task collaborating, promote portability and universal accessibility of data, as well as provide easy and convenient solutions to some other problems. For example, for disaster recovery purpose, organizations should maintain secondary off-premise data backups. Storage of sensitive data, such as financial, personal, or medical data are subject to more and more regulations and legal constraints. Cloud storage offered by a regulation-complied service provider can relieve data owners from the complicated process.

However, the promising new paradigm of cloud computing brings up unique challenges in terms of performance, availability, security, and scalability (known as

[1] Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA
[2] Loyola Marymount University, Los Angeles, CA, USA
[3] Information Science and Engineering School, Southeast University, Nanjing, Jiangsu, China
[4] Information and Networked Systems Security Research, Macquarie University, North Ryde, Australia

Corresponding author:
Lei Huang
Email: lei.huang@lmu.edu

PASS). Among these challenges, security issues have been reported as the biggest concern preventing enterprises and organizations from adopting cloud services according to recent researches [1]. Therefore, it is imperative to provide security strategies, tools, and mechanisms that meet user's requirements in the cloud. Security in cloud computing is a complex issue spanning across many aspects including physical security, infrastructure (distributed computers, servers and other hardware) security, data security, network security, and software security. Moreover, it involves shared responsibilities and obligations among the constituents of the cloud service. Security enforcement would not be successful without agreement, trust, regulations, and coordination among service providers and cloud users.

Since storage is one of the necessary core infrastructure in clouds, security of data in storage is one of the key concerns of any cloud computing systems, particularly in cloud storage services. The consequences of security breaches in cloud storage could be seriously damaging to both service providers and users. Without trust from users, the service provider could lose their customers. On the other hand, users whose valuable data lost, or sensitive information hacked could experience irrecoverable loss or damage. There have been many cases reported as threats of cloud storage security. Many leading service providers, including Amazon, Window and Google, encountered disconnections of their web-based cloud services due to different reasons such as power failure, hardware, and software failures. For instance, Amazon Web Service's server was hit by lightning, causing destruction on power generator. Although Amazon successfully transferred data to a backup server, the service still stopped after their *uninterruptible power supply* (UPS) went out. There was bulk email deletions in Gmail happened in 2006; numerous users found that they lost their emails and contact information without further notification from Google. Google was unable to restore the accounts after users responded the problem. Another incident happened recently in July, 2012. Because of a security loophole on the access control, Dropbox, a popular cloud storage service, was attacked by hackers. Some users reported that they received tons of spam emails, and some users' passwords were even leaked.

Although the security requirements for cloud storage vary with different applications and users, they share the same three basic objectives as any computer information systems [2]: integrity, confidentiality and availability. Many different tools have been developed to achieve these objectives, such as authentication, access control, encryption, certification, audition, and digital signature. This paper aims at providing a thorough study on recent data security mechanisms developed for the cloud storage. Based on the results of the study, we give our insights and suggestions on the future research directions in achieving each security objectives.

The rest of this paper is organized as follow: the models of cloud storage systems and related security implications are firstly introduced in Section II. A general conceptual system architecture model of the cloud storage is also proposed to address security issues in different layers. In Section III, recent researches on data integrity protection such as proofs of retrievability (POR) and third party audition are reviewed and compared. In Section IV, data confidentiality and its related research work are discussed. A promising new encryption technique, fully homomorphic encryption (FHE), which allows algebraic operations performed on encrypted data, is examined in detail first, followed by discussion on access control and searchable encryption. In Section V, we probe methods for ensuring data availability in distributed cloud storage systems, such as data synchronization, data recovery, and information dispersal algorithms. At last, concluding remarks are given in Section VI.

II. OVERVIEW OF SECURITY IN CLOUD STORAGE

The scope and requirements for cloud security vary significantly with different cloud deployment models. National Institution of Standards and Technology (NIST) has defined [3] four deployment models of cloud computing. *Private cloud* is provisioned for exclusive use by a single organization comprising multiple users. *Community cloud* is provisioned for exclusive use by a specific community of users from multiple organizations that have shared concerns. *Public cloud* is provisioned for open use by the general public. *Hybrid cloud* is a combination of at least two of the above three. Apparently, the price of deployment decreases from private cloud to public cloud at the cost of increasing security concerns.

Cloud storage can be deployed in any of the four deployment models. Internet-based public cloud storage services are rapidly growing because they are able to provide users with biggest cost saving and most elasticity. Numerous storage service providers (SSPs), including Amazon, IBM, Google, Microsoft, EMC, HP, Symantec, Rackspace, to name just a few, are competing in this enormous market. However, these public cloud storage services also face highest potential risks of security breaches because the shared infrastructure is open to the public. As a matter of fact, cloud storage systems deployed in other forms of multitenancy clouds, including hybrid clouds and community clouds, are also exposed to higher risks than private cloud. Even in the private cloud, it is highly likely that cloud storage is managed and operated by a service provider off premises, in order to take the most advantage of the cloud computing. In fact, the cloud storage usage, such as disaster recovery backup, requires off-premises storage. When data are no longer stored and managed by the data owner on its own premises, the data owner has less control over their data. Therefore, cloud storage security is challenging if the service providers are not trusted, regardless of the deployment model.

NIST has also defined three primary cloud service models. *Software as a Service* (SaaS) implies consumers utilize service provider's application running on cloud infrastructure, such as SalesForce CRM, YouTube, and Google Apps (Gmail, Google Document). *Platform as a Service* (PaaS), means the service provider builds an environment for consumers to establish acquired applications with programming languages, libraries, and tools that are already supported in the platform. Famous PaaS includes Google App Engine, Microsoft Azure, and Cloud Foundry from VMware. *Infrastructure as a Service* (IaaS), represents consumers can deploy and manage application, operating system with provided network, and storage devices. Amazon's Elastic Compute Cloud (EC2) is a leading example, with other offerings such as Rackspaces Mosso and GoGrid's ServePath [3]. From SaaS to PaaS, and to IaaS, users have progressively deeper control over the stack of cloud architecture, thus share more responsibility on security enforcement.

Basic cloud storage services are categorized as IaaS service model, although many cloud storage providers are offering value-added PaaS and SaaS services built upon their baseline IaaS services. As an IaaS, cloud storage allows users to strengthen the security measure using their own security protection mechanisms. For example, users can encrypt their data before moving them into the cloud storage using a private key managed by themselves. In this case, even if the data were accessed by unauthorized parties, the sensitive information would not be revealed without obtaining the key. However, users of SaaS services can only rely on the service provider's security measures.

The basic architecture of a cloud storage system is composed of a storage resource pool, including the distributed file system, the *Service Level Agreements* (SLA), and service interfaces [4, 5]. In order to conceptually understand the cloud storage systems, and how security protection mechanisms could be integrated and implemented in the system, we decompose the system architecture into a three-layer reference model based on the logical function boundaries as shown in Fig. 1.

In physical storage infrastructure layer, there are distributed wired and wireless networks connecting a distributed storage device network. The second layer is storage management layer, which processes necessary operations, such as data placement, replication, and reduction, on the stored data in the first layer. By means of virtualization technology, this layer becomes the intelligent abstraction layer, which hides the complexity of the underlying layer. The service interface layer provides the interface for users to access their data stored in the cloud storage. Basic cloud storage systems mostly provides either a client-side software or a web browser interface, or sometimes both. Client-side software has to be installed on the user's devices used to access the data, whereas a web browser interface allows access of data from any place without local installations. Some advanced cloud storage systems also provide an *Application Programming Interface* (API), which can be used to directly integrate access of stored data into other applications. Most of those applications belong to PaaS or SaaS based on the cloud storage infrastructure.

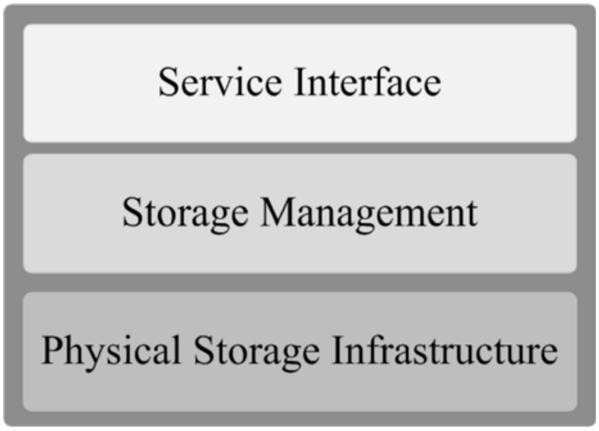

Fig. 1. Cloud storage architecture [5].

Since different layers have different functionalities, the security concerns in each layer have different emphasis. The physical storage infrastructure layer deals with physical and hardware security. The storage management layer should efficiently control the resource allocation and reliably perform data management. In the service interface layer, how to avoid the encroachment on rights of both clients and service providers using secure interfaces and APIs has been extensively discussed. In every layer, there could be risks, intrusions, and attacks against data integrity, confidentiality and/or availability. Therefore, storage security protection mechanisms should be integrated into every layer, and the security objectives cannot be achieved without the collaborated efforts across all three layers. For example, to ensure data availability under any circumstances such as hardware failure or disasters, the physical storage infrastructure layer usually have duplicated data stored at different locations. In case data stored in one location was lost, the storage management layer should be able to locate the available data in another location and route it to the users upon their request. The service interface layer should be able to effectively receive incoming requests from anywhere and provide reliable access method to legitimate users.

Given the architecture overview of cloud storage and its security implications, we will discuss recent research efforts in achieving the three main security objectives, namely, data integrity, data confidentiality, and data availability, in the following three sections, respectively.

III. DATA INTEGRITY

Data integrity refers to the property that data have not been altered or destroyed in an unauthorized manner [6]. In cloud storage, since users no longer possess the physical storage of their data, how to efficiently verify the correctness of out-sourced data stored in cloud server has become a challenging as well as a promising research topic for data storage security.

In traditional data communication networks, data integrity is usually threatened by malicious attackers only. Both the sender and the receiver of data are trusted and collaborated in detecting and protecting data integrity.

