Chapter 5.3: Data Security in Cloud Computing

Shucheng Yu[1], Wenjing Lou[2], and Kui Ren[3]

[1] University of Arkansas at Little Rock, AR, USA
[2] Virginia Polytechnic Institute and State University, VA, USA
[3] Illinois Institute of Technology, IL, USA

Abstract. Cloud Computing has come into reality as a new IT infrastructure built on top of a series of techniques such as distributed computing, virtualization, etc. Besides the many benefits that it can bring forth, Cloud Computing also introduces the difficulty of protecting the security of data outsourced by cloud users. This chapter will first study the basic concepts and analyze the essentials of data security issues pertaining to Cloud Computing. Then we elaborate on each issue by discussing its nature and existing solutions if available. Specifically, we will emphasize on issues of protecting data confidentiality/integrity/availability, securing data access and auditing, and enforcing the regulations and compliances regarding to data security and privacy.

Keywords Cloud Computing, data security, confidentiality, integrity, availability, access control

1 Overview

Cloud Computing is a new IT infrastructure in which computing resources are provided as a utility to cloud users in the pay-as-you-go manner. By integrating techniques such as Service Oriented Architecture (SOA), virtualization, distributed computing and etc, cloud computing offers elastic, on-demand and measured services to cloud users anytime anywhere whenever Internet is available, and enable them to enjoy the illusionary unlimited computing resources. The services provided by the cloud can be at different levels of the system stack, which can be described by the terminology of "X as a service (XaaS)" where X could be Software, Infrastructure, Hardware, Platform and etc. For example, Amazon EC2 provide Infrastructure as a service and allow cloud users to manage virtual machine instances and control almost the entire software stack above the OS kernel; Google AppEngine provides Software as a service which is targeted at traditional web applications; Microsoft Azure offers services which are intermediate between AppEngine and EC2. By deploying applications in the cloud, cloud users are able to enjoy massive and elastic computing resources without the large capital outlays in building their own data centers. Such a fact will significantly benefit the IT industry, especially small and medium IT enterprises, letting alone individuals, who were greatly limited by computing resources. For this reason, Cloud computing is believed to have the potential to shape the IT industry in the future.

1.1 What is Cloud Computing?

Although the benefits of cloud computing are obvious, it is not trivial to provide a concrete definition for cloud computing due to its intrinsic complexity. To the date when this book is written, there is no standardized definition of the term *cloud computing* except several attempts by leading institutions and standard organizations. A research group from the University of California at Berkeley [19] defines cloud computing as below:

> *Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS). The datacenter hardware and software is what we will call a Cloud. When a Cloud is made available in a pay-as-you-go manner to the general public, we call it a Public Cloud; the service being sold is Utility Computing. We use the term Private Cloud to refer to internal data centers of a business or other organization, not made available to the general public. Thus, Cloud Computing is the sum of SaaS and Utility Computing ...*
>
> - "Above the Clouds: A Berkeley View of Cloud Computing"

NIST [15] gives the following unofficial definition of cloud computing:

> *cloud computing is a "pay-per-use model for enabling available, convenient and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*
>
> - NIST unofficial draft

Notwithstanding that there is no such a unique definition of cloud computing, these works together do outline several most important characteristics of cloud computing: 1) Computing resources at different level of the system stack are provided as cloud services in the pay-as-you-go manner like traditional utility services, e.g., Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). Cloud users just need to pay for what they have actually used. 2) Rapidly elastic and scalable resources are available to cloud users. Cloud users are able to launch more computing resources at peak time and release them at nonpeak times, and saves their capital expenditure in hardware/software to deal with the fluctuation in workloads. 3) The services are provided in the on-demand manner and can be configured by cloud users themselves. This makes it very convenient for cloud users to access cloud services as they no longer need to interact with the system administrator and go through the usually lengthy processing routines. 4) Cloud services are made accessible via the Internet. Cloud users can launch these services on any platform that supports web techniques. 5) Computing resources are pooled and provided to cloud users independent of their locations. Besides these essential characteristics, Cloud Computing also has other properties such as multi-tenant architecture, i.e., applications of numerous customers may co-run or co-exist on the same physical device. According to its ownership and the technical architecture, Cloud Computing can be categorized as *Public Cloud*, *Private Cloud*, *Hybrid Cloud* and *Community Cloud*. Pubic Clouds provide services to the general public across the Internet while a Private Cloud exclusively serves a single enterprise/organization. Hybrid Clouds integrate models of both Public Cloud and Private Cloud to meet specific business and technology needs. Community Clouds are usually used by organizations with similar objectives.

