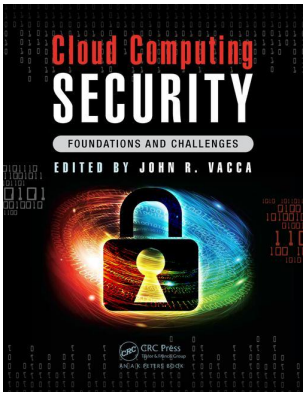

Cloud Computing Security
Foundations and Challenges

John R. Vacca

Cloud Security Baselines

Publication details
https://www.routledgehandbooks.com/doi/10.1201/9781315372112-5
Daniela Oliveira, Anna Squicciarini, Dan Lin
Published online on: 09 Sep 2016

Cloud Security Baselines

Daniela Oliveira

University of Florida
Gainesville, Florida

Anna Squicciarini

Pennsylvania State University
University Park, Pennsylvania

Dan Lin

Missouri University of Science and Technology
Rolla, Missouri

CONTENTS

3.1 INTRODUCTION

This chapter discusses the essentials of cloud computing security, one of the main challenges of the field. It starts with an overview of computer security, discussing its three pillars—confidentiality, integrity, and availability—and other important concepts such as authenticity and non-repudiation. The chapter also discusses the concepts of vulnerabilities, threats, and attacks in general, and, in the context of cloud computing, is followed by a review of the most common mitigations for cloud computing threats. This chapter also considers privacy and security in cloud storage services and multiclouds and cloud accountability. The chapter concludes with a summary and a discussion of research challenges.

3.2 AN OVERVIEW OF COMPUTER SECURITY

Computer security comprises three main pillars, which are well-known by the CIA acronym: confidentiality, integrity, and availability. Confidentiality involves the concealment of sensitive information from unauthorized parties. There are three mechanisms that help enforce confidentiality. The first is cryptography, which conceals plain text information using mathematical transformations. The second is access control, which defines the parties permitted access to certain parts of the system or certain pieces of information. The third is authorization, which determines what actions each authorized party is allowed to do with a piece of data or a system module [1–3].

The integrity pillar means that a system and its data were not altered by unauthorized parties. Mechanisms protecting integrity usually try to prevent an alteration or tampering from occurring in the first place or to detect an intrusion after it has happened.

The third pillar, availability, refers to the property that a system and its data should be available to authorized parties in a timely manner. There are other important concepts such as authenticity, which is the property of data and transactions being genuine, and non-repudiation, which is the assurance that a party cannot deny a transaction, a statement, or a signature.

Software, firmware, and hardware design and implementation processes have errors or corner cases that can be exploited by an adversary. In computer security we call these weaknesses vulnerabilities. Computer systems will never be free from vulnerabilities because they are designed, implemented, and tested by humans, who always make mistakes. A vulnerability is thus a threat to security. We call an attack a threat that is realized by an adversary, usually exploiting one or more of a system's vulnerabilities.

The security challenges in cloud computing are not very different from those in traditional computing, except the cloud environment exacerbates the number of vulnerabilities and the impact of attacks. As the cloud environment comprises all layers of abstraction—application, operating system, architecture, and network—an attacker has several avenues for compromising the security of a cloud service. For example, a vulnerability in a web-based cloud application that does not sanitize inputs might cause the disclosure of sensitive data stored in a data center.

3.3 VULNERABILITIES AND ATTACKS

As we have mentioned, cloud computing services can present vulnerabilities in all layers of abstraction [4]. Figure 3.1 shows cloud security vulnerabilities according to the layer of abstraction where they can occur.

3.3.1 Application Layer

At the application level, a cloud application might have several weaknesses that allow an adversary to

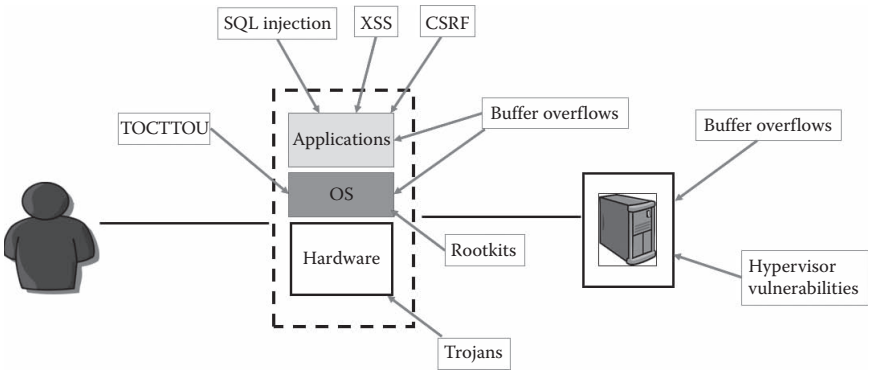

FIGURE 3.1 Cloud security vulnerabilities.

compromise a system. Many cloud applications are web based and have classic web vulnerabilities such as insufficient user input sanitation, which allows attacks such as SQL injection [5]. An SQL injection vulnerability allows an attacker to inject external code into a web scripting engine to be executed by a SQL query interpreter. It is still commonly reported in vulnerability databases even though it has been researched since the mid-2000s.

To understand this vulnerability, first consider a typical scenario of a user interacting with a web server hosting a web application that stores its data in a database. Usually the code for the web application and the database are stored in different machines. In a typical scenario, a web application deployed by a book retailer (e.g., Amazon) allows users to search for books based on their author, title, publisher, etc. The entire book catalog is held in a database and the application uses SQL queries to retrieve book details. Suppose a user searches for all books published by Wiley. The web scripting engine at the web application side receives, manipulates, and acts upon these data interpreting them as user-supplied data. The web scripting engine then constructs a SQL command that is a mix of instructions written by the application developer and the user input. This query causes the database to check every row within the books table, extract each of the records where the publisher column has the value "Wiley", and return the set of all these records. This record set is processed by the web application and presented to the user within an HTML page. Now, consider a scenario in which an attacker could cause a fracture in the interpretation of the data and break out of the data context. String data in SQL queries must be encapsulated within single quotation marks, to separate it from the rest of the query. In this scenario, an attacker supplies input containing a quotation mark to terminate the string (′) that she controls, plus a new SQL command modifying the query that the developer intended the application to execute. There is a clear fracture in how the input is interpreted at the boundary between the web scripting language and the SQL query interpreter. The double hyphen (--) in the attacker's input tells the query interpreter to ignore the remainder of the line even though there could be other commands included by the application developer. In this case, the consequence of this attack is the deletion of the entire "books" table from the database as illustrated in Figure 3.2.