However, in cloud storage, the cloud storage servers (CSSs) are not always trusted. The cloud SSP has motivations to elude the service users on stored data status. For instance, A service provider may remove the rarely accessed data in order to economize the storage usage, or hide the data loss incidents for maintaining its reputation. Moreover, a malicious server may change or replace the stored data. In order to prevent the above instances, it is more valuable to have data integrity verification process in place and regularly query the correctness of data in storage servers. An effective verification mechanism can also allow the user to detect the threats of data integrity in cloud storage sooner, and take necessary actions to minimize the damage or recover the lost caused.

There are three basic requirements for data integrity verification process, namely, efficiency, unbounded use, and self-protect mechanism. Efficiency implies minimal network bandwidth and client storage capacity are needed for the verification process. The client does not need to access the entire data for verification purpose. Unbounded use represents verification process should support unlimited number of queries. Self-protect mechanism means the process itself should be secure against malicious server that passes the integrity test without accessing the data.

A number of different techniques and mechanisms have been proposed and designed for cloud data integrity verification process. The mainstream of research in this field belongs to *POR* and *Provable Data Possession* (PDP), both were designed to the above three requirements. The two methods originally emerged with a similar concept but different approaches. Since then, each one had gone through further development along different directions such as dynamic data support, public verifiability, and privacy against verifiers. Dynamic data support allows a client to dynamically update their data partially after uploading the data. Public verifiability enables everyone, not just data owner or verifier, to perform verification process. Privacy against verifiers ensures that the verification process does not contain any private information of data owner. POR and PDP schemes with their developments will be discussed and compared in more detail later in this section.

Besides those two approaches, there are several methods studied to address the storage data integrity issue resulted from data insertion, modification, and deletion at the block level. In 2010, *Proof of Erasability* (POE) scheme was proposed by Paul and Saxena [7]. POE addresses clients' need to ensure a comprehensive destruction of the stored data in the storage when they withdraw the data and disassociate with the storage provider. This model plays a role as probing engineering or destructor, which can ensure the stored data are shredded partially or fully based on the rules of data store. Nevertheless, this scheme only allows the data owner knowing the data are being destroyed. Another parallel scheme called *Proofs of Secure Erasure* (PoSE-s) also has a similar function on remote attestation [8]. Even though this scheme was proposed to replace hardware-based attestation, it is suitable for updating secure code and secure storage erasure for cloud.

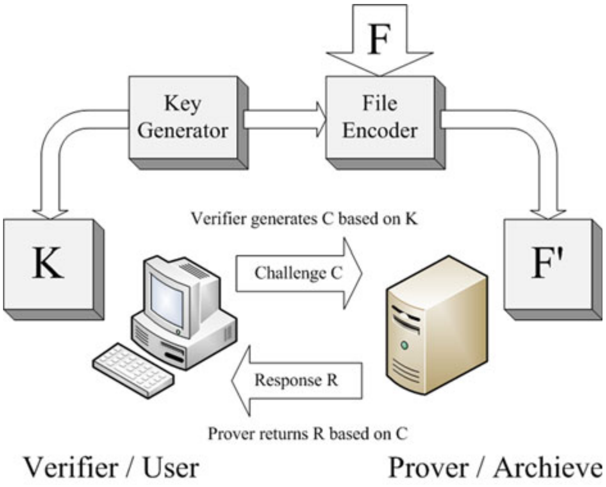

Fig. 2. Schematic of a POR system [9].

In the following subsections, we will first introduce POR and PDP, followed by their developments to improve efficiency, dynamic data support, and public verifiability.

A) Introduction to POR and PDP

As s widely studied mechanism to ensure data integrity, POR was firstly proposed by Juels and Kaliski in 2007 [9]. Figure 2 depicts the general schematic of the proposed POR system, which ensures the server (prover) to a client (verifier) that the stored data are intact during the storing and retrieving process of the client. The client first encode a raw file F through an encoding algorithm into an encoded file F' and then stores it in the prover. A key generation algorithm produces a key K stored in the verifier, and it is used to encode. For the checking process, the verifier can perform challenge–response process with prover in order to check if F can be retrieved.

The first POR scheme introduced by Juels and Kaliski employed a sentinel scheme. POR protocol encrypts F and inserts randomly several *sentinels* into the other file data blocks after encryption. These sentinels play an crucial role for verification. The verifier can challenge the prover by pointing out the positions of a collection of sentinels, and the prover should return the values of the sentinels. If the values are different from the verifier's data, then it shows that prover has deleted or modified F. POR also includes error-correcting code to recover a small portion F if corrupted. However, this scheme requires pre-processing and encoding of F prior to store into the data storage, and it is bounded use – number of sentinels can be used up for limited queries. Therefore, Juels and Kaliski proposed another technique from Lillibridge *et al.* [10], Naor and Rothblum [11]. It stores the redundantly encoded data blocks with *message authentication code* (MAC) to replace sentinels, and the MACs are stored together with data blocks. In this case, verification algorithm can examine the data integrity and ensures retrievability by requesting random number of block positions with their MACs. This approach resolves bounded use problem of the previous scheme, but at the cost of higher communication complexity of the audit.

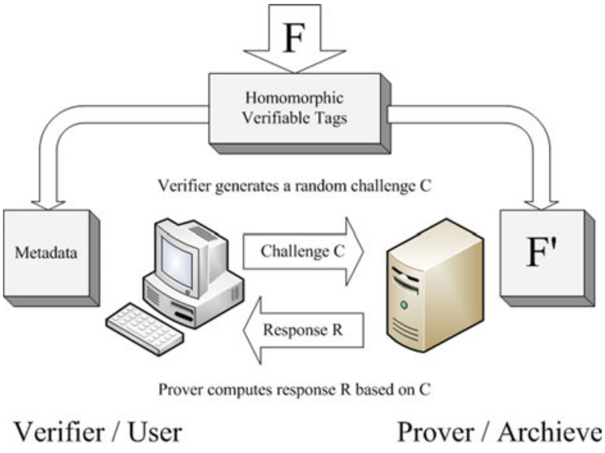
F
Homomorphic Verifiable Tags
Verifier generates a random challenge C
Metadata
Challenge C
Response R
F'
Prover computes response R based on C
Verifier / User
Prover / Archieve

Fig. 3. Schematic of a PDP system.

On the other hand, PDP came out concurrently with Juels–Kaliski's scheme. It was proposed by Ateniese *et al.* [12] and constructed based on symmetric key cryptography. PDP firstly chose RSA-based homomorphic verifiable tags [13] to combine multiple file blocks into a single value. A similar approach was also adopted later by Shacham and Waters [14]'s POR scheme in 2008. PDP scheme also provides data format independence, and it puts no restriction on the format of data. In other words, PDP allows any verifier (not only client) to query the server. POR and PDP both employed erasure code, which is a *Forward Error Correction* (FEC) for the binary erasure channel, helping recovery of the original message from slightly damaged data. The major difference between initial POR and PDP is that POR ensures not only data integrity at the server end but also retrievability, whereas PDP guarantees only data integrity at cloud data storage. Nevertheless, PDP is more efficient compared to Juels–Kaliski's POR, since it does not require any bulk encryption, and PDP requires smaller storage space on the client side and fewer bandwidths for challenges and responses. However, both schemes work on static data only, even though Ateniese *et al.* [15] proposed a dynamic version later in 2008, but it is restricted by number of queries and basic block operations (Fig. 3).

B) Improvement on public verifiability

Since the Juels–Kaliski's original POR scheme was proposed without implementation of public verifiability, and its complexity was still high for communication and client storage, it became a popular topic for researchers to improve public verifiability and efficiency (discussed in the next subsection). In 2008, Shacham and Waters [14] proposed two new PORs system structures based on Juels–Kaliski's POR concept. Both solutions allow only one authentication value for the purpose of verification. The first one is privately verifiable using pseudorandom functions (PRFs); the second one is publicly verifiable, and it was built based on signature scheme of Boneh, Lynn, and Shacham in a bilinear group [16]. Since the BLS signature was adopted, the public retrievability was achieved, and the proofs are reduced to a single authentication value, thus reduced communication complexity from $O(t)$ to $O(1)$, where t is the number of block positions. However, this scheme still works on static data only, without support of dynamic data update. Besides, the security parameter relies on Random Oracles, which means the client's challenge size grows up to $O(t^2)$.

A new system model, as depicted in Fig. 4, which aimed at establishing a trustable mechanism between client and *CSS* by introducing a *Third Party Auditor* (TPA), was proposed in 2009 [18]. By using a privacy-preserving third-party auditing protocol, the TPA is trusted to monitor the stored data in cloud and transactions between the client and CSS, as well as assess and expose risks of the cloud services. This new scheme has been further developed based upon existing PORs and newly developed cryptographic primitives [17, 19–22].

TPA typically adopts a public-key-based homomorphic authenticator with random masking to perform traffic auditing without a local copy of the data for integrity check. This public audit system can be constructed from the setup stage, which allows a user to initialize the secret parameters of the system, send the verification metadata to TPA, and audit the corresponding result. In this process, TPA will issue an audit message to the server for checking the user's data.

Homomorphic authenticators are used to verify metadata generated from individual data blocks while the aggregated authenticators can justify a linear combination of data blocks. As a paradigm, one can use a homomorphic token with distributed verification to check the integrity of erasure-coded data. The erasure-correcting codes play a vital role in preparing files for distribution so that the distributed files have redundancy parity vectors and the data dependability property. However, the linear combination of data blocks may potentially reveal users' privacy. With random masking, TPA cannot derive user's data content by building a correct group of linear equations.

The above model was further improved in [23] by integrating dynamic data support in 2011. Zhu *et al.* [24] also proposed a construction of dynamic audit services for untrusted and out-sourced storage. It can detect abnormal behavior by using fragment structure, random sampling, and index-hash table.

Even though TPA-based schemes allows public verification of data integrity checking, They have a potential obstacle that requires an additional constituency, which is a third party auditor, added to the entire existing data storage scheme. The implementation of such schemes might be a burden for service providers because of additional costs. To address this concern, Han and Xin [25] proposed a new scheme offering the traditional TPA functions provided by CSP in a trustful manner. This scheme utilizes RSA and Bilinear Diffie–Hellman techniques, creates message header and mechanisms to achieve authentication process, while reducing complexity of cloud computing. Another work that provides public verifiability without help from the third party auditor was examined in [26] based on the work of Sebe *et al.* [27], and it has been proved to be secure from an untrusted server.