1.2 Key Enabling Techniques of Cloud Computing

Although the term *Cloud Computing* is new, the underlying concept of cloud computing is actually not new. In the 1960s, John McCarthy mentioned that "computation may someday be organized as a public utility" in his speaking at the MIT Centennial. Douglas Parkhill in his 1966 book [47] thoroughly explored the characteristics of the "Computer Utility" which are very similar to those characteristics of the modern-day cloud computing. However, cloud computing, or the "Computer Utility", had not become a reality until the late 2000s when several critical enabling techniques at various levels of the system stack are all made available: broadband networks, the Web technology, Service Oriented Architecture (SOA), Software as a Service (SaaS), virtualization, distributed computing and the plentiful of software and operating systems. The broadband networks serve as a fundamental element in cloud computing for efficiently coupling physically distributed resources into a logically integrated service and providing smooth remote access for cloud users. The Web technologies offer platform independent ways for users to visualize and configure remote services. SOA makes it possible to deploy applications based on a loosely-coupled suite of services across multiple separate systems/servers over the Internet. SaaS provides application level of services in a pay-as-you-go model. Virtualization abstracts logical devices from physical devices and allows co-residence of multiple logically isolated instances such as operation systems on a single physical machine. Virtualization and distributed computing together make computing as utility and elasticity of computing resources possible. The availability of high-performance and cost-effective computing and storage hardware devices is fundamental to the illusion of unlimited resource.

1.3 Security in Cloud Computing

The many characteristics of Cloud computing have made the long dreamed vision of computing as a utility a reality and will have the potential to benefit and shape the whole IT industry. When deciding whether or not to move into the cloud, potential cloud users would take into account factors such as service availability, security, system performance and etc, among which security is the main concern according to a survey conducted by the IDC Enterprise Panel in 2008. However, the security issue of Cloud Computing is intrinsically complicated, which can be explained by the fact that cloud computing is built on top of existing techniques and architectures such as SOA, SaaS, distributed computing and etc. When combining all the benefits of these techniques and architectures, Cloud Computing also inherits almost all their security issues at various levels of the system stack. Besides this, the operation model of Cloud Computing will also reshape the trust model when cloud users move their applications from within their enterprise/organization boundary into the open cloud. By doing so, cloud users may lose physical control over their applications and data. In cloud environments network perimeters will no longer exist from cloud users' perspective, which renders traditional security protection mechanisms such as firewalls not applicable to cloud applications. Cloud users have to heavily rely on the cloud service providers for security protection. On the other hand, in cloud computing (except private clouds) users and cloud service providers are not necessarily from the same trust domain. In applications such as heathcare, cloud service providers and/or their system administrators may not even be allowed to access sensitive user data when providing security protection according to corresponding regulations/compliances. It requires that cloud service providers are able to provide necessary security services to meet individual cloud users' security requirements while abiding to the regulations/compliances. In non-sensitive applications, it is also important to protect cloud users' critical data and help them verify security services provided by the cloud. Secure auditing mechanisms are usually necessary for this purpose. In Cloud Computing the multi-tenancy property will make applications from different organizations and trust domains reside and interact on the same physical computing resources. This will inevitably bring forth more security risks in the sense that any intentional or inadvertent misbehavior by one cloud user would make other co-residences victims, and creates more opportunities for malicious attackers from the Internet. To address all these security issues in Cloud Computing, we need to explore the nature of Cloud Computing security problems and answer the following questions: Which objects are we going to protect? Who can be the potential attackers and how would they attack? What kind of security services should we provide? Which security mechanisms should we use?

In this chapter, we particularly focus on the issue of data security. More specifically, we want to identify the types of data that we need to protect, potential attackers in Cloud Computing and attacks they may launch to compromise data security. Then we discuss necessary security services for data security as well as corresponding security mechanisms for providing these security services.