Other web-based vulnerabilities include cross-site scripting (XSS) [6] and cross-site request forgery [7]. Further, the application code might be vulnerable to remote code injection via buffer overflows [8] if it is coded in a programming language that does not verify array bounds, such as C or C++. The application might also be vulnerable to sensitive data disclosure if it does not employ cryptography to maintain the confidentiality of the data it manipulates. The application code might also be vulnerable to compromise if its authentication and access control procedures have flaws. Another source of vulnerability at the application layer is flawed application programming interface (API) functions.

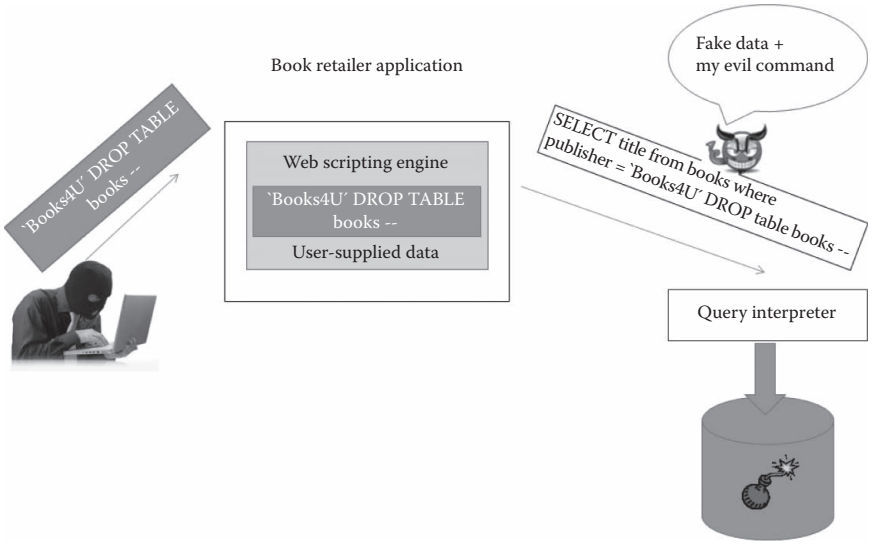

FIGURE 3.2 The SQL injection vulnerability.

3.3.2 Operating System Layer

Vulnerabilities at the operating system level can also compromise cloud security. The operating system (OS), as any piece of software written in C/C++, is vulnerable to buffer overflow vulnerabilities. OSs are also susceptible to vulnerabilities related to race conditions, such as the TOCTTOU (time of check to time of use) vulnerability. The TOCTTOU vulnerability is caused by changes in a system that occur between the *checking* of a condition (such as a security credential) and the *use* of the results of that check. TOCTTOU consists of a check phase, which establishes an invariant precondition (e.g., access permission), and a use phase, which operates on the object assuming the invariant is still valid. TOCTTOU usually occurs in SETUID processes, which have administrator privileges but can be invoked by unprivileged users so the process can perform tasks on the user's behalf. For example, a printing program is usually SETUID-root in order to access the printer device, which is an operation that requires administrative privileges. Running as if the user had the root privileges, the printer program discovers whether a user invoking its execution has permission to read and print a certain file by using the access function from the operating system. A classic example of TOCTTOU is given by the sequence of system calls* `access()` followed by `open()`. Processes are executed by OSs in an interleaved way, as the OS schedules one process at a time per CPU. Thus, a process does not execute at the CPU from start to finish without being interrupted. The CPU executes a process for a certain amount of time, then the OS pauses the current process and resumes the execution of a suspended process. Suppose that the process whose code given in Figure 3.3 is suspended before `fopen()` is executed after access is granted to access the file `/home/bob/symlink`. Then suppose an adversary's process is selected for execution and changes the symbolic link to point to `/etc/passwords`. When the first process is executed again, it will open a file (passwords) it did not have permission to do.

```
if (access("/home/bob/symlink", R_OK | W_OK) != -1)
{
        // Symbolic link can change here
        f = fopen("/home/bob/symlink", "rw");
        ...
}
```

FIGURE 3.3 The TOCTTOU vulnerability.

* Operating system API to user applications.

Operating systems are also vulnerable to installations of malicious drivers and extensions. Kernel extensions, especially device drivers, currently make up a large part of modern kernel code bases (approximately 70% in Linux and a larger percentage in Windows) [9]. Most of these extensions are benign and allow the system to communicate with an increasing number of diverse I/O devices without the need of OS reboot or recompilation. However, they pose a threat to system trustworthiness because they run with the highest level of privileges and can be vulnerable or malicious. Detection of malicious extensions is very challenging and most of the proposals in the literature have not been adopted by modern operating systems [10–12].

3.3.3 Hypervisor, Storage, Hardware, and Network

Because cloud computing depends on virtualization technology, vulnerabilities in hypervisors and virtual machine (VM) images can also compromise security. Any weakness or flaw in the hypervisor complex code might compromise the isolation between VMs residing in the same server. Cloud computing is also vulnerable to flaws in code performing migration of VMs among servers, and VM snapshot and rollback [13]. These vulnerabilities can lead to integrity compromises, data disclosure, and denial of service (DoS) attacks. VM images in public repositories might contain malware, or vulnerable or unpatched code.

There are several vulnerabilities regarding data that are unique to cloud computing. Data may be located in different countries which have different laws about the ownership of data [14]. Also, data disclosure might happen if data are not properly cleaned from secondary storage when moved or deleted [13].

At the architecture layer, just like in traditional computing, cloud computing is vulnerable to hardware Trojans, that introduce malicious functionality at the gate level [15]. At the network layer, cloud computing is also susceptible to vulnerabilities found in network protocols, usually causing a DoS attack such as a TCP-SYN flood, where an adversary sends more connection request packets (SYN packets) than a server can process, and consequently makes it unavailable to legitimate clients [16]. Cloud computing is also vulnerable to sniffing and snooping in virtual networks: a malicious VM can listen to the virtual network or even use address resolution protocol (ARP) spoofing to redirect packets from/to other VMs [17].

3.3.4 Cloud Security Mechanisms

There are many security mechanisms that when applied to the cloud environment can help mitigate several of the vulnerabilities described in the previous section. In this section we discuss some of them including mechanisms for data and virtualization security.

3.3.4.1 Data Security

Encryption is commonly used to protect the confidentiality of cloud data. It involves the transformation (encryption) of information using mathematical methods and a secret key, called an encryption key. Encrypted information can only be revealed to authorized parties with the use of a decryption key. Encryption prevents an adversary from eavesdropping cloud-sensitive data when it is stored in a data center and in transit over the network. There are two types of encryption techniques: symmetric and asymmetric [18,19].