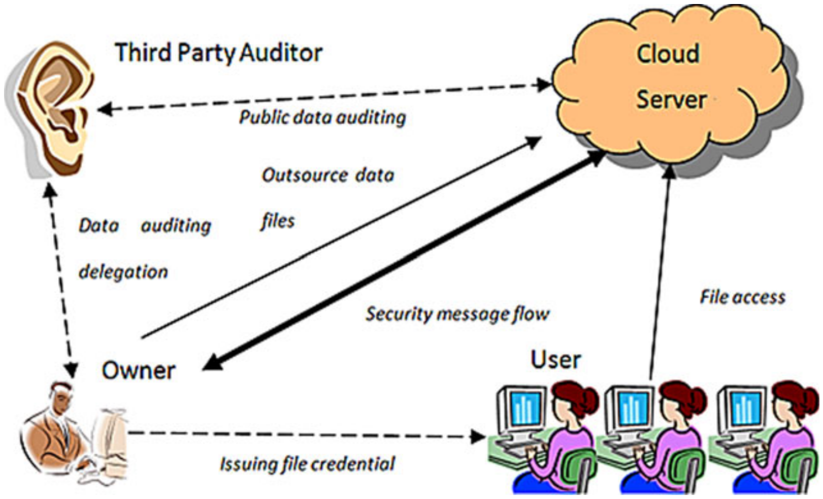

Fig. 4. TPA structure [17].

C) Improvement on efficiency

Efficiency of POR and PDP has been improved from different aspects of the verification process. For instance, Curtmola *et al.* [28] showed how to integrate error-correcting codes with PDP and an adversarial error-correcting code construction similar to PORs. It also enabled PDP scheme to secure multiple replicas over distributed system without encoding each separate replica.

Besides, Dodis *et al.* [29] provided different solutions of optimized POR schemes under different constraints, such as bounded-use or unbounded-use, knowledge-soundness, or information-soundness. They analyzed the tradeoffs in parameters and security between bounded and unbounded use schemes, and they also compared PORs under different circumstances in detail. It also improved the Shacham–Waters POR scheme by avoiding the usage of Random Oracles, which reduced the challenge size down to be linear in the security parameter, from $O(t^2)$ to $O(t)$.

In addition, a theoretical framework of PORs improvement was concurrently proposed by Bowers *et al.* [30]. The model offers an improvement over the protocols of Juels–Kaliski [9] and Shacham–Waters [31] by proposing a new variant to achieve lower storage overhead and tolerate higher error rates. The proposed POR scheme also decreased the challenge size to be linear of the security parameter. Another POR scheme was proposed by Kumar and Saxena in 2011 [32]. It targeted on simplification of Juels–Kaliski's sentinel scheme, making it suitable for limited computational power or small storage at verifier end. For PDP, Ateniese and Burns *et al.* concluded previous research developments and implementations of PDP in 2011 [33], and proposed two improved provably secure PDP schemes with higher efficiency than previous ones.

D) Improvement on dynamic data support

Supporting dynamic data update in data integrity verification schemes are especially challenging. Ateniese *et al.* [15] proposed the first partially dynamic PDP scheme in 2008. This scheme was more efficient in setup and verification phase compared to its previous version in [12], since it was only relied on symmetric-key cryptography. On the other hand, it only allowed a limited number of queries and basic block operations with limited functionality. For example, block insertion was not supported. Moreover, public verifiability was not supported either.

In 2009, Erway *et al.* proposed an improvement on PDP, referred as dynamic provable data possession (DPDP) [34]. In order to support provable updates on the stored data, this new model utilized authenticated directories based on rank information, and it defined the update as block insertion, modification, or deletion to achieve dynamic PDP scheme. Nevertheless, this scheme maintains skip list [35] for tags and stores root metadata in clients side to prevent replay attack, so its computational and communication complexity can be up to $O(\log t)$.

The dynamic data updates on POR were first considered in 2009. Wang *et al.* [17, 18] proposed the first scheme that achieved efficient data dynamics of the POR model by utilizing the homomorphic token with distributed verification of erasure-coded data, and manipulation of the *Merkle Hash Tree* (MHT) [36], respectively. The first scheme supported block update, delete and append operations only, while the second scheme provided both public verifiability and data dynamics for remote data integrity check, but the verification complexity increased to $O(\log n)$ from $O(1)$ as a trade-off, and it achieved partially dynamic instead of fully dynamic. Both schemes also showed a new system model involving *TPA*.

In addition, Zheng and Xu presented a new POR scheme with a fresh property, namely, fairness, to deal with dynamic data [37]. This property prevents unscrupulous clients from accusing a legitimate server about modifying their stored data. This issue arises because of the feature of dynamic data. POR for static data storage can solve this problem simply by asking the verifier to approve and sign digitally when the data has not been stored into the storage. The proposed *fair and dynamic proof of retrievability* (FDPOR) was mainly composed of two parts, a new authenticated data structure: *range-based 2–3 tree* (rb23Tree) and a new incremental signature scheme called hash-compress-and-sign. However, FDPOR did not support public verifiability, and complexity for both the verifier and the prover were higher than that of previous PORs.

E) Summary

PORs and PDPs are the major remote data integrity checking protocols proposed in cloud storage systems. The original POR and PDP protocols differs in many aspects. PORs are considered to be more secure compared to PDPs, because it requires encryption of the original data and error correction coding to recover damaged data, whereas PDPs are known for higher efficiency and applicability to large-scale public databases, such as digital libraries. With further improvement of each, the two schemes have been converging toward the same objectives. For example, although public verifiability and homomorphic verifiable tags were first known for PDPs, these characteristics are also applicable to PORs. On the other hand, some PDP variants may also adopt encryption and/or error correction coding tools to strengthen their security measurement. Therefore, it is all about making tradeoffs among security functionalities and efficiency.

In Table 1, we summarize the above reviewed POR and PDP schemes by a thorough comparison of their performances. It is noteworthy that schemes with dynamic data support suffers higher complexities compared to their counterparts. Future research directions include further improvements on efficiency and fully dynamic data support. To improve efficiency of those schemes, reducing communication cost and storage overhead are rightful considerations. However, fully dynamic data support is a challenging objective, because it increases complexity but reduces update information on server-end.

IV. DATA CONFIDENTIALITY

Data confidentiality in cloud storage security refers to the property that information stored in the cloud storage is not made available or disclosed to unauthorized individuals, entities, or processes. Access control and data encryption have been widely deployed to protect data confidentiality in the traditional data communication networks. It is natural to extend their deployment in cloud storage systems. For instance, *Secure Socket Layer* (SSL) and AES-256 bit encryption are adopted in Dropbox to ensure data security. However, data confidentiality in cloud storage systems faces new risks and challenges, thus calls for new techniques or improved mechanisms. In this section, we discuss new challenges faced by access control and data encryption mechanisms, as well as recent developments to meet those challenges of data confidentiality protection in cloud computing.

Although traditional encryption techniques can hide the information of data from the cloud server, it would not provide a satisfactory solution if users demand to compute on their stored data. Since the computing can not be functionally performed on the ciphertext, users would have to decrypt the data before performing any computation and re-encrypt after the computation. During this process, sensitive information could have been leaked to the curious server. Otherwise, user would be forced to compromise with the service provider by uploading plaintext and signing SLA, which exposes their data to higher risks. To solve this problem, there have been research attentions drawn to a newly proposed encryption primitive, namely, *FHE*, which allows ciphertext to be computed without affecting decipher process.

In the following of this section, we first examine new access control mechanisms with higher efficiency and fine-grain user control suitable for cloud storage. Then introduce some new concept of data encryption schemes, such as searchable encryption and FHE, and discuss their potential applications in protecting data confidentiality in cloud computing. Then, Other data confidentiality approaches are also briefly discussed. We provide our insights of the current research efforts and future directions in data confidentiality to summarize this topic.

A) Access control

As mentioned above, access control has been one of the key mechanisms to protect data confidentiality in traditional data networks. It is designed to block unauthorized users and malicious hackers from accessing data. Although the objective of access control in cloud storage does not differ from that in traditional data network, the requirement does change. Traditional access control enforced by the service provider could not stop a curious cloud service provider accessing users' sensitive data, which was stored in the service provider's infrastructure and managed by the service provider. A curious cloud storage server trying to derive sensitive information from its stored data, or from data operations performed by data owner and authorized users, is a new threat model against data confidentiality in cloud storage service. Moreover, a malicious service provider could intentionally leak the data to unauthorized parties for profit, or a malicious attacker could compromise the service provider and get unauthorized access to the data.

To address this challenge, cryptographic access control schemes that shifted the access control agency from the service provider to the users have been proposed. Instead of relying on untrusted service provider to grant access control, users can enforce their own access control by selectively granting different decryption access to a certain part of encrypted data. By means of encryption, the owners of data, i.e., cloud storage users who lost their physical control over their own data could regain their control at the semantic level.