2 Data Security in Cloud Computing

Data protection is a crucial security issue for most organizations. Before moving into the cloud, cloud users need to clearly identify data objects to be protected and classify data based on their implication on security, and then define the security policy for data protection as well as the policy enforcement mechanisms.

For most applications, data objects would include not only bulky data at rest in cloud servers (e.g., user database and/or filesystem), but also data in transit between the cloud and the user(s) which could be transmitted over the Internet or via mobile media (In many circumstances, it would be more cost-effective and convenient to move large volumes of data to the cloud by mobile media like archive tapes than transmitting over the Internet.). Data objects may also include user identity information created by the user management model, service audit data produced by the auditing model, service profile information used to describe the service instance(s), temporary runtime data generated by the instance(s), and many other application data. Different types of data would be of different value and hence have different security implication to cloud users. For example, user database at rest in cloud servers may be of the core value for cloud users and thus require strong protection to guarantee data confidentiality, integrity and availability. User identity information can contain *Personally Identifiable Information* (PII) and has impact on user privacy. Therefore, just authorized users should be allowed to access user identity information. Service audit data provide the evidences related to compliances and the fulfillment of *Service Level Agreement* (SLA), and should not be maliciously manipulated. Service profile information could help attackers locate and identify the service instances and should be well protected. Temporary runtime data may contain critical data related to user business and should be segregated during runtime and securely destroyed after runtime.

Security Services: The basic security services for information security include assurance of data *Confidentiality*, *Integrity*, and *Availability* (CIA). In Cloud Computing, the issue of data security becomes more complicated because of the intrinsic cloud characteristics. Before potential cloud users are able to safely move their applications/data to the cloud, a suit of security services would be in place which we can identify as follows (not necessarily all needed in a specific application):

1) *Data confidentiality assurance:* This service protects data from being disclosed to illegitimate parties. In Cloud Computing, data confidentiality is a basic security service to be in place. Although different applications may have different requirements in terms of what kind of data need confidentiality protection, this security service could be applicable to all the data objects discussed above.

2) *Data integrity protection:* This service protects data from malicious modification. When having outsource their data to remote cloud servers, cloud users must have a way to check whether or not their data at rest or in transit are intact. Such a security service would be of the core value to cloud users. When auditing cloud services, it is also critical to guarantee that all the audit data are authentic since these data would be of legal concerns. This security service is also applicable to other data objects discussed above.

3) *Guarantee of data availability:* This service assures that data stored in the cloud are available on each user retrieval request. This service is particularly important for data at rest in cloud servers and related to the fulfillment of Service

Level Agreement. For long-term data storage services, data availability assurance is of more importance because of the increasing possibility of data damage or loss over the time.

4) *Secure data access:* This security service is to limit the disclosure of data content to authorized users. In practical applications, disclosing application data to unauthorized users may threat the cloud user's business goal. In mission-critical applications, inappropriate disclosure of sensitive data can have juristic concerns. For better protection on sensitive data, cloud users may need fine-grained data access control in the sense that different users may have access to different set of data. This security service is applicable to most of the data objects addressed above.

5) *Regulations and compliances:* In practical application scenarios, storage and access of sensitive data may have to comply specific compliance. For example, disclosure of health records may be limited by the Health Insurance Portability and Accountability Act (HIPAA) [12]. In addition to this, the geographic location of data would frequently be of concern due to export-law violation issues. Cloud users should thoroughly review these regulation and compliance issues before moving their data into the cloud.

6) *Service audition:* This service provides a way for cloud users to monitor how their data are accessed and is critical for compliance enforcement. In the case of local storage, it is not hard to audit the system. In Cloud Computing, however, it requires the service provider to support trustworthy transparency of data access.

Adversary Model: In Cloud Computing, cloud users move applications from within their enterprise/organization boundary into the open cloud. By doing so, cloud users lose physical control over their data. In such an open environment, cloud users may confront all kinds of attacks. Although there might be various categorization methods for the attacks, it is useful to identify where these attackers come from and what kind of attacks they can launch. Based on this criteria we divide attackers in Cloud Computing as two types: *insiders* and *outsiders*.