Symmetric cryptography uses identical keys to both encrypt and decrypt the data, which makes the algorithms less complex than their asymmetric counterparts. Thus, they have better performance than asymmetric cryptography algorithms, usually 100 to 1000 times faster [19]. The level of security depends on the length of the key. The U.S. National Institute of Standards and Technology (NIST) recommends 160–512 bits. Symmetric cryptography is usually adopted for bulk encryption of data and is applied in protocols like Internet protocol security (IPSec) and transport layer security (TLS). A challenge for symmetric cryptography is how to securely distribute keys.

Asymmetric cryptography uses two different keys: a private key for encryption and a public key for decryption. For example, suppose that Alice wants to send a secret message to Bob using asymmetric cryptography. Alice first encrypts the plaintext message using Bob's public key, which is public and available to anyone. For example, Bob can make his public key available in a directory in his organization or in a public website. Bob also has a private key known only to him, which is different from his public key. In spite of being different, these keys are complimentary in function because Bob will use his private key to decrypt Alice's message. In other words, information in plaintext that is encrypted with a public key can only be decrypted with the correspondent private key. Asymmetric cryptography is used to solve the challenge of key distribution in symmetric cryptography and also as a mechanism to implement digital signatures.

3.3.4.2 Digital Signature

A digital signature is a way for a party to assure the authenticity of a message. Digital signatures should achieve non-repudiation; that is, it should be difficult for a party to forge a digital signature and to use a valid signature for a different message. Asymmetric cryptography is commonly used as a digital signature mechanism in cloud environments. In public/private key cryptography, data encrypted with someone's private key can only be decrypted with that person's public key. So, if for example, Bob has Alice's public key and wants her trusted digital signature in a document sent over the network (a usually untrusted channel), he can ask Alice to "sign" or encrypt the data or document with her private key. If Bob is able to decrypt the data with Alice's public key, he can be sure that only Alice could have encrypted the document because only she knows her private key.

Now let us consider a typical web application in the cloud environment. SSL is a network protocol used to secure data between two machines with encryption. To understand the importance of SSL for the security of web applications, consider a scenario when you wanted to buy a book from Amazon. While you are browsing the site, your web browser and the Amazon server are connected through the standard TCP protocol. Through this type of connection, the data exchanged between your browser (i.e., the items being searched) and the Amazon server (i.e., data about Amazon's items) are not encrypted. Because this exchange is not encrypted, an adversary that is situated in your local network can leverage network-sniffing programs that inspect the raw data in the network packets exchanged between your browser and Amazon's server. For example, an adversary could learn that the user of a specific machine is interested in books about Julius Caesar. Clearly, the user needs to prevent an adversary from being able to "sniff" her sensitive data such as credit card numbers and shipping information. Another challenge is that your browser has no guarantee that the server it is "talking to" is actually Amazon.com. For example, a user could have typed Amazom.com by mistake and could be communicating with a fake browser. So, the user would also need to be sure that the server taking the information is actually Amazon.com.

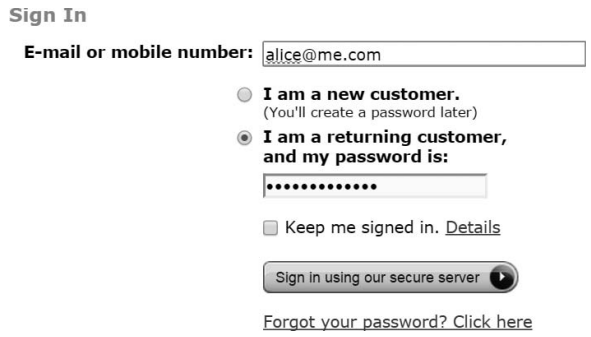

FIGURE 3.4 Example of secure authentication with SSL.

The SSL protocol accomplishes both needs by using cryptographic methods to hide what is being sent from one computer to another and by employing identification techniques that ensure the computer a browser is talking to can be trusted. In other words, by using SSL to purchase a book from Amazon, the user can be sure that no adversary will discover his/her credit card information and that any shared information is exclusively exchanged with the real Amazon.com.

As shown in Figure 3.4, when a user makes a financial transaction with Amazon.com, the browser and the server establish an SSL connection under HTTP (https). Notice how the lock becomes green showing that it is a trusted connection. Also notice that it reads *https* and not *http*.

When the user presses the sign-in or login" button in e-commerce servers (e.g., Amazon), his browser and the server will establish an SSL connection through a "handshake" and then a subsequent validation phase. In the handshake phase, the browser and the server agree on a particular encryption algorithm and the server sends a certificate to the client. This certificate is a piece of data issued by a trusted certification authority (CA) that binds a cryptographic public key to a particular entity (e.g., Amazon.com) and is designed to legitimate that the server is actually who it is claiming to be.

3.3.4.3 Hashing

Hashing is used to generate a one-way non-reversible representation of data for security [3,16,20]. A hash function converts a plain text message *m* into a fixed-size hash code, *H(m)*, that is usually called hash. An interesting property of a hash function is that no two different messages have the same hash. Also, once a piece of information is hashed there is no way to reverse it, as there is no "unhash key." Hashing can inform whether data stored in a data center or in transit were compromised [20]. For example, one can derive a fixed-length hashing code or message digest for some data. The message digest is usually smaller than the message itself. If a party needs to send the message over the network to another party, it can attach the message digest to the message. The receiving party verifies the integrity of the message by applying the same hash function to the message and verifying that the message digest is the same as the message. If an adversary tampered with the data, the message digest will differ.

3.3.5 Virtualization Security

There are many works in the literature addressing the security of VMs running in the cloud environment. Wang and Jiang introduced HyperSafe [21], an approach providing control flow integrity for hypervisors. It locks down write-protected memory pages and prevents them from being manipulated at runtime, which protects the integrity of hypervisor's code.

Santos et al. [22] proposed an approach for a trusted cloud computing platform (TCCP) that enables infrastructure as a service (IaaS) services such as Amazon EC2 to provide a closed-box execution environment. TCCP assures confidential execution of guest VMs and allows users to attest to the IaaS provider and determine if the service is secure before they launch their VMs.

Zhang et al. [23] introduced PALM, a VM live migration framework. This strategy consists of three modules for the privacy and integrity protection of the sensitive data, the metadata, and the live migration process itself.

3.4 PRIVACY AND SECURITY IN CLOUD STORAGE SERVICES

Nowadays more and more organizations have been adopting public cloud services to store their data (e.g., Microsoft Skydrive and Dropbox), as well as Amazon EC2 and MapReduce Framework, to process their data. Survey results have shown a steady increase of cloud adoption whereby 75% of those surveyed in 2013 used cloud platforms compared to 67% in 2012 [24]. The use of cloud technology will also be instrumental in pushing data sharing across multiple organizations (e.g., government organizations), which will be beneficial for many societal critical applications, such as homeland protection, cybersecurity, disease spreading control, and the green economy.