Plutus [39] and SiRiUS [40] are examples of using encryption to secure file sharing on remote untrusted storage. These schemes encrypted different files with different keys, thus changing the problem of access to files to the problem of key management. However, this approach is not scalable when applying to cloud storage, because the complexity of key management increases with the number of

Table 1. Performance comparison for data integrity verification schemes

	Data dynamic	Public verifiability	Retrievability	Server comp.	Verifier comp.	Communication comp.	TPA
2007 JK [9]	Static	No	Yes	$O(1)$	$O(1)$	$O(t)$	No
2008 SW [31]	Static	Yes	Yes	$O(1)$	$O(1)$	$O(1)$	No
2009 Wang [17, 18]	Partially dynamic	Yes	Yes	$O(logt)$	$O(logt)$	$O(logt)$	Yes
2009 Dodis [29]	Static	Yes	Yes	$O(1)$	$O(1)$	$O(1)$	No
2009 Bowers [30]	Static	Yes	Yes	$O(1)$	$O(1)$	$O(1)$	No
2010 Wang [38]	Static	Yes	Yes	$O(1)$	$O(1)$	$O(1)$	Yes
2011 Saxena [32]	Static	No	Yes	$O(1)$	$O(1)$	$O(1)$	No
2011 Zheng [37]	Partially dynamic	No	Yes	$O(logt)$	$O(logt)$	$O(logt)$	No
2007 Ateniese [12]	Static	Yes	No	$O(1)$	$O(1)$	$O(1)$	No
2008 Ateniese [15]	Partially dynamic	No	No	$O(1)$	$O(1)$	$O(1)$	No
2008 Curtmola [28]	Static	Yes	No	$O(1)$	$O(1)$	$O(1)$	No
2009 Erway [34]	Fully dynamic	Yes	No	$O(logt)$	$O(logt)$	$O(logt)$	No
2011 Ateniese [33]	Partially dynamic	Yes	No	$O(1)$	$O(1)$	$O(1)$	No
2011 Hao [26]	Fully dynamic	Yes	No	$O(logt)$	$O(logt)$	$O(logt)$	No

files and/or the number of users, which both could be enormous in a cloud storage system. As a large number of users are sharing the same infrastructure in a public cloud storage built upon a complicated network scale, it is crucial to have efficient, scalable and reliable access control mechanism in place.

In the following, we examine recent research on more efficient access control using encryption techniques developed for cloud storage systems.

1) *Access control using attribute-based encryption (ABE)*
In attribute-based access control (ABAC) model, access is granted based on attributes of the user. When applied to cloud storage, access control is enforced on data encrypted using ABE schemes. In an ABE system, a user's keys and ciphertexts are labeled with sets of descriptive attributes. A particular key can decrypt a particular ciphertext only if there is a match between the attributes of the ciphertext and the user's key.

The concept of ABE was introduced by Sahai and Waters [41]. Their access control allowed for decryption when the number of overlapped attributes between a ciphertext and a private key exceeds a specified threshold k. The fuzzy nature of this scheme was originally designed for error-tolerant identity-based encryption scheme that could use biometric identities. However, with a threshold-based flat access structure, it could not be generalized to other applications. Two prominent ABE schemes with more general tree-access structures, namely, *Key-Policy Attribute-Based Encryption* (KP-ABE) [42] and *Ciphertext-Policy Attribute-Based Encryption* (CP-ABE) [43], were proposed in 2006 and 2007, respectively. Both algorithms associated a set of expressively descriptive attributes with a tree-access structures to enforce access control on the encrypted data, but they work in a reverse manner. In KP-ABE, each ciphertext was labeled with a set of attributes during encryption, while the users' private keys were associated with an access tree specifying which ciphertexts the key can decrypt. On the contrary, in CP-ABE, Users' private keys were based on a set of their attributes while ciphertexts are associated with an access tree over the attributes during encryption. As a result, in KP-ABE scheme, it is the key distributor (usually the service provider), who decides the access policy, while in CP-ABE scheme, it is the encryptor (usually the data owner) who controls the access over the encrypted data.

In the above-mentioned ABE schemes, the access policy can only contain logical formula "and" and "or", and threshold gates. A KP-ABE scheme was introduced in [44] which allows "negative" constraints to be represented in access policies. Additionally, many CP-ABE schemes were proposed such as [45–47] which either achieve chosen-ciphertext attack (CCA) secure or are built on different security assumptions. Even though the KP-ABE and CP-ABE work in reverse manner, Goyal *et al.* [48] provided a generic approach to transform a KP-ABE scheme into a CP-ABE one. Malek and Miri combined the two ABE schemes into one system, and proposed a balanced access control that allows both service provider setting up system wide access policies and data owner setting up access structure to their own data [49]. Further research on ABE is also discussed in [50, 51]. In a dynamic system, access policies may differ from time to time, and user qualifications may also change. Therefore, the ability to revoke attributes from a user is desired in ABE systems. Several revocable ABE schemes [52, 53] were proposed where an ABE system is able to revoke users from accessing encrypted data to which they used to have access in the system.

When using the ABE in a system where there is a large number of attributes, assessing the qualification of users and generating decryption keys by a central authority becomes impractical. *Multi-Authority Attribute-Based Encryption* (MA-ABE) was first proposed to address this issue in 2007 [54]. In a MA-ABE scheme, attributes are divided into different sets, and each set can be managed by an independent attribute authority. Corresponding attribute keys for decryption are issued by multiple attribute authorities, and encryptors can specify an access policy that requires a user to obtain decryption keys for appropriate attributes from different authorities in order to decrypt a message. Subsequently, several other MA-ABE constructions were proposed in [55, 56].

2) *Role-based access control (RBAC)*

Another access control model called *RBAC* [57, 58], has also been commonly adopted in traditional storage system in order to simplify management of permissions. Its access policy is determined based on different roles assigned to users by the system, while the data owner can specify a set of permissions of their data to different roles. By separation the tasks of role assignment and permission assignment, RBAC is much more efficient and scalable compared to other access control based on individual users, because the number of roles are usually significantly less than the number of users. Furthermore, it makes dynamic access control easier. For example, in applications where permissions for roles change slowly, while users may enter, leave, or change roles rapidly, the role manager can simply assign a new role to the user or revoke a role from the user. On the other hand, the data owner can also add permissions to a role or revoke permissions from a role. The authors of [59] suggested including RBAC in a new access control model for the health care system that can provide flexible access rights, because it can be modified dynamically while the task changed. However, one of the major criticisms of RBAC schemes is the complicated process when setting up the role structure. To make RBAC more efficient, roles can be structured hierarchically so that some roles inherit permissions from others.

To enforce RBAC policies, one approach is to transform the access control problem into a key management problem. In the literature, there exist many hierarchical access control schemes [60–62] which have been constructed based on hierarchical key management (HKM) schemes. Because of the similarity in structures between hierarchical access control and RBAC, a hierarchical access control scheme can be easily used to enforce RBAC access policies in cloud environment. In 2010, a role-based encryption (RBE) scheme [63] was built directly on RBAC policies. The security of the hierarchical access control scheme relies on the correct execution of the key assignment process, while the security of the RBE is based on the security of the cryptographic algorithm. More specifically, when a user is assigned to a role in RBE, a decryption key is calculated through a cryptographic algorithm by taking as input of the secret value and the identity of user and role. In the hierarchical access control scheme, the key for the user is generated based on the access control policies of the whole system. In 2011, Zhu *et al.* [64] proposed a revocable RBE scheme which allows users to be granted or revoked role memberships dynamically.

In the above schemes for enforcing RBAC policies, user membership of each role and role hierarchy are managed by a central authority. However in large-scale RBAC, systems which have hundred or even thousands of roles and hundreds of thousands of users and permissions, it is impractical to centralize the task of managing these users and permissions, and their relationships with the roles in a small team of security administrators. Zhou *et al.* [65] proposed a new RBE scheme using an *identity-based broadcast encryption* (IBBE) algorithm [66], which allows user memberships to be managed by individual roles. In the new RBE scheme, plaintext can be encrypted to a specified role, and only users in that role and its predecessor roles can decrypt the data with their role secrets and decryption keys. The employment of a broadcast encryption algorithm allows dynamically adding new users into a role without re-encryption, as well as revoking an existing user from a role without affecting any other existing users. In addition, this scheme has other features such as constant size keys and ciphertexts.

There have also been combined ABAC and RBAC schemes proposed in order to take advantage of both to provide effective access control for distributed and rapidly changing applications [67]. Hong *et al.* [68] implemented RBAC system for cloud storage via CP-ABE. In their work, permission assignments were handled by data owner while role assignments were handled by other users through propagation.

B) Searchable encryption

With more and more data moving to the cloud storage, it becomes imperative to enable search over the huge amount of data for many user applications. To preserve data confidentiality and integrity, it is necessary to store encrypted data in the cloud storage servers. To perform searching over data, the user has to either store an index locally, or download all the encrypted data, decrypt it and search locally. Neither approach is efficient when the data size grows in the cloud. When users seek to search and download relevant files from a cloud storage system, it is often desirable for the SSP to host search service, because it can minimize the network traffic and reduce management complexity for the users.Therefore, how to perform searching on encrypted databases without the need of decryption has become an increasingly fascinating topic in cloud storage systems. Recently, there have been new cryptographic primitives, called *searchable encryption* schemes [69, 70], proposed to address this problem.

The basic idea of searchable encryption schemes is to encrypt a search index generated over a collection of data in such a way that its contents are hidden without appropriate tokens, which can only be generated with a secrete key. Given a token for a keyword, one can retrieve pointers to the encrypted data files that contain the keyword. During the retrieval process, there is no contents of either the data files or the keyword revealed, other than the fact that all the retrieved data files contain one keyword in common.

Searchable encryption schemes, including *Symmetric Search Encryption* (SSE) [70], *Asymmetric Search Encryption* (ASE) [69] and other improvements on both schemes are reviewed in [71]. SSE employs symmetric cryptographic algorithms, such as block cipher or hash function, therefore is suitable when the party that performs search over the data are also the one who generates it, whereas ASE employs asymmetric cryptographic algorithms such as elliptic curve, thus is also suitable when the party that performs search over the data are different than the one who generates it. Therefore, ASE has wider applications than SSE in cloud storage than SSE. Meanwhile, compared to SSE schemes, ASE can achieve more complex search queries, such as conjunctions of terms, but at the cost of higher complexity and weaker security guarantees [72]. Efficient ASE, or ESE scheme was introduced in [73] to improve the efficiency when the keywords are hard to guess. However, it is more vulnerable to dictionary attacks.

Since SSE achieves higher efficiency and stronger security, it has been further developed recently. For example, dynamic SSE [74, 75] extended the inverted index approach [76] to allow update of the encrypted index and data files, and to achieve adaptive security against chosen-keyword attacks. Furthermore, Parallel and dynamic SSE [74] enables more efficient and scalable construction based on a keyword red-black tree-based multi-map data structure. On the other hand, SSE schemes with improved functionalities but compromised security have been proposed. Kuzu *et al.* [77] utilized locality sensitive hashing (LSH), which is widely used for fast similarity search in high-dimensional spaces for plain data, and proposed a search scheme to enable fast similarity search in the context of encrypted data. Another approach, which was proposed by Wang and Cao *et al.* [78] to secure ranked keyword search in encrypted cloud data. This method utilized the *Order-Preserving Symmetric Encryption* (OPSE) [79, 80], which achieves both security and privacy-preserving by protecting sensitive weighted information.