1) *Insiders:* The insiders refer to the subjects within the system. They could be malicious employees with authorized access privileges inside of the cloud user's organization, malicious employees at the Cloud Service Provider's side, and even the Cloud Service Provider itself. In practice, an employee, at both the cloud user side and the Cloud Service Provider side, could become malicious for reasons such as economic benefits. These insider attackers can launch serious attacks such as learning other cloud users' passwords or authentication information, obtaining control of the virtual machines, logging all the communication of other cloud users, and even abusing their access privilege to help unauthorized users gain access to sensitive information. Although in practical deployments cloud users may have to establish trust relationship with cloud service providers, the occasionally possible misbehavior of cloud server can be anyone or the combination of the following: 1) potentially decide to hide data

corruptions caused by server hacks or Byzantine failures to maintain reputation; 2) neglect to keep or deliberately delete some rarely accessed data files so as to save resources; 3) try to acquire as much data information as possible by eavesdropping and monitoring the network traffic; 4) even collude with a small number of malicious users for the purpose of harvesting the data file contents when it is highly beneficial. Cloud users should thoroughly review all the potential vulnerabilities and protect their assets on any intentional or inadvertent security breach. More specifically, cloud users should be aware what kind of security services these providers can offer and how the providers implement these security services. Verification mechanisms should be available to cloud users for verifying the security services provided by the service providers. For valuable and/or sensitive data, cloud users may also have to implement their own security protection mechanisms, e.g., strong cryptographic protection, in addition to whatever security service cloud service providers offer.

2) *Outsiders*: By moving data into the cloud users will lose their conventional network perimeters and expose their data in an open system. Just like any other open systems, Cloud Computing could be vulnerable to malicious attacks from the Internet. This is because Cloud Computing usually does not limit the type of user when providing services. For example, in Amazon EC2 anybody can register as a cloud user if they provide their credit card information. Malicious attackers can easily log into the cloud and launch attacks. More specifically, outsider attackers can launch both passive attacks such as eavesdropping the network traffic, and active attacks like phishing legitimate users' credential, manipulating network traffic and probing the cloud structure. For some cloud services, outsider attackers can launch very severe attacks by taking advantage of the system flaw. For example, by launching cross virtual machine attacks [49], attackers are able to monitor VMs from their co-resident VMs and threaten their security. By bluepilling/subverting hypervisors [4,8], attackers are even able to control the whole system stack above the hypervisor. To address outsider attacks, cloud service providers have the responsibility to secure their cloud infrastructure, isolate user application in the cloud, patch system flaws timely, and notify cloud users with any discovered security risks. Cloud users should strictly abide to the security guidance when using cloud services for the purpose of reducing the possibility of security breach. Cloud users need to negotiate recovery and backup mechanism with service providers for better security protection.

System Model: From the high level, the system architecture for cloud computing data services can be depicted as figure 1. At its core, the architecture consists of four different entities: the data owner, who is also a cloud user and has large amount of data files to be stored in the cloud; the cloud user, who is authorized by the data owner to access his data files; the cloud server, which is managed by cloud service providers to provide data storage and data sharing services and has significant storage space and computation resources; the third party auditor (TPA), which is the trusted entity that assesses the cloud storage security on

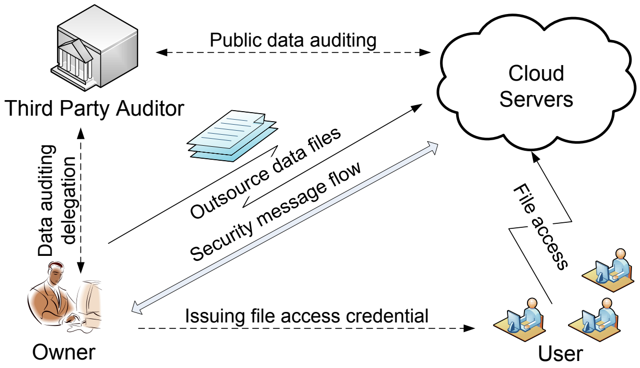

Fig. 1. The architecture of cloud data service

behalf of the data owner upon request. In the cloud paradigm, the data owner may represent either the individual or the enterprise customer, who relies on the cloud server for remote data storage and maintenance, and thus is relieved from the burden of building and maintaining local storage infrastructure. In most cases, cloud service providers also provides benefits like availability (being able to access data from anywhere) and relative low cost (paying as function of needs). Cloud service providers implement the necessary security protection mechanisms for data services. The data owners can also implement their own security protection mechanisms for better security protection such as end-to-end security. Instead of auditing the cloud services by themselves, data owners (cloud users) may delegate all the auditing tasks to the third-party auditors.