However, critical issues of data confidentiality [25–28] hinder the wide adoption of cloud technology, especially public clouds. It is critical that sensitive data stored and shared in the cloud be strongly secured from unauthorized access. That means data stored in the cloud should be selectively shared among different users, possibly within different organizations, based on access control policies. When enforcing access control, it is also important to protect authorized users' profile information (e.g., user role and location), which otherwise may lead to inferences about data contents. This is because advanced access control systems, such as the well-known attribute-based access control model [29], require disclosing enforcement system information about users to the access control. Moreover, data in the cloud may be transferred between data centers which may be located in different regions (or even countries), where cloud users do not have much information about where their data are stored and processed. Thus, ensuring privacy compliance during day-to-day operations is a very challenging task for both cloud service providers (CSPs) and their customers. There is a clear need to develop cloud-specific data securing techniques. The following section first presents a generic cloud data protection model and then reviews the state-of-the-art cloud data protection techniques. The section concludes with a discussion of some other possible causes of data leakage in the cloud.

3.4.1 Cloud Data Protection Models

In the cloud, we observe the following two important characteristics that impose challenges to the development of data protection techniques. A cloud service can be provided through a chain of service providers [27]. Let us denote the direct service provider as S and its direct and indirect contractors as S1, S2 … S*n*.

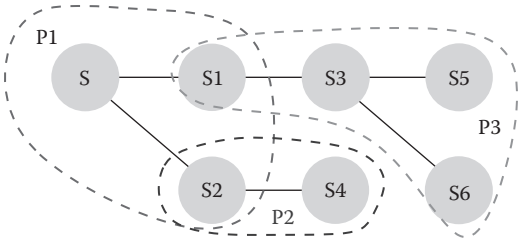

FIGURE 3.5 Service chain in the cloud.

As shown in Figure 3.5, the direct service provider is connected with six other service providers, whereby P1, P2, P3 denote the respective policies within the subgroups of service providers. For a user to select a service provider, the candidate service providers' privacy policies need to be checked to ensure they conform to users' privacy preferences. More strictly, indirect contractors' policies may need to be verified to satisfy the users' privacy requirements. When establishing the service relationship, policy agreement needs to be achieved not only between the user and S, but also between S and S1, S1 and S2, and so on. It is even more challenging to guarantee and enforce user's privacy requirements across multiple parties through the entire service period.

Some possible changes to the parties involved in a cloud service need to be considered as discussed in the literature [26]: a participating party may need to update its privacy policies, or a service provider may need to transfer its operations together with users' data to someone else because of the sale of company, a merger, seizure by the government, etc. All these events can affect the current policies agreed by all parties. The challenge is how to efficiently and effectively reflect such a change so the impact of the change on achieving policy agreement and policy enforcement can be minimized.

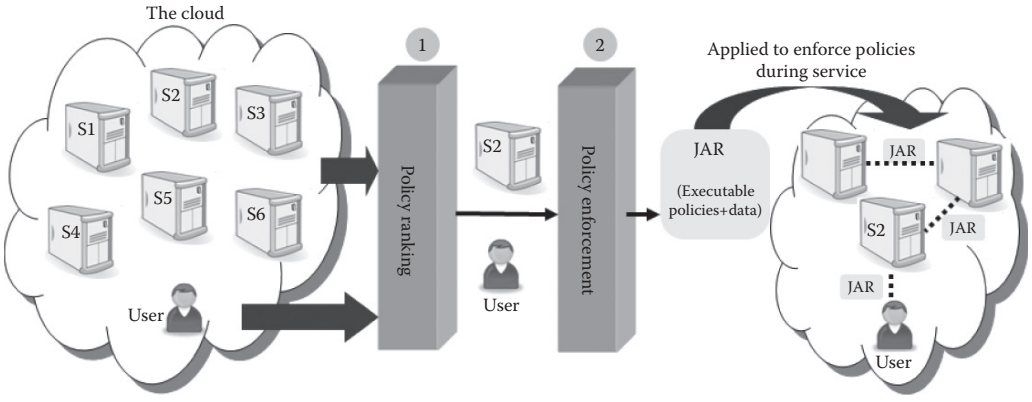

FIGURE 3.6 Cloud data protection model.

Based on the above observations, a cohesive data protection framework has been proposed [29]. The framework, as illustrated in Figure 3.6, consists of three major components: policy ranking, policy integration, and policy enforcement. In particular, a user joins the cloud and faces several CSPs, each of them able to provide the service he needs. In order to find the service provider whose privacy policies best fit the user's privacy requirements, the user's privacy requirements and service providers' current privacy policies are fed into the policy ranking module together. The ranking module helps the user select the service provider that has the most compatible privacy policies. Since the selected service provider still may not have policies that exactly match the user's requirements, the second step is to send their policies to the policy integration module which will automatically generate an integrated policy as agreed by both parties. The integrated policy will be in two formats. One is in an actual policy format, i.e., a policy written in a certain policy language. The other is in an executable format (like a Java JAR file) which will be used for the subsequent policy enforcement. Throughout the service, the user's data privacy will be protected by the executable policy, and the executable policy may also travel among contractors associated with the direct service provider.

3.4.2 Enforcing Access Control Policies in the Cloud

As shown in the above data protection model, the policy enforcement component is critical to the overall security of the data in the cloud [25,26]. There have been some existing efforts that aim to address this issue.

Some approaches [30–32] employ broadcast key management schemes to provide access control on the cloud data. Such kind of approaches group data items based on access control polices and encrypt each group with a different key. Users are then given only the keys for the data items they are allowed to access. All these encryption activities have to be performed at the owner's premises, thus incurring high communication and computation costs. For example, if a user is revoked, the owner must download the data affected by this change from the cloud, generate a new encryption key, re-encrypt the downloaded data with the new key, and then upload the re-encrypted data to the cloud. Besides broadcast key management, attribute-based encryption (ABE) [33] has also been applied to preserve data privacy in the cloud. However, this approach is not efficient either in handling frequent user joins and departures. To improve the performance, some work introduces a third party called "proxy" [28] to conduct re-encryption in case of the change of data recipients. However, they do not protect the identity attributes of the users.

Another interesting category of work [34] aims to tightly bind data with access control policies to ensure that policies will be automatically enforced whenever and wherever the data are accessed. The basic idea is to leverage Java archives (JARs). The advantages of using Java techniques are mainly twofold. First, JARs provide a lightweight and portable container for the data as well as the enforcement engine. Second, JARs have minimal infrastructure requirements (i.e., a valid Java runtime environment (JRE) running at the remote end) which allow our approach to be easily adopted. The cage is a nested JAR consisting of one outer JAR and one or more inner JARs as shown in Figure 3.7. User data items regarding different policies will be stored in different inner JARs in encrypted format along with encrypted log files. The outer JAR contains the authentication module and policy enforcement engine. It is responsible for authenticating entities who want to access the data, selecting the correct inner JAR and enforcing the corresponding policies. The cage will be sealed and signed at the construction.