For cloud storage that are accessible with multiple users, how to enforce privileges and access control while searching through cloud storage has attracted researchers' attentions. One approach was proposed by Singh *et al.* [81] in 2009 which performs indexing in the trusted enterprise domain, and utilizes the resulting indices systematically with the *Access Control Barrel* (ACB) [82] primitives and concepts of user access hierarchy. This solution improves indexing efficiency and allows transferring the indices to the SSP for hosting, and it can be developed based on the integrity of search results returned by the SSP in the future.

Other than search algorithms on encrypted database, more general computation on encrypted database is a related topic. *Secure Computation ON an Encrypted Database* (SCONEDB) [83] was proposed to solve the *k-Nearest Neighbor* (kNN) computation in an encrypted database utilizing asymmetric scalar-product preserving encryption (ASPE). Besides, SCONEDB can incorporate other existing techniques, such as OPSE for the range query and homomorphic encryption for aggregate queries. CryptDB [84] implemented an integrated system that supports more general SQL query operations over encrypted database, by adapting a number of existing and new SQL-aware encryption primitives with different security properties and functionalities. CryptDB dynamically adjusts the encryption strategies using layered onion structures, where each data was dressed in increasingly stronger encryption, such that the outmost layer provides maximum security, whereas inner layers provide more functionality. A trusted proxy determines whether layers of encryption need to be removed when receiving a query from the user application.

C) Fully homomorphic encryption

Homomorphic encryption allows specific algebraic operations to be manipulated on a ciphertext, so it can produce the same encrypted result as the ciphertext of the result of the same (or different but known) operations performed on the plaintext. In other words, the operations to be performed on original data can now be performed on the encrypted ciphertext without knowing the original data. Homomorphic encryption can be categorized into two types: *partially homomorphic encryption* (PHE) and *FHE*. PHE allows only one homomorphic operation, either addition (e.g. Paillier [85]) or multiplication (e.g., unpadded RSA), while FHE supports both addition and multiplication operations. Since the original unpadded RSA algorithm published in 1977, there have been many PHE algorithms developed. However, the partially homomorphic property of an encryption algorithm has rarely been considered advantageous, but rather vulnerable to adaptive CCAs. Therefore, PHE algorithms have been found useful only in limited security applications such as electronic voting systems. On the other hand, since the first FHE algorithm was announced in 2009 [86], it has been recognized as a huge breakthrough in the computing security field. Practical application of FHE cryptosystems will potentially enable development of computing programs, which runs on encrypted input data to generate encrypted output. These programs can thus be run by untrusted entities without revealing any sensitive information during the computing process.

A homomorphic cryptosystem ε consists of four algorithms, $KeyGen_\varepsilon$, $Encrypt_\varepsilon$, $Decrypt_\varepsilon$, and an $Evaluate_\varepsilon$ algorithm. The first three algorithms are defined the same as those in any public-key cryptosystems. The $KeyGen_\varepsilon(\lambda)$ produces key-pair (pk, sk) given a security parameter λ. The $Encrypt_\varepsilon$ algorithm takes pk and a plaintext π as input, and it outputs a ciphertext ϕ. The $Decrypt_\varepsilon$ takes sk and ϕ as input, and outputs the plaintext π. In addition, the $Evaluate_\varepsilon$ algorithm takes as input pk, a circuit C from a permitted set C_ε, and a set of ciphertexts $\varphi = (\phi_1, \ldots \phi_t)$, consequently outputs a ciphertext ϕ. The homomorphic cryptosystem ε is correct for C_ε if for any key-pair (pk, sk) generated by $KeyGen_\varepsilon(\lambda)$, any circuit $C \in C_\varepsilon$, any plaintexts $\pi_1, \ldots, \pi_t$, and any ciphertexts $\varphi = (\phi_1, \ldots \phi_t)$ with $\phi_i \to Encrypt_\varepsilon(pk, \pi_i)$, it is the case that

$$If\ \phi \leftarrow Evaluate_\varepsilon(pk, C, \varphi),$$
$$then\ Decrypt_\varepsilon(sk, \phi) \to C(\pi_1, \ldots, \pi_t)$$

The computation complexity of all the above algorithms has to be polynomial in the size of C and security level parameter λ, which is defined as all known attacks against the scheme take time at least 2^{λ}. ε is fully homomorphic if it is homomorphic for all circuits [86].

A family of schemes $\varepsilon(d) : d \in Z^{+}$ is leveled fully homomorphic if they all use the same decryption circuit, $\varepsilon(d)$ is homomorphic for all circuits of depth at most d (that use some specified set of gates Γ), and the computational complexity of $\varepsilon(d)$'s algorithms is polynomial in λ, d, and (in the case of $Evaluate_{\varepsilon(d)}$) the size of C.

The first FHE scheme proposed by Craig Gentry in 2009 [86] applies lattice-based cryptography to construct the scheme, where lattice L was a set of points in the n-dimensional Euclidean space R^n with a strong periodicity property. The proposed scheme started from a somewhat homomorphic encryption scheme using ideal lattices, which is limited to "low-degree" polynomials evaluation on encrypted data due to the augment of noise in the ciphertext during evaluation. After this "*initial construction*" stage, a "*squash the decryption circuit*" technique was used to modify the scheme to make it "*bootstrappable*". The modified encryption scheme can evaluate its own decryption circuit, and effectively refresh the ciphertext to reduce the augmented noises, which eliminates the limitation on the depth of circuit evaluated over the ciphertext. In short, Craig Gentry slightly modified somewhat homomorphic encryption by recursive self-embedding. The resulting scheme can reduce the accumulated noise caused by multiple algebraic operations, thus make it possible to realize FHE in arbitrary depth.

However, this first FHE scheme is impractical since the computation complexity and ciphertext size are high-order polynomials in the security level parameter λ, which means they increase sharply in order to achieve a practically high-enough security level. This prohibit the practical application of the FHE, especially in the cloud computing context where high security level is crucial. Another major concern of this scheme is that its security was based on two relatively new assumptions, namely, the hardness of the worst-case *bounded distance decoding problem* (BDD) on ideal lattice, and the hardness of the average-case *sparse subset sum problem* (SSSP) of the squashing step. Both are relatively untested cryptographic assumptions.

More recently, there have been growing research efforts made in searching practical FHE algorithms, which are more efficient and/or based on more reliable security assumptions. A second version of FHE scheme, known as DGHV, was proposed by Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan in 2010 [87]. DGHV uses Gentry's techniques with only elementary modular arithmetic over integers to convert a simple somewhat homomorphic encryption scheme to a bootstappable FHE scheme. This scheme achieved conceptual simplicity because all computations were performed over integers instead of ideal lattice. It also reduced the security assumption to the hardness of the greatest common divisor (GCD) problem. However, the price of this tradeoff is the immense size of public key, which can be impractical for the current systems. Stehle and Steinfeld [88] presented a faster homomorphic encryption in order to improve Gentry's scheme by a more aggressive analysis of the SSSP assumption, and introducing a probabilistic decryption algorithm implemented by an algebraic circuit of low multiplicative degree. With these two enhancements, this scheme obtains $O(\lambda^{3.5})$ bit complexity for refreshing a cipher text, whereas previous scheme claimed $O(\lambda^{6})$ for the same task, where λ is the security parameter. However, there is a non-zero probability of decryption error associated with this scheme.

Besides, Zvika Brakerski and Vinod Vaikuntanathan gave another improvement on Gentry's scheme [89] by changing the two security assumptions made in [86]. First, the somewhat homomorphic encryption was based on ring learning with errors (RLWE) assumption from Lyubashevsky, Peikert and Regev [90] instead of the ideal lattices BDD problem. Second, to make the somewhat homomorphic encryption scheme bootstrappalbe, it used a dimension-modulus reduction technique instead of Gentry's squashing technique, thus eliminating the assumption of SSSP. This new bootstrapping technique also shortened the ciphertext and reduced the complexity. Based on the above improvement, Brakerski, Gentry, and Vaikuntanathan worked together to propose a new leveled FHE scheme without Gentry's bootstrapping procedure in 2011 [91]. By applying RLWE, this FHE scheme has $O(\lambda \cdot L^{3})$ per-gate computation for L-level arithmetic circuits. As an optional approach, they also proposed a leveled FHE scheme using bootstrapping as optimization to further reduce the per-gate computation down to $O(\lambda^{2})$, independent of L.

Following up in 2011, Coron *et al.* proposed an improvement of FHE over the integers described by van Dijk *et al.* The proposed new scheme shortened the public key size from $O(\lambda^{10})$ to $O(\lambda^{7})$ [92]. This procedure is done by using quadratic form instead of linear one in the public key elements, so that the full-length public key is compressed to a smaller subset of the original key. Instead of proposing any further improvement on FHE, Ron Rothblum manifests how to transform any additively homomorphic private-key encryption scheme into a public-key encryption scheme [93]. To construct this process, this scheme develops a theorem that any compact additively homomorphic with respect to addition modulo two can be transformed into a semantically scheme. In consequence, the public-key encryption scheme save one hop homomorphic with regard to the same set operations with private-key encryption, which are prior FHE schemes.

With all the theoretical development of different FHE algorithms, it is necessary to investigate their practical implementation. There were several implementations of Gentry's FHE in 2010, and the first attempt was made by Smart and Vercauteren [94]. They were able to implement the somewhat homomorphic scheme using "*principle-ideal lattices*" of prime determinant, which can be implied by two integers only. However, they were not able to implement the bootstrapping functionality to obtain a fully homomorphic scheme. Bottleneck of this implementation was the failure to support a large amount of parameters.