2.1 Data Confidentiality

Data confidentiality is a basic security service for data protection. In cloud computing, providing such a service is of great importance because of the following characteristics of cloud computing that will increase the risk of data breach: remote data storage, lacking of network perimeter, third-party cloud service providers, multi-tenancy and massive sharing of infrastructure. In addition, since Cloud Computing, by its very nature, integrates many existing and new techniques, it will inevitably introduce new security risks due to both system design flaws and its implementation flaws. The challenges in providing satisfying security assurance in terms of data confidentiality exist in the following folds: data security versus usability, system scalability and dynamics. To ensure data confidentiality, the most straightforward method is to encrypt all the sensitive data when being stored, processed, and transmitted by cloud servers. When data encryption provides satisfying level of security protection, there are several subtle and challenging issues to be addressed which we can list as follows

- how to efficiently distribute data decryption keys to authorized cloud users?
- how to efficiently deal with user dynamics, in particular user revocation?

- how to efficiently handle data dynamics in terms of data modification?
- how to guarantee accountability of users?
- how to enable computing over encrypted data?

The first three questions are related to the issue of key management. In particular, efficient key distribution is always a sophisticated issue in large-scale application scenarios. As the very characteristic of Cloud Computing is to provide elastic and scalable computing resources to potentially large scale applications, it is very possibly that there will be a large volume data and a large number of users presented in the system. It is challenging to efficiently and securely distribute the key(s) to authorized users when the users enter the system as it usually requires the data owner to stay online providing the key distribution service. More than this, user revocation is anther prohibiting issue as it is in traditional cryptography. In many cases, user revocation will involve broadcasting with all the users in the system and/or re-encryption of existing data stored in the cloud. Similarly, data dynamics would also involve data re-encryption and/or re-distribution of decryption key(s), which would represent a huge computation and communication overhead in the system. In large-scale systems an ideal solution is those that can make data encryption operation independent to, or having minimal impact on, the process of key distribution in the sense that, any modification/re-encryption of data will not introduce update/re-distribution of decryption key. For this purpose special attention should be paid to the system design as well as the choice of the underlying cryptographic primitive(s). Such an issue is particularly related to cryptography based data access control. In section 2.4, we will present detailed discussion on the issue as well as the related solutions.

For encryption based solutions, data access privilege is granted by possession of the corresponding decryption key(s). This opens up the door for authorized but malicious users to abuse their access privilege by re-distributing data decryption keys to unauthorized users. To prevent such key abuse from happening, one way is to secure the data decryption key with temper-resistant hardware on user's side so that the potentially malicious user is not able to access the key while enabling her/him to decrypt data. Temper-resistant devices are usually designed in the way that, when interfered with, they will zeroise the sensitive data, e.g., the decryption key, or the chip just fractures. In this way, the only way that the malicious user is able to abuse the key is by sharing the physical device with others, which greatly limit the ability of attackers. Nevertheless, as the malicious attacker physically possesses the device, it is possible to launch clever attacks which can bypass the protection mechanism inside of the device, e.g., chosen message attacks, fingerprinting attacks [18] and etc. Alternatively, people can use reactive instead of proactive techniques for addressing the issue of key abuse. More specifically, one can take action upon any detected event of key abuse (the detection process can be various, be it technical or non-technical). A well-accepted solution for reactively thwarting key abuse is to go through a process of data forensics and enable the authority to identify the key abuser and generate the corresponding evidence upon detected key abuse. In broadcast encryption [27, 40, 46] such techniques are usually called traitor tracing [26, 29, 31, 44, 59]. The main issue with this technique is its efficiency and scalability.