3.4.3 Other Possible Causes of Data Leakage in the Cloud

Besides enforcing users' privacy policies on their actual data file, there is another interesting and very important privacy problem caused by data indexing. Indexes may contain a great amount of information concerning the data itself. Since indexes are usually constructed after the service provider receives the user's data and decides to build indexes to improve search performance, users may not even be aware of such usage of their data which probably leaks much more information than that intended by the users. More detailed discussion is the following.

The most common scheme for supporting efficient search over distributed content is to build a centralized inverted index. The index maps each term to a set of documents that contain the term and is queried by the searcher to obtain a list of matching documents. This scheme is usually adopted by web search engines and mediators. As suggested in [35], the scheme can be extended to support access-controlled search by propagating access control policies along with content to the

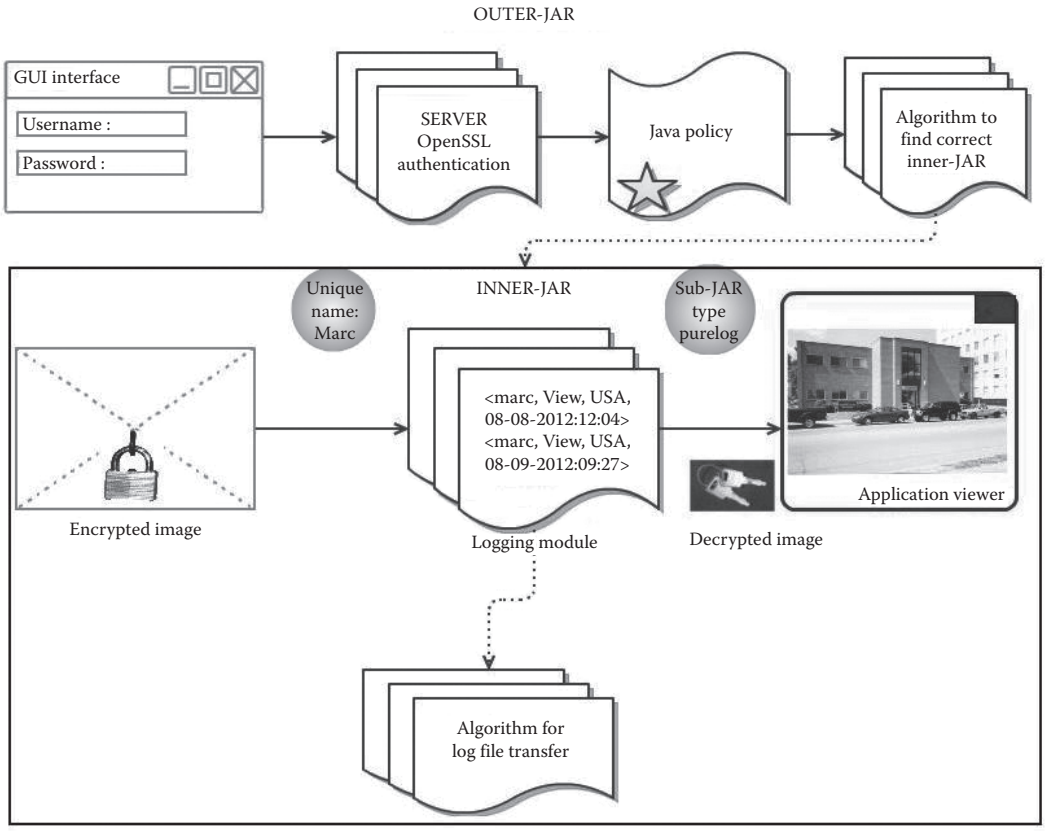

FIGURE 3.7 Nested JARs.

indexing host. The index host must apply these policies for each searcher to filter search results appropriately. Since only the indexing host needs to be contacted to completely execute a search, searches are highly efficient. A centralized index however exposes content providers to anyone who has access to the index structure. This violation of access control may not be tolerable in the cloud, where assumptions on the trust of the indexing server no longer hold. Further, compromise of the index host by hackers could lead to a complete and devastating privacy loss. Decentralized indexing is an alternative architecture used to identify a set of documents (or hosts of documents) matching the searcher's query. These hosts are then contacted directly by the searcher to retrieve the matching documents. Access control can be supported by simply having the providers enforce their access policies before providing the documents. However, indexes are still hosted by untrusted machines over which providers themselves have no control.

In order to overcome the aforementioned issues, some works have explored the possibility of creating private indexes by relying on predicate-based cryptography [36,37]. While notable, these works lack concrete applicability due to the key management requirements and the computational overhead. Bawa et al. [35] proposed an interesting approach to private indexing by introducing a distributed access control enforcing protocol. A more recent work [38] deals with the privacy problem from a different perspective by empowering the users to gain better control over the indexes. In particular, a three-tier data protection framework was proposed, which provides strong, medium, and low protection according to the data owner's needs. This is achieved by building a similar JAR file as discussed in the previous section. The JAR file encloses both data and policies, and implements the different levels of protection as follows:

- *Strong protection:* The service provider is not allowed to read the sensitive portion of the user's file, so as to negate the risk of indexing being conducted on a sensitive portion of the document that could lead to privacy leaks. Users need to provide sensitive fields regarding their data files. Then the JAR performs the function of selecting which fields

are to be read by the CSPs. The protected fields are simply skipped during the sequential reading of the file by identifying the position where the protected field starts.

- *Medium protection:* This option disables random access to the data file in order to prevent effective indexing over the file. The CSP will be enforced to read the file in a sequential order before it can locate the content that it needs.
- *Low protection:* The user specifies clearly in the policy the usage of his data file and the usage of indexing. The service provider is assumed trusted and will inform and negotiate with the user the keywords to be used for indexing purposes.

3.5 PRIVACY AND SECURITY IN MULTICLOUDS

Cloud computing is growing exponentially, whereby there are now hundreds of CSPs of various sizes [39]. A concept of a cloud-of-clouds (also called an intercloud) is proposed and has been studied in recent years [39,40]. In a cloud-of-clouds, we disperse data, with a certain degree of redundancy, across multiple independent clouds managed by different vendors, such that the stored data can always be available even if a subset of clouds becomes inaccessible.