Based on this work, in 2011, Gentry and Halevi developed a series of simplifications and optimizations that made bootstrapping implementation possible. As the result, the asymptotic complexity is reduced from Smart and Vercauteren's $O(n^2.5)$ to $O(n^1.5)$. The optimizations from this paper were also used in [92] in order to implement the fully homomorphic DGHV scheme under new variant. With the result of having similar performance, Coron *et al.* successfully showed that FHE can be implemented with a simple arithmetic scheme.

In Table 2, we provide a comparison of performance for different FHE schemes. BDD and SSSP are the problems that stated in the first FHE scheme.

D) Other data confidentiality approaches

There are several other data confidentiality methods beside above. For instance, The application of cryptographic algorithms to data blocks in the cloud storage is a popular method used to ensure the confidentiality of stored data. A data confidentiality scheme in coreFS, which is a user-level network file system, was proposed in 2009 [96]. This scheme is constructed based on a new universal-hash stateful MAC. It has smaller computational overhead of cryptographic operations comparing to the MHT. Besides, it allows better communication capability. However, the choice of caching strategy, MAC tree update schedule, and the method to store the tree can affect the performance of this scheme.

Another data confidentiality scheme exploited the newly proposed *secure provenance* (SP) model based on the bilinear pairing techniques in 2010 [97]. This scheme records the ownership and the process history of data objects in the cloud storage in order to increase the trust from public users. The SP model consists of the following modules: system setup, key generation, anonymous authentication, authorized access, and provenance tracking. The provable security technique has been tested on this scheme under the standard SP model. It demands some practical considerations in real-world applications and further improvement under the current framework.

Different from above schemes, an compelling statement was proposed by Dijk and Juel in 2010 [98], which claimed that no cryptographic protocol, even including power primitives such as FHE, can enforce privacy requested by common cloud services alone. This paper also demonstrated that above demand can be achieved by other enforcements instead, such as tamperproof hardware, distributed computing, and complex trust ecosystems.

E) Summary

Data confidentiality is one of the most critical issues for applications with sensitive data, such as personal information, customer's account information, financial and healthcare information. The new challenge of storing those data in clouds is how to prevent accidental or intentional data leakage to the cloud SSP, such that even if the service provider is compromised, the information can still be kept confidential. The fundamental solution to this problem is still data encryption. The user has to encrypt the data before they are moved to the cloud server, and keep it encrypted for the entire period during which the data are in the cloud. When the data needs to be accessed or processed by either the data owner or other legitimate users who have the key for decryption, it is not efficient to retrieve the encrypted data, decrypt it, process and re-encrypt it before sending back to the server. Therefore, new encryption mechanisms that allow for processing of the ciphertext directly without revealing the original information in the plaintext will have a significant potential in cloud storage of sensitive data. WIth encryption, data owners or users regain their control over their data that are not physically stored by themselves.

FHE is an ideal example of these encryption algorithms. However, the promising applications of current FHE algorithms are hindered by its computation complexity and other implementation difficulties. Improvements must be made before it can be put in practical applications. In addition, more implementations of various improvements are awaited to be evaluated on current platforms.

Unlike FHE, which has an ambitious aim at arbitrary computing on the ciphertext, other cloud encryption schemes aimed at specific type of control over encrypted data. For example, ABE allows access control being enforced on the encrypted data by incorporating attribute-based access structure into either the ciphertext or the decryption key. Searchable encryption schemes provide a way to search the ciphertext for a keyword token without revealing the real content of either data or the keyword.

V. AVAILABILITY

As a different security measure, availability in cloud storage refers to that the data are accessible and usable when authorized users request them from any machine at any time. In an earlier stage of cloud computing, availability was of more security concern due to the lack of mature and reliable infrastructure. Many incidents of service unavailability occurred due to hardware failure and resulted in severe consequences. With better and more reliable infrastructures in place, the challenge facing the availability of cloud storage service is how to preserve the user's data in case of emergency, such as a natural disaster.

The most straightforward solution is to keep backup copies of data in multiple physical locations. Amazon EC2 and S3 provide a perfect example based on availability zone, which locates within divided geographic regions, for example, US-West and US-East. Each region contains several instances with same data. When accident occurs, Amazon EC2 and S3 can easily recover damaged or lose data from other availability zones within the same region to save power and time. However, this approach is not efficient in terms of storage resource utilization.

There have been backup storage management schemes, such as incremental backup and data deduplication,

Table 2. Performance comparison for FHE schemes

	Solution on ideal lattices BDD	Solution on SSSP	Per-gate comp.	Public key size	Asymptotic comp.
2009 Gentry [86]	SVP	Availability of SVP Oracle	$O(\lambda^6)$		
2010 Stehle [88]		Refined Analysis	$O(\lambda^3)$		
2011 Brakerski [89]	RLWE		$O(\lambda^3)$		
2010 Dijk [87]	Replace Ideal Lattice	Choosing Large Enough θ	$O(\lambda^3)$	$O(\lambda^{10})$	
2011 Coron [92]		Refined Analysis	$O(\lambda^3)$	$O(\lambda^7)$	
2010 Vercauteren [94]			$O(\lambda^3)$		$O(n^{2.5})$
2011 Halevi [95]			$O(\lambda^3)$		$O(n^{1.5})$

developed to improve the storage utilization. Incremental backup has been used widely in file backup services. It exploits the correlation between current files with previous backup version and only stores the differences. When incremental backup being deployed in a data block level or even data byte level, it becomes more efficient in storage utilization, but with higher processing overhead. Delta encoding is a famous incremental backup example applied by Dropbox. Data deduplication is a specialized data compression technique that identifies common data chunks within and across different files, and stores them only once to improve storage utilization. Unfortunately, data deduplication potentially undermining the data security in terms of both data integrity and data confidentiality. First, by definition, data deduplication alters the original data from the user and stores them in a different form in the cloud storage, thus results in concerns of data integrity. Second, data deduplication attempts to identify and exploit identical data chunks, while encryption algorithms usually try to randomize them to conceal the real contents. The encrypted ciphertext for the same plaintext will likely to be extremely different. To address these issues, efficient and secure data deduplication which allows data deduplication performed on encrypted ciphertext have been developed [99]. This technique utilized *convergent encryption*, in which the encryption key is generated using a hash function of the plaintext of the data chunk. Therefore, the same plaintext data chunk will be encrypted using the same key, no matter when and by whom it is encrypted. This results in the same ciphertext data chunk for the same plaintext. The scheme stores unique chunks of data or bytes during data analysis, and then compares other chunks to the stored data. If the compared result is matched, then the redundant part is replaced by a small pointer pointing to the location of the matched stored data.

Another proactive approach is to predict future availability failure occurrences so that actions could be taken earlier to avoid interruption of service. Guan *et al.* proposed two learning approaches to predict failure dynamics in cloud computing systems by using Bayesian methods and decision trees [100]. An initial stage is required for monitoring data, and then an ensemble Bayesian methods labels data that have anomalous behaviors. After all the anomalies are identified, the model can predict future failure occurrences based on decision tree classifiers.

Once the failure has occurred, data recovery schemes are necessary to reduce or eliminate the loss. Zhang *et al.* [101] presented a data recovery method that examines the damage in a fine-grained cloud database and allows the cloud database owner to know and locate the damage precisely for the recovery purpose. Information dispersal algorithm [102] is used to enable greater availability of data when encountering physical failures and network outages.

Besides the above techniques, data recovery can also be achieved by new service framework. Chi-won Song *et al.* proposed *Parity Cloud Service* (PCS) in 2011 [103]. It generates virtual disk in user system for private backup and makes parity group of multiple users. The same data among those users in the parity group are stored at the server-end. Therefore, when users find out that original file requires recovery, they can request data from the server-end without violating privacy since private backup is stored at each user's virtual disk. This approach is simple and secure, but each user has to build up virtual disk, which costs additional overhead for users.

In summary, ensuring availability of users' data whenever users demands it is the basic and primary requirement in cloud storage. The main challenge arises when taking other performance and security concerns into consideration. Trade-offs between efficiency and reliability have to be made to balance the interests of service provider and the user. Furthermore, data integrity and data confidentiality should not be compromised by improved availability.

VI. CONCLUSION AND FUTURE WORK

With the trend of rapid deployment of cloud storage and computing nowadays, it is essential for the cloud storage systems to be equipped with security solutions proven to be reliable and trustworthy. In this work, we conducted a survey on most recently developed or proposed primitives to ensure three of the most critical security measurements, namely, data integrity, data confidentiality, and availability, for the cloud storage systems. For each aspect, we identified the unique challenges that are different from those in traditional data network or file storage systems, summarized the existing development progress up to date, and provided insight into the future directions of research. Overall, we feel that the cloud storage security is still in its infancy and expect to see more salient breakthrough in the near future. For example, although the cloud storage security solutions have been developed rapidly in recent years, we have not yet seen a widely accepted model for the implementation. Besides the system design, the cloud storage security system should be flexible enough so that it can be improved by new cryptographic algorithms.

ACKNOWLEDGEMENTS

This work is supported by the Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No. BM2003201, and the Key Laboratory of Information Network Security, the Ministry of Public Security.

REFERENCES

[1] Chantry, D.: Mapping applications to the cloud. Technical Report, January 2009.

[2] Guttman, B.; Roback, E.A.: Sp 800-12. an introduction to computer security: the NIST handbook. Technical Report, Gaithersburg, MD, USA, 1995.

[3] Mell, P.; Grance, T.: The NIST definition of cloud computing. Technical Report, July 2009.

[4] Tim Jones, M.: Anatomy of a cloud storage infrastructure. Technical Report, IBM, 2010.

[5] Zeng, W.; Zhao, Y.; Ou, K.; Song, W.: Research on cloud storage architecture and key technologies. In *Proc. 2nd Int. Conf. on Interaction Sciences: Information Technology, Culture and Human, ICIS '09*, New York, NY, USA, 2009, 1044–1048, ACM.

[6] CCITT Recommendation X.800. Security architecture for open systems interconnection for CCITT applications. Technical Report, March 1991.

[7] Paul, M.; Saxena, A.: Proof of erasability for ensuring comprehensive data deletion in cloud computing. In *Recent Trends in Network Security and Applications*, volume 89 of *Communications in Computer and Information Science*, Springer–Berlin–Heidelberg, 2010, 340–348.