Another important issue is to enable processing over encrypted data. This is an extremely challenging issue as there are various types of data processing operations. Enabling computing over encrypted data for some operations may logically contradict with the goal of data confidentiality by its very nature. In specific applications, one needs to clearly define to which extend data confidentiality should be achieved in the sense that which kind of information related to the data can be disclosed and which can not. For example, given the encrypted version of two numbers, one may not be able to know the exact numbers without decrypting them. But she may be able to tell the order of the two numbers given their encrypted versions with encryption schemes like order preserving encryption [17]. In this case, the order between the two numbers may be not a piece of sensitive information and one can sort the numbers given their encrypted version without knowing their original value. Similarly, for keyword search one may want to hide the actual keywords but do not need to protect the pattern of the search queries. In the literature, many interesting cryptographic primitives have been proposed for supporting operations over encrypted data, e.g., searchable encryption [23,30,33,39,54], homomorphic encryption [9], format-preserving encryption [24], order-preserving encryption [17] and etc. Recently, Gentry proposed a fully homomorphic encryption scheme [37] which enables us to evaluate arbitrary functions over encrypted data without being able to decrypt. However, its current construction is far from practical due to its complexity and can just serve as a theoretical feasibility.

Data encryption provides an effective way for protecting data confidentiality. The price of it is the degradation of efficiency and flexibility for data processing. An alternative way to deal with data confidentiality is to remove sensitive data and just store non-sensitive data in the cloud. For example, when dealing with data containing personal identifiable information (PII), one would remove these uniquely identifiable information to protect user privacy. This technique is similar to the ideas of k-anonymity and its enhancements [16, 32, 45, 50, 55] (e.g., l-diversity, t-closeness) in database. As compared to data encryption, this method preserves the efficiency and flexibility for data processing. This method also greatly simplifies the complexity of system management since there is no longer the need for key distribution and management. The main downside of this solution is that it will cause information loss by removing the sensitive information. In many application situations, this process will make the data useless though data confidentiality is preserved.

In addition to the above two methods for data confidentiality, there is another method which is so-called ”information-centric” protection [13]. With this method, data is self-describing and defending: Data are encrypted with a kind of usage policy. Whenever the data is accessed, the system will run a program that checks the environment by the data usage policy. If the verifying program makes sure that the environment is secure enough, it will create a secure virtualization environment and decrypt the data. The data can be accessed by applications in plaintext in this secure environment. This ”information-centric” protection mechanism is based on the Trusted Computing technique to check the environment security. This method provides a novel idea of protecting data security while preserving usability of data. However, in practical deployment, it would be not trivial to implement such a ”information-centric” protection mechanism. Moreover, when the scheme verify the environment security at the time of data extraction, it does not provide security protection or detection when the virtualization environment is running. In particular, it is still possible to launch attacks during the runtime of the virtualization environment, e.g., launching cross VM attacks.

In Cloud Computing, various types of data in different applications may have their specific nature in terms of dynamics, data processing operations, sensitivity and etc. For example, in some application cloud users may store a large volume of data on cloud servers for persistence and will be just queried and/or retrieved by themselves and other authorized users. In some other applications, data stored in the cloud will be frequently accessed and processed by applications running on the cloud servers. It is not practical to give a uniform solution for protecting data in all these applications. Instead, we may want to choose a data protection method according to the nature of data. For this purpose, it is necessary to first classify data according to the pre-defined natures. For relatively static data at rest on the cloud, e.g., log data of a system, we may need to encrypt the data in bulky in the way that allows simple data query and retrieval operations. Existing techniques such as searchable encryption [23,30,33,39,54] can be applied to this type of data. For data frequently subjected to modification, we may want to encrypt the data block by block so that updating one data block does not other data blocks. Techniques such as hierarchical key tree [48] might be suitable for this case. However, one must be aware of the tradeoff between data confidentiality and usability and choose an appropriate data encryption scheme. There might be the situation in which no suitable data encryption scheme is available to simultaneously meet both the goal of security and that of usability for the cloud user. In such a case, cloud users might have to either rely on legal practices such as the service level agreement for data protection, or store the data on a more trustworthy cloud environment, e.g., using a private or community cloud instead of a public cloud. For sensitive data such as personally identifiable information, it would cause legal issue if disclosed to any unauthorized party. In such a case, one may have to trade usability for data security when these two goals can not be achieved simultaneously, or store the data on a trustworthy cloud. During the runtime of virtual machine instances, they may also access or generate temporary sensitive data. It is important that the VM instances destroy all the sensitive temporary data after their execution.