The multicloud environment (Figure 3.8) [40] offers plenty of new opportunities and avenues to cloud consumers. Cloud consumers will be able to leverage not just one cloud provider, but many, to solve their diverse needs and switch providers if one ceases service. To promote the multiple clouds, IEEE has initiated the Intercloud Testbed project [39] that helps make interactions among multiple clouds a reality.

3.5.1 Desired Security and Privacy Properties in Multiclouds

While cloud consumers may enjoy cheaper data storage and powerful computation capabilities offered by multiple clouds, consumers also face more complicated reliability issues and privacy preservation problems of their outsourced data. More specifically, as it is difficult to obtain clear guarantees on the trustworthiness of each CSP [41], cloud consumers are typically advised to adopt searchable encryption techniques [42,43] to encrypt their outsourced data in a way that the encrypted data can be directly searched by the CSPs without decryption. Despite many efforts devoted to improving the efficiency and security of the searchable encryption, there is little consideration for ensuring the reliability of the searchable encrypted data. Though cloud storage provides an on-demand remote backup solution, it inevitably raises dependability concerns related to having a single point of failure and to possible storage crash. An ideal multicloud environment should possess the following properties:

- *Reliability:* Given n CSPs, the system should still function if at least t $(t < n)$ CSPs are available, where t is a predefined threshold value for the system.
- *Semantic security:* The system should be semantically secure [44] by satisfying the following two requirements. First, given the file index and the collection of encrypted files, no adversary can learn any information about the original files

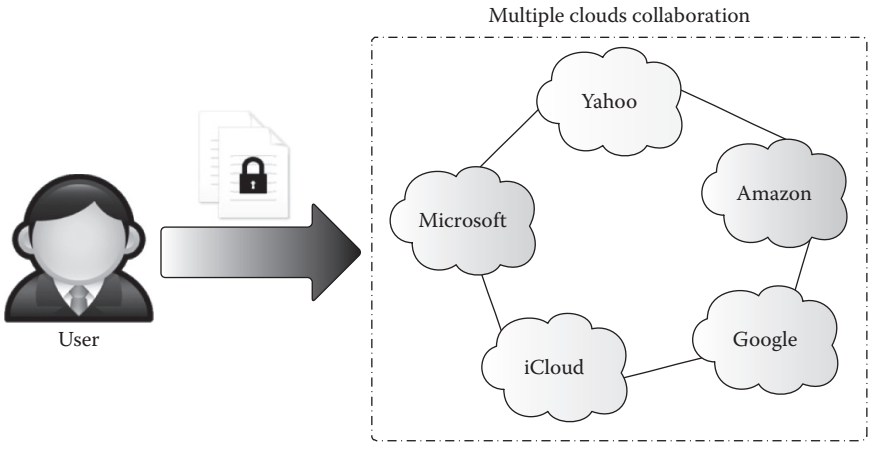

FIGURE 3.8 Multiclouds.

except the file lengths. Second, given a set of trapdoors for a sequence of keyword queries, no adversary can learn any information about the original files except the access pattern (i.e., the identifiers of the files that contain the query keyword) and the search pattern (i.e., whether two searches are looking for the same keyword or not).

- *Trapdoor security:* This requires that any information about the query keyword—including the search pattern—should not be leaked before the multiple CSPs' collaborative search. The requirement holds even if $t - 1$ CSPs are compromised by an adversary.
- *Robustness:* When the protocol successfully completes, the correct files are returned and reconstructed by the users. When the protocol aborts, even in the collaborative search stage, nothing is returned and CSPs learn nothing about the file collection or the underlying searched keyword.

3.5.2 Ensuring Security, Privacy, and Reliability in Multiclouds

It is worth noting that very few works consider the issue of simultaneously ensuring searchability, privacy, and reliability on data outsourced to multiple clouds. Existing reliability guarantees solely rely on each CSP's own backup solution, which however could be a single point of failure. For example, the crash of Amazon's elastic computing service in 2011 took some popular social media sites offline for a day and one energy department collaboration site unavailable for nearly 2 days. More seriously, this crash has permanently destroyed many customers' data with serious consequences for some users [45]. Recent studies [46,47] proposed regenerating codes for data reliability in distributed storage like the cloud. Regenerating codes built on the concept of network coding aim at intelligently mixing data blocks that are stored in existing storage nodes, and then generating data at a new storage node. It is shown that regenerating codes reduce the data repair/recovery traffic over traditional erasure codes subject to the same fault tolerance level.

One naïve approach to achieve searchability, privacy, and reliability on data outsourced to multiple clouds is the trivial replication. In particular, we can replicate the single-user searchable encryption scheme to n CSPs. Each CSP stores the same searchable ciphertexts. Thus, even if $n - 1$ CSPs are unavailable, the remote files are still accessible. However, this approach is not space efficient as it takes a lot of capacity to save the replicas. To reduce redundancy while tolerating CSP failures, another possible approach is to use erasure coding. Specifically, we can employ secret sharing techniques to encode the files into a set of shares and distribute the shares to n CSPs so that if a certain number of CSPs are unavailable, the shares from the rest of the CSPs can be used for reconstructing the original files. Compared with the first approach, the second approach saves storage space by reducing the reliability guarantee from $n - 1$ to $n - t$. Besides the downgrade of the reliability guarantee, the second approach is also more time consuming due to the multiple rounds of communications needed among CSPs to complete the protocol.

In order to achieve both space and time efficiency, a recent work called the privacy-preserving STorage and REtrieval (STRE) mechanism has been proposed [48]. The STRE mechanism enables cloud users to distribute and search their encrypted data in CSPs residing in multiple clouds while obtaining reliability guarantees. The STRE mechanism follows a similar spirit to the second naïve approach and proposes more efficient secret sharing–based protocols. Moreover, the STRE mechanism also offers better protection on the user's search pattern. Specifically, many works [44,49] on searchable encryption would completely disclose the user's search pattern that indicates whether two searches are for the same query keyword or not. In the STRE mechanism, this kind of pattern leak risk is reduced because the search is conducted on a distributed basis and the search pattern will be revealed only if more than t CSPs collude. An overview of the STRE mechanism is given below.

The STRE mechanism consists of three major phases: the setup phase, storage phase, and retrieval phase. During the setup phase, a master secret key is generated from a security parameter and assigned to the cloud user. During the storage phase, the user takes a collection of input files and the master secret key, and generates a set of file shares and a file index. The file shares and file index are uploaded to the corresponding CSP. The retrieval phase is to search the files containing a certain keyword. The user generates a set of trapdoor shares based on the query keyword and his/her master secret key. The trapdoor share is sent to the respective CSP. Then, the CSPs collaborate to search with their individual trapdoor share, and relay the search results back to

the user. Finally, the user reconstructs and decrypts the results and obtains the clear files, each of which contains the query keyword.