[8] Perito, D.; Tsudik, G.: Secure code update for embedded devices via proofs of secure erasure. In *Proc. 15th European Conf. on Research in Computer Security, ESORICS'10*, Berlin, Heidelberg, 2010, 643–662, Springer-Verlag.

[9] Juels, A.; Kaliski, B.S. Jr.: Pors: proofs of retrievability for large files. In *Proc. 14th ACM Conf. on Computer and Communications Security, CCS '07*, New York, NY, USA, 2007, 584–597.

[10] Lillibridge, M.; Elnikety, S.; Birrell, A.; Burrows, M.; Isard, M.: A cooperative Internet backup scheme. In *Proc. USENIX Annual Technical Conf., ATEC '03*, Berkeley, CA, USA, 2003, 3–3, USENIX Association.

[11] Naor, M.; Rothblum, G.: The complexity of online memory checking. Cryptology ePrint Archive, Report 2006/091, 2006.

[12] Ateniese, G. *et al.*: Provable data possession at untrusted stores. In *Proc. 14th ACM Conf. on Computer and Communications Security, CCS '07*, New York, NY, USA, 2007, 598–609.

[13] Johnson, R.; Molnar, D.; Song, D.; Wagner, D.: Homomorphic signature schemes. In *Topics in Cryptology CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2002, 204–245.

[14] Shacham, H.; Waters, B.: Compact proofs of retrievability. In *Proc. 14th Int. Conf. on Theory and Application of Cryptology and Information Security: Advances in Cryptology, ASIACRYPT '08*, Berlin, Heidelberg, 2008, 90–107, Springer-Verlag.

[15] Ateniese, G.; Di Pietro, R.; Mancini, L.V.; Tsudik, G.: Scalable and efficient provable data possession. In *Proc. 4th Int. Conf. on Security and Privacy in Communication Networks, SecureComm '08*, New York, NY, USA, 2008, 9:1–9:10, ACM.

[16] Boneh, D.; Lynn, B.; Shacham, H.: Short signatures from the weil pairing. In *Advances in Cryptology ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2001, 514–532.

[17] Wang, Q.; Wang, C.; Li, J.; Ren, K.; Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In *Proc. 14th Eur. Conf. on Research in Computer Security, ESORICS'09*, Berlin, Heidelberg, 2009, 355–370, Springer-Verlag.

[18] Wang, C.; Wang, Q.; Ren, K.; Lou, W.: Ensuring data storage security in cloud computing. In *17th Int. Workshop on Quality of Service, IWQoS 2009*, July 2009, 1 –9.

[19] Wang, C.; Chow, S.S.M.; Wang, Q.; Ren, K.; Lou, W.: Privacy-preserving public auditing for secure cloud storage. IEEE Trans. Comput., **62** (2) (2013), 362–375.

[20] Wang, C.; Ren, K.; Lou, W.; Li, J.: Toward publicly auditable secure cloud data storage services. IEEE Netw., **24** (4) (2010), 19 –24.

[21] Wang, C.; Wang, Q.; Ren, K.; Cao, N.; Lou, W.: Toward secure and dependable storage services in cloud computing. IEEE Trans. Serv. Comput., **5** (2) (2012), 220–232.

[22] Wang, C.; Wang, Q.; Ren, K.; Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In *Proc. 29th Conf. on Information Communications, INFOCOM'10*, Piscataway, NJ, USA, 2010, 525–533, IEEE Press.

[23] 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) (2011), 847–859.

[24] Zhu, Y.; Wang, H.; Hu, Z.; Ahn, G.-J.; Hu, H.; Yau, S.S.: Dynamic audit services for integrity verification of outsourced storages in clouds. In *Proc. 2011 ACM Symp. on Applied Computing, SAC '11*, New York, NY, USA, 2011, 1550–1557.

[25] Han, S.; Xing, J.: Ensuring data storage security through a novel third party auditor scheme in cloud computing. In *2011 IEEE Int. Conf. on Cloud Computing and Intelligence Systems (CCIS)*, September 2011, 264–268.

[26] Hao, Z.; Zhong, S.; Yu, N.: A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. IEEE Trans. Knowl. Data Eng., **23** (9) (2011), 1432–1437.

[27] Sebé, F.; Domingo-Ferrer, J.; Martinez-Balleste, A.; Deswarte, Y.; Quisquater, J.-J.: Efficient remote data possession checking in critical information infrastructures. IEEE Trans. Knowl. Data Eng., **20** (2008), 1034–1038.

[28] Curtmola, R.; Khan, O.; Burns, R.: Robust remote data checking. In *Proc. 4th ACM Int. Workshop on Storage Security and Survivability, StorageSS '08*, New York, NY, USA, 2008, 63–68.

[29] Dodis, Y.; Vadhan, S.; Wichs, D.: Proofs of retrievability via hardness amplification. In *Proc. 6th Theory of Cryptography Conf. on Theory of Cryptography, TCC '09*, Berlin, Heidelberg, 2009, 109–127, Springer-Verlag.

[30] Bowers, K.D.; Juels, A.; Oprea, A.: Proofs of retrievability: theory and implementation. In *Proc. 2009 ACM Workshop on Cloud Computing Security, CCSW '09*, New York, NY, USA, 2009, 43–54.

[31] Shacham, H.; Waters, B.: Compact proofs of retrievability. J. Cryptol., **26** (3) (2013), 442–83.

[32] Sravan Kumar, R.; Saxena, A.: Data integrity proofs in cloud storage. In *2011 3rd Int. Conf. on Communication Systems and Networks (COMSNETS),* January 2011, 1–4.

[33] Ateniese, G. *et al.*: Remote data checking using provable data possession. ACM Trans. Inf. Syst. Secur., **14** (1) (2011), 12:1–12:34.

[34] Erway, C.; Küpçü, A.; Papamanthou, C.; Tamassia, R.: Dynamic provable data possession. In *Proc. 16th ACM Conf. on Computer and Communications Security, CCS '09*, New York, NY, USA, 2009, 213–222.

[35] Papamanthou, C.; Tamassia, R.; Triandopoulos, N.: Authenticated hash tables. In *Proc. 15th ACM Conf. on Computer and Communications Security, CCS '08*, New York, NY, USA, 2008, 437–448.

[36] Merkle, R.C.: *Protocols for Public Key Cryptosystems*, IEEE Computer Society Press, 1980, 122–134.

[37] Zheng, Q.; Xu, S.: Fair and dynamic proofs of retrievability. In *Proc. of the first ACM Conference on Data and Application Security and Privacy, CODASPY '11*, New York, NY, USA, 2011, 237–248.

[38] Wang, C.; Cao, N.; Li, J.; Ren, K.; Lou, W.: Secure ranked keyword search over encrypted cloud data. In *2010 IEEE 30th Int. Conf. on Distributed Computing Systems (ICDCS),* June 2010, 253 –262.

[39] Kallahalla, M.; Riedel, E.; Swaminathan, R.; Wang, Q.; Fu, K.: Plutus: Scalable secure file sharing on untrusted storage. In *Proc. 2nd USENIX Conf. on File and Storage Technologies*, Berkeley, CA, USA, 2003, 29–42, USENIX Association.

[40] Goh, E.j.; Shacham, H.; Modadugu, N.; Boneh, D.: Sirius: Securing remote untrusted storage. In *Proc. Network and Distributed Systems Security (NDSS) Symp. 2003*, 2003, 131–145.

[41] Sahai, A.; Waters, B.: Fuzzy identity-based encryption. In *Advances in Cryptology – EUROCRYPT 2005, 24th Annu. Int. Conf. on the Theory and Applications of Cryptographic Techniques*, Aarhus, Denmark, May 22–26, volume 3494 of *Lecture Notes in Computer Science*, Springer, 2005, 457–473.

[42] Goyal, V.; Pandey, O.; Sahai, A.; Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. 13th ACM Conf. on Computer and Communications Security, CCS '06*, New York, NY, USA, 2006, 89–98.

[43] Bethencourt, J.; Sahai, A.; Waters, B.: Ciphertext-policy attribute-based encryption. In *Proc. 2007 IEEE Symp. on Security and Privacy*, SP '07, Washington, DC, USA, 2007, 321–334.

[44] Ostrovsky, R.; Sahai, A.; Waters, B.: Attribute-based encryption with non-monotonic access structures. In *Proc. 14th ACM Conf. on Computer and Communications Security, CCS '07*, New York, NY, USA, 2007, 195–203.

[45] Cheung, L.; Newport, C.: Provably secure ciphertext policy abe. In *Proc. 14th ACM Conf. on Computer and Communications Security, CCS '07*, New York, NY, USA, 2007, 456–465.

[46] Lewko, A.B.; Okamoto, T.; Sahai, A.; Takashima, K.; Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, 2010, 62–91.

[47] Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In *Public Key Cryptography PKC 2010*, volume 6571 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2011, 53–70.

[48] Goyal, V.; Jain, A.; Pandey, O.; Sahai, A.: Bounded ciphertext policy attribute based encryption. In *35th Int. Colloq. Automata, Languages and Programming, 2008*, volume 5126 of *Lecture Notes in Computer Science*, Springer, 2008, 579–591.

[49] Malek, B.; Miri, A.: Combining attribute-based and access systems. In *Int. Conf. on Computational Science and Engineering, 2009. CSE '09*, volume 3, aug. 2009, 305–312.

[50] Yu, S.; Wang, C.; Ren, K.; Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM, 2010 Proc. IEEE*, March 2010, 1–9.

[51] Zhao, F.; Nishide, T.; Sakurai, K.: Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems. In *Proc. 7th Int. Conference on Information Security Practice and Experience, ISPEC'11*, Berlin, Heidelberg, 2011, 83–97, Springer-Verlag.

[52] Sahai, A.; Seyalioglu, H.; Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, Springer, 2012, 199–217.

[53] Zhang, F.; Li, Q.; Xiong, H.: Efficient revocable key-policy attribute based encryption with full security. In *IEEE 8th Int. Conf. on Computational Intelligence and Security 2012*, 2012, 477–481.