2.2 Data Integrity Protection

Data integrity is another important security issue in cloud computing. Such a security assurance is necessary not only for communications between cloud users and cloud servers, but also for data at rest on cloud servers. In particular, cloud users may have great concerns on data integrity when outsourcing valuable data assets in the cloud for storage. The possible long lifetime of outsourced data would make it more likely vulnerable to intentional or inadvertent modification, corruption, or deletion, be it caused by careless system maintenance or for the purpose of cost saving. While the issue of data integrity for communications can be addressed with off-the-shelf techniques such as message integrity code, that for data storage seems to be more cumbersome because of the following facts:

First, cloud users may not be willing to fully rely on cloud service providers for providing data integrity protection. This is because cloud services are usually provided by third-party providers who are not necessary in the same trust domain of the cloud users. Although cloud users can establish the trust relationship with cloud service providers via mechanisms such as service level agreement, this practice is still not able to give cloud users the full confidence on data integrity due to the possible occasional purposive or inadvertent misbehaviors from the cloud service providers. Such misbehaviors could be the result of providers' decision to hide data corruptions caused by server hacks or Byzantine failures to maintain reputation, or their neglect of keeping or deliberate deletion of some rarely accessed data files so as to save resources. Given such a fact, cloud users would like to protect integrity of their own data assets by themselves or through their trusted agents.

Second, data integrity service should be provided in the timely manner. This is because in practical applications it is usually too late for cloud users to find out data corruption when they are actually retrieving the data. This is particularly true for long term storage of large volume data, in which many portion/blocks of data could be seldom accessed in a long period of time. When some portion of data is found corrupted on retrieval, it could be impossible to recover as information needed for recovery may have been lost during the long interval. For example, disk recovery is usually not possible when the physical disk location of the data has been overwritten by new data. The longer it is since data corruption, the more likely it is that the data can not be recovered. To provide timely data integrity service to cloud users to reduce the risk of data corruption of lost, it is necessary to supply them with efficient data integrity check mechanism, which should be able to process the possibly large volume of data without introducing too much computation/communication overhead.

Third, the "self-served" data integrity check requires not only the active involvement of cloud users, but also the necessary expertise and computing power of them. In cloud computing, however, cloud users vary greatly in their available resources and expertise. It turns out that most cloud users may not have the ability to perform data integrity check by themselves. A reasonable solution to this issue is to let the cloud users delegate the task of data integrity check to a third professional party of their trust (i.e., a third party auditor (TPA)) which has the necessary resources and expertise. During this delegation process, however, the tension exists between TPA verifiability and data privacy. This is because in practical applications user data may contain sensitive information, which cloud users may not want to disclose to TPA though they trust the TPA in performing data integrity check. It is desirable, but challenging, to provide such a solution to cloud users that allows them to delegate the task of data integrity check without violate their data privacy.

Fourth, as data stored on cloud servers may subject to modification by cloud users, the data integrity mechanism should efficiently support such data dynamics. It means that, the overhead for supporting data dynamics introduced to both cloud servers and the verifier, be it cloud users themselves or a third party auditor, should be in a reasonable range. Ideally, modification of one block of data should not affect other data blocks in terms of data integrity protection.