3.6 CLOUD ACCOUNTABILITY

One of the main motivations underlying cloud computing systems is the possibility to outsource complex computation and to store large amounts of data. However, to this date users have few, if any, technical ways to actually check resource consumption and data storage status, once they are "shipped" to the cloud providers. Cloud providers do not allow users to observe their inner workings, and users have no reliable information about their data whereabouts or the status of their actual computation. While encryption can ensure confidentiality of outsourced data, and access control can enhance these guarantees by controlling who is accessing what portion of data (or, more generally, resources), ensuring integrity and verifying data usage are difficult.

To cope with these issues, accountability is an important security requirement in cloud systems. Accountability aims at providing a detective, rather than preventive solution to cloud users [50]. The goal is not to protect data privacy or control resource usage, but rather to verify who has obtained access to resources, and how. According to Ruebsamen and Reich [51], cloud accountability can help "make data processing in the cloud more transparent", so that "captured data-lifecycle events can be matched against policies in audits and thereby show the customer, that his data are handled appropriately".

Unlike privacy protection technologies, which are built on the hide-it-or-lose-it perspective, accountability focuses on keeping the data usage transparent and trackable. Traditional accountability methods focus on data collection and post-mortem analysis, and third-party audits. Typically, an accountability system will include data collection, through log collection and event monitoring, followed by auditing and analysis of the collected evidence to detected possible anomalies or simply check actual data lineage. However, as stated by Ko et al. [50], cloud computing's promise of elasticity empowered by virtualization introduces several new complexities in accountability, related to the ability of processing and monitoring multiple virtual and physical resources, connected in a highly dynamic fashion. Further, log collection becomes a challenging task in itself, due to the potentially limited trust assumed at remote nodes, which may not be reliable enough to collect the evidence required for effective accountability.

Though the current industry practice is based on regulatory and contractual agreements between clients and cloud providers to ensure resource and data accountability [52], researchers have investigated ways to overcome the above challenges through technical means. A common approach is to rely on logs produced as part of the computation and storage process, possibly augmented with ad-hoc information necessary to track access and other actions against the data. Recent work has proposed doing so by using *digital evidence bags* to address interoperability and evidence integrity, while additional metadata can be used to facilitate evidence processing [51]. Others have proposed using agents to facilitate distributed data collection and provide a more dynamic infrastructure [26,53].

A related body of work, aiming at addressing similar security issues, is the growing literature on cloud provenance. Cloud provenance focuses on recording ownership and processing history of data objects, and it is considered a vital component to the success of data forensics in cloud computing [54,55]. In particular, researchers have highlighted that provenance can be a way to derive the so-called chain of custody, which should clearly depict how the data were collected, analyzed, and preserved. This is particularly important for cloud forensics, i.e., in case of legal dispute, to ensure that provenance information is admissible as evidence in court. Among existing proposals, a recent proposal on provenance relies on the notion of provable data possession (PDP) [26], which essentially builds cryptographic proofs of data possession that can be verified at the client end.

3.7 SUMMARY

This chapter discussed the security baselines for cloud computing and how its distributed and networked environment can scale the number of security threats cloud software and data face. The same vulnerabilities, threats, and attacks from traditional computing apply to cloud computing, with additional threats coming from the use of virtualized resources and hypervisors. This chapter also discussed countermeasures for threats at several layers of abstraction. Finally, this chapter addressed privacy and security in cloud storage services and multiclouds, and cloud accountability.

The biggest research challenge in cloud security is how cloud providers can assure a level of security to cloud clients. Cloud clients have no option but to trust their cloud providers, and they usually have no assurances on the level of security actually being provided.

REFERENCES

1. M. Bishop, *Computer Security: Art and Science*, Addison Wesley, 2003.
2. W. Stallings and L. Brown, *Computer Security Principles and Practice*, Pearson, 2012.
3. M. Goodrich and R. Tamassia, *Introduction to Computer Security*, Pearson, 2010.
4. B. Grobauer, T. Walloschek and E. Stocker, Understanding Cloud Computing Vulnerabilities, *IEEE Security & Privacy*, vol. 9, no. 2, pp. 50–57, 2011.
5. Z. Su and G. Wassermann, The Essence of Command Injection Attacks in Web Applications, *ACM SIGPLAN-SIGACT Symposium on Principles of Programming (POPL)*, pp. 372–382, 2006.
6. G. Wassermann and Z. Su, Static Detection of Cross-Site Scripting Vulnerabilities, *International Conference on Software Engineering (ICSE)*, pp. 171–180, 2008.
7. A. Barth, C. Jackson and J. C. Mitchell, Robust Defenses for Cross-Site Request Forgery, *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*, 2008.
8. K.-S. Lhee and S. J. Chapin, Buffer Overflow and Format String Overflow Vulnerabilities, *Software—Practice and Experience*, vol. 33, no. 5, pp. 423–460, 2003.
9. A. Kadav and M. M. Swift, Understanding Modern Device Drivers, *Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2012.
10. A. Srivastava and J. Giffin, Efficient Monitoring of Untrusted Kernel-Mode Execution, *Network and Distributed System Security Symposium (NDSS)*, 2011.
11. X. Xiong, D. Tian and P. Liu, Practical Protection of Kernel Integrity, *Network and Distributed System Security Symposium (NDSS),* 2011.
12. D. Oliveira, N. Wetzel, M. Bucci, D. Sullivan and Y. Jin, Hardware-Software Collaboration for Secure Coexistence with Kernel Extensions, *ACM Applied Computing Review Journal*, vol. 14, no. 3, pp. 22–35, 2014.
13. K. Hashizume, D. G. Rosado, E. Fernández-Medina and E. B. Fernandez, An Analysis of Security Issues for Cloud Computing, *Journal of Internet Services and Applications*, vol. 4, pp. 5, 2013.
14. L. Ertaul, S. Singhal and S. Gökay, Security Challenges in Cloud Computing, *Proceedings of the International Conference on Security and Management (SAM)*, pp. 36–42, 2010.
15. M. Tehranipoor, H. Salmani and X. Zhang, *Integrated Circuit Authentication—Hardware Trojans and Counterfeit Detection*, Springer, 2014.
16. J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*, Pearson, 2012.
17. J. S. Reuben, *A Survey on Virtual Machine Security*, Helsinki University of Technology, 2007.
18. Atmel, Bits and Pieces, Symmetric vs. Asymmetric Encryption: Which Way Is Better?, 2013, available at http://blog.atmel.com/2013/03/11/symmetric-vs-asymmetric-encryption-which-way-is-better/.
19. Microsoft, Microsoft Technet, Encryption, 2015, available at https://technet.microsoft.com/en-us/library/Cc962028.aspx.
20. T. Erl, R. Puttini and Z. Mahmood, *Cloud Computing: Concepts, Technology & Architecture*, Prentice Hall, 2013.
21. Z. Wang and X. Jiang, HyperSafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity, *IEEE Symposium on Security and Privacy*, pp. 380–395, 2010.
22. N. Santos, K. Gummadi and R. Rodrigues, Towards Trusted Cloud Computing, *Conference on Hot Topics in Cloud Computing*, 2009.
23. F. Zhang, Y. Huang, H. Wang, H. Chen and B. Zang, PALM: Security Preserving VM Live Migration for Systems with VMM-Enforced Protection, *Trusted Infrastructure Technologies Conference*, pp. 9–18, 2008.
24. N. Bridge, 2013 Future of Cloud Computing Survey Reveals Business Driving Cloud Adoption in Everything as a Service Era, 2013, available at http://www.northbridge.com/2013-future-cloud-computing-survey-reveals-business-driving-cloud-adoption-everything-service-era-it.
25. A. Cavoukian, Privacy in the Clouds, *Identity in the Information Society*, vol. 1, no. 1, pp. 89–108, 2008.
26. R. Gellman, *Privacy in the Clouds: Risks to Privacy and Confidentiality from Cloud Computing*, World Privacy Forum, 2009.
27. P. T. Jaeger, J. Lin and J. M. Grimes, Cloud Computing and Information Policy: Computing in a Policy Cloud?, *Journal of Information Technology and Politics*, vol. 5, no. 3, pp. 269–283, 2009.
28. S. Pearson and A. Charlesworth, *Accountability as a Way Forward for Privacy Protection in the Cloud,* Hewlett-Packard Development Company (HPL-2009–178), 2009.
29. D. Lin and A. Squicciarini, Data Protection Models for Service Provisioning in the Cloud, *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies*, pp. 183–192, 2010.
30. M. Nabeel, E. Bertino, M. Kantarcioglu and B. Thuraisingham, Towards Privacy Preserving Access Control in the Cloud, *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaborate Com),* pp. 172–180, 2001.
31. M. Nabeel, N. Shang and E. Bertino, Privacy Preserving Policy-Based Content Sharing in Public Clouds, *IEEE Transaction on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2602–2614, 2013.