[54] Chase, M.: Multi-authority attribute based encryption. In *4th Theory of Cryptography Conf.*, volume 4392 of *Lecture Notes in Computer Science*, Springer, 2007, 515–534.

[55] Chase, M.; Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In *Proc. 2009 ACM Conf. on Computer and Communications Security*, 2009, 121–130.

[56] Lewko, A.B.; Waters, B.: Decentralizing attribute-based encryption. In *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, Springer, 2011, 568–588.

[57] Ahn, G.-J.; Sandhu, R.: Role-based authorization constraints specification. ACM Trans. Inf. Syst. Secur., **3** (2000), 207–226.

[58] Sandhu, R.S.; Coyne, E.J.; Feinstein, H.L.; Youman, C.E.: Role-based access control models. Computer, **29** (2) (1996), 38–47.

[59] Narayanan, H.A.J.; Gunes, M.H.: Ensuring access control in cloud provisioned healthcare systems. In *2011 IEEE Consumer Communications and Networking Conf. (CCNC)*, January 2011, 247–251.

[60] Atallah, M.J.; Blanton, M.; Fazio, N.; Frikken, K.B.: Dynamic and efficient key management for access hierarchies. ACM Trans. Inf. Syst. Secur., **12** (3) (2009), 18:1–18:43.

[61] De Capitani di Vimercati, S.; Foresti, S.; Jajodia, S.; Paraboschi, S.; Samarati, P.: Encryption policies for regulating access to outsourced data. ACM Trans. Database Syst., **35** (2) (2010), 12:1–12:46.

[62] Samarati, P.; De Capitani di Vimercati, S.: Data protection in outsourcing scenarios: issues and directions. In *Proc. 5th ACM Symp. on Information, Computer and Communications Security, 2010*, 2010, 1–14.

[63] Zhu, Y.; Ahn, G.-J.; Hu, H.; Wang, H.: Cryptographic role-based security mechanisms based on role-key hierarchy. In *Proc. 5th ACM Symp. on Information, Computer and Communications Security, 2010*, 2010, 314–319.

[64] Zhu, Y.; Hu, H.; Ahn, G.-J.; Wang, H.; Wang, S.-B.: Provably secure role-based encryption with revocation mechanism. J. Comput. Sci. Technol., **26** (4) (2011), 697–710.

[65] Zhou, L.; Varadharajan, V.; Hitchens, M.: Enforcing role-based access control for secure data storage in the cloud. Comput. J., **54** (10) (2011), 1675–1687.

[66] Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In *Proc. Advances in Crypotology*

13th Int. Conf. on Theory and application of cryptology and information security, ASIACRYPT'07, Berlin, Heidelberg, 2007, 200–215, Springer-Verlag.

[67] Richard Kuhn, D.; Coyne, E.J.; Weil, T.R.: Adding attributes to role-based access control. Computer, **43** (6) (2010), 79–81.

[68] Hong, C.; lv, Z.; Zhang, M.; Feng, D.: A secure and efficient role-based access policy towards cryptographic cloud storage. In *Proc. 12th Int. Conf. on Web-age Information Management*, WAIM'11, Berlin, Heidelberg, 2011, 264–276, Springer-Verlag.

[69] Boneh, D.; Di Crescenzo, G.; Ostrovsky, R.; Persiano, G.: Public key encryption with keyword search. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2004, 506–522.

[70] Song, D.X.; Wagner, D.; Perrig, A.: Practical techniques for searches on encrypted data. In *2000 IEEE Symp. Security and Privacy, 2000, SP 2000, Proc.*, 2000, 44–55.

[71] Kamara, S.; Lauter, K.: Cryptographic cloud storage. In *Financial Cryptography and Data Security*, volume 6054 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2010, 136–149.

[72] Abdalla, M. *et al.*: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. J. Cryptol., **21** (3) (2008), 350–391.

[73] Bellare, M.; Boldyreva, A.; O'Neill, A.: Deterministic and efficiently searchable encryption. In *Proc. of the 27th Annu. Int. Cryptology Conf. on Advances in Cryptology, CRYPTO'07*, Berlin, Heidelberg, 2007, 535–552, Springer-Verlag.

[74] Kamara, S.; Papamanthou, C.: Parallel and dynamic searchable symmetric encryption. In *Financial Cryptography*, 2013, 258–274.

[75] Kamara, S.; Papamanthou, C.; Roeder, T.: Dynamic searchable symmetric encryption. In *ACM Conf. on Computer and Communications Security*, 2012, 965–976.

[76] Curtmola, R.; Garay, J.; Kamara, S.; Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions. J. Comput. Secur., **19** (5) (2011), 895–934.

[77] Kuzu, M.; Islam, M.S.; Kantarcioglu, M.: Efficient similarity search over encrypted data. In *Proc. 2012 IEEE 28th Int. Conf. on Data Engineering, ICDE '12*, 2012, 1156–1167.

[78] Wang, C.; Cao, N.; Ren, K.; Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans. Parallel Distrib. Syst., **23** (8) (2012), 1467–1479.

[79] Boldyreva, A.; Chenette, N.; Lee, Y.; O'Neill, A.: Order-preserving symmetric encryption. In *Proc. 28th Annu. Int. Conf. on Advances in Cryptology: the Theory and Applications of Cryptographic Techniques, EUROCRYPT '09*, Berlin, Heidelberg, 2009, 224–241, Springer-Verlag.

[80] Boldyreva, A.; Chenette, N.; O'Neill, A.: Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Proc. 31st Annual Conf. on Advances in Cryptology, CRYPTO'11*, Berlin, Heidelberg, 2011, 578–595, Springer-Verlag.

[81] Singh, A.; Srivatsa, M.; Liu, L.: Search-as-a-service: Outsourced search over outsourced storage. ACM Trans. Web, **3** (2009), 13:1–13:33.

[82] Singh, A.; Srivatsa, M.; Liu, L.: Efficient and secure search of enterprise file systems. In *IEEE Int. Conf. on Web Services, 2007. ICWS 2007*, July 2007, 18 –25.

[83] Wong, W.K.; Wai-lok Cheung, D.; Kao, B.; Mamoulis, N.: Secure KNN computation on encrypted databases. In *Proc. 35th SIGMOD Int. Conf. on Management of data, SIGMOD '09*, New York, NY, USA, 2009, 139–152, ACM.

[84] Popa, R.A.; Redfield, C.M.S.; Zeldovich, N.; Balakrishnan, H.: Cryptdb: processing queries on an encrypted database. Commun. ACM, **55** (9) (2012), 103–111.

[85] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology EUROCRYPT 99*, volume 1592 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 1999, 223–238.

[86] Gentry, C.: Fully homomorphic encryption using ideal lattices. In *Proc. 41st Annu. ACM Symp. on Theory of Computing, STOC '09*, New York, NY, USA, 2009, 169–178.

[87] Van Dijk, M.; Gentry, C.; Halevi, S.; Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2010, 24–43.

[88] Stehle, D.; Steinfeld, R.: Faster fully homomorphic encryption. In *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2010, 377–394.

[89] Brakerski, Z.; Vaikuntanathan, V.: Fully homomorphic encryption from ring-lWE and security for key dependent messages. In *Proc. 31st Annu. Conf. on Advances in Cryptology, CRYPTO'11*, Berlin, Heidelberg, 2011, 505–524, Springer-Verlag.

[90] Lyubashevsky, V.; Peikert, C.; Regev, O.: On ideal lattices and learning with errors over rings. J. ACM, **60** (6) (2013), 43:1–43:35.

[91] Brakerski, Z.; Gentry, C.; Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011.

[92] Coron, J.-S.; Mandal, A.; Naccache, D.; Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In *Proc. 31st Annu. Conf. on Advances in Cryptology*, CRYPTO'11, Berlin, Heidelberg, 2011, 487–504, Springer-Verlag.

[93] Rothblum, R.: Homomorphic encryption: From private-key to public-key. In *Theory of Cryptography*, volume 6597 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2011, 219–234.

[94] Smart, N.; Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2010, 420–443.

[95] Gentry, C.; Halevi, S.: Implementing gentry's fully-homomorphic encryption scheme. In *Proc. 30th Annu. Int. Conf. on Theory and Applications of Cryptographic Techniques: Advances in Cryptology, EUROCRYPT'11*, Berlin, Heidelberg, 2011, 129–148, Springer-Verlag.

[96] Yun, A.; Shi, C.; Kim, Y.: On protecting integrity and confidentiality of cryptographic file system for outsourced storage. In *Proc. 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, New York, NY, USA, 2009, 67–76.

[97] Lu, R.; Lin, X.; Liang, X.; Shen, X.S.: Secure Provenance: the Essential of Bread and Butter of Data Forensics in Cloud Computing, *ACM*, New York, 2010, 282–292.

[98] Van Dijk, M.; Juels, A.: On the impossibility of cryptography alone for privacy-preserving cloud computing. In *Proc. 5th USENIX Conf. on Hot Topics in Security, HotSec'10*, Berkeley, CA, USA, 2010. USENIX Association, 1–8.

[99] Storer, M.W.; Greenan, K.; Long, D.D.E.; Miller, E.L.: Secure data deduplication. In *Proc. 4th ACM Int. Workshop on Storage Security and Survivability, StorageSS '08*, New York, NY, USA, 2008, 1–10.

[100] Guan, Q.; Zhang, Z.; Fu, S.: Proactive failure management by integrated unsupervised and semi-supervised learning for dependable

cloud systems. In *Proc. 2011 6th Int. Conf. on Availability, Reliability and Security, ARES '11*, Washington, DC, USA, 2011, 83–90.

[101] Zhang, M.; Cai, K.; Feng, D.: Fine-grained cloud db damage examination based on bloom filters. In *Proc. 11th Int. Conf. on Web-age Information Management, WAIM'10*, Berlin, Heidelberg, 2010, 157–168, Springer-Verlag.

[102] Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. J. ACM, **36** (2) (1989), 335–348.

[103] won Song, C.; Park, S.; wook Kim, D.; Kang, S.: Parity cloud service: A privacy-protected personal data recovery service. In *2011 IEEE 10th Int. Conf. Trust, Security and Privacy in Computing and Communications (TrustCom)*, November 2011, 812 –817.