32. N. Shang, M. Nabeel, F. Paci and E. Bertino, A Privacy-Preserving Approach to Policy-Based Content Dissemination, *IEEE 26th International Conference on Data Engineering (ICDE)*, pp. 944–955, 2010.
33. A. Sahai and B. Waters, Fuzzy Identity-Based Encryption, *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pp. 457–473, 2005.
34. S. Sundareswaran, A. Squicciarini, D. Lin and S. Huang, Ensuring Distributed Accountability for Data Sharing in the Cloud, *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 9, no. 4, pp. 556–568, 2012.
35. M. Bawa, R. J. Bayardo, R. Agrawal and J. Vaidya, Privacy-Preserving Indexing of Documents on the Network, *Very Large Data Base Journal*, vol. 18, no. 4, pp. 837–856, 2009.
36. Y.-C. Chang and M. Mitzenmacher, Privacy Preserving Keyword Searches on Remote Encrypted Data, 2004.
37. Z. Yang and S. Z. N. Wright., Towards Privacy-Preserving Model Selection, *PinKDD*, pp. 138–152, 2007.
38. A. Squicciarini, S. Sundareswaran and D. Lin, Preventing Information Leakage from Indexing in the Cloud, *IEEE International Conference on Cloud Computing,* 2010.
39. J. Weinman, Will Multiple Clouds Evolve into the Intercloud?, 2013, available at http://www.wired.com.
40. Y. Zhu, H. Hu, G.-J. Ahn and M. Yu, Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage, *IEEE Transactions on Parallel Distributed Systems*, vol. 23, no. 12, pp. 2231–2244, 2012.
41. D. Owens, Securing Elasticity in the Cloud, *Communications of the ACM*, vol. 53, pp. 46–51, 2010.
42. D. X. Song, D. Wagner and A. Perrig, Practical Techniques for Searches on Encrypted Data, *IEEE Symposium on Security and Privacy*, pp. 44–55, 2000.
43. S. Kamara and K. Lauter, Cryptographic Cloud Storage, *Financial Cryptography and Data Security, Ser. Lecture Notes in Computer Science*, vol. 6054, pp. 136–149, 2010.
44. R. Curtmola, J. A. Garay, S. Kamara and R. Ostrovsky, Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions, *ACM Conference on Computer and Communications Security (CCS),* pp. 79–88, 2006.
45. H. Blodget, Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers', 2011.
46. B. Chen, R. Curtmola, G. Ateniese and R. Burns, Remote Data Checking for Network Coding-Based Distributed Storage Systems, *ACM Workshop on Cloud Computing Security,* pp. 31–42, 2010.
47. Y. Hu, Y. Xu, X. Wang, C. Zhan and P. Li, Cooperative Recovery of Distributed Storage Systems from Multiple Losses with Network Coding, *IEEE Journal on Selected Areas in Communications,* vol. 28, pp. 268–276, 2010.
48. J. Li, D. Lin, A. Squicciarini and C. Jia, STRE: Privacy-Preserving Storage and Retrieval over Multiple Clouds, *10th International Conference on Security and Privacy in Communication Networks (SecureComm),* 2014.
49. S. Kamara, C. Papamanthou and T. Roeder, Dynamic Searchable Symmetric Encryption, *ACM Conference on Computer and Communications Security,* pp. 965–976, 2012.
50. R. K. L. Ko, et al., TrustCloud: A Framework for Accountability and Trust in Cloud Computing, *IEEE World Congress on Services*, 2011.
51. T. Ruebsamen and C. Reich, Supporting Cloud Accountability by Collecting Evidence Using Audit Agents, *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom),* vol. 1, 2013.
52. S. Pearson, Toward Accountability in the Cloud, *IEEE Internet Computing*, vol. 4, pp. 64–69, 2011.
53. S. Sundareswaran, A. C. Squicciarini and D. Lin, Ensuring Distributed Accountability for Data Sharing in the Cloud, *IEEE Transactions on Dependable and Secure Computing,* pp. 556–568, 2012.
54. R. Lu, X. Lin, X. Liang and X. Shen, Secure Provenance: The Essential of Bread and Butter of Data Forensics in Cloud Computing, *5th ACM Symposium on Information, Computer and Communications Security (ASIACCS),* pp. 282–292, 2010.
55. M. R. Asghar, M. Ion, G. Russello and B. Crispo, Securing Data Provenance in the Cloud, *Proceedings of the IFIP WG 11.4 International Conference on Open Problems in Network Security (iNetSec),* 2011.