Preferably, a data integrity protection mechanism should address all these issues, i.e., it should support frequent data integrity check on large volume of data when allowing third-party verification and data dynamics. To provide strong protection on data integrity, cryptographic methods can be applied. Intuitively, one may want to use message authentication codes (MAC) for data integrity as follows. Initially, data owners (cloud users) locally generate a small amount MACs for the data files to be outsourced and maintain a local copy of these MACs. Whenever the data owner needs to retrieve the file, he can verify the data integrity by re-calculating the MAC of the received data file and comparing it to the locally pre-computed value. In case the size of data file is large, a hash tree [10] can be employed, where the leaves are hashes of data blocks and internal nodes are hashes of their children of the tree. The data owner only needs to store the root nodes of the hash tree to authenticate his received data. Whenever the data owner needs to retrieve a block or blocks of data, the server sends the data bock(s) as well as the necessary internal hash nodes, which can be either computed on the fly or pre-computed by the cloud servers, to the data owner. The data owner calculates the hash value(s) of the received data block(s), with which he can compute the root hash given other internal hash nodes sent by the server. Data integrity is verified against the stored root hash. Given the second pre-image resistance property of the hash function, security of the data integrity verification mechanism can be achieved. While this method allows data owners to verify the correctness of the received data from cloud, it does not give any assurance about the correctness of other outsourced data. In other words, it does not give any guarantee that the data in the cloud are all actually *intact*, unless the data are all downloaded by the owner. Because the amount of cloud data can be huge, it would be quite impractical for data owner to retrieve all of his data just in order to verify the data is still correct. In case that the data auditing task is delegated to TPA, this method inevitably violates our suggested requirements, including: large auditing cost for cloud server (for accessing and transferring the whole data), and data privacy exposure to TPA (for retrieving local copy of data).

To avoid retrieving data from the cloud server, a simple improvement to this straightforward solution can be performed as follows: Before data outsourcing, the owner chooses a set of random MAC keys, pre-computes the MACs for the whole data file, and publishes these verification metadata to TPA. The TPA can each time reveal a secret MAC key to the cloud server and ask for a fresh keyed MAC for comparison. In this way, the bandwidth cost for each auditing is only at bit-length level (keys and MACs). However, a particular drawback is the number of times a data file can be verified is limited by the number of secret keys that must be a fixed priori, which might introduce additional on-line burden to the data owner: Once all possible secret keys are exhausted, data owner then has to retrieve data from the server in order to re-compute and re-publish new MACs to TPA. Another drawback of this improved approach is its inability to deal with data dynamics, as any data change would make those pre-computed MACs unusable at all.

To significantly reduce the arbitrarily large communication overhead for public verifiability without introducing the on-line burden to data owner, Wang et. al [58] employ the homomorphic authenticator technique [20,51]. Homomorphic authenticators are unforgeable metadata generated from individual data blocks, which can be securely aggregated in such a way to assure a verifier that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. Using this technique requires additional information encoded along with the data before outsourcing. Specifically, data file is divided into n blocks m_i $(i = 1, \ldots, n)$ and each block m_i has a corresponding homomorphic authenticator σ_i computed as its metadata to ensure the integrity. Specifically, σ_i is computed as $\sigma_i = (H(m_i) \cdot u^{m_i})^{\alpha}$, where H is a cryptographic hash function, u is random number, and α is a system master secret defined on the integer field being used. Every time to verify that the cloud server is honestly storing the data, data owner or TPA can submit challenges $chal = \{(i, \nu_i)\}$ for sampling a set of randomly selected blocks, where $\{\nu_i\}$ can be arbitrary weights. Due to the nice property of homomorphic authenticator, server only needs to response a linear combination of the sampled data blocks $\mu = \sum_i \nu_i \cdot m_i$, as well as an aggregated authenticator $\sigma = \prod_i \sigma_i^{\nu_i}$, both computed from $\{m_i, \sigma_i, \nu_i\}_{i \in chal}$. Once the response of μ and σ is verified by TPA, then high probabilistic guarantee on large fraction of cloud data correctness can be obtained. Because off-the-shelf error-correcting code technique can be adopted before data outsourcing [42,51], large fraction of correct cloud data would be sufficient to recover the whole data.

2.3 Data Availability

The unlimited and elastic resources offered by cloud computing would greatly improve cloud users' ability in data storage and processing. For example, by creating multiple replicas of data in the cloud, cloud users can enjoy robust data storage which may not be available locally due to limited resources. To provide high quality data services to their own customers, cloud users (data owners) may replicate data on geographically distributed cloud servers and allow their customers to access data efficiently via local cloud servers (the use of which is similar to that in content distribution networks (CDNs)). Cloud users can also save the effort for data maintenance by delegating it to the cloud service provider who may have more expertise in doing this. In brief, with cloud computing cloud users would be able to operate high quality and large scale data services with minimal local deployment and maintenance effort. During this process, one of the main concerns from the data user would be data availability in the following sense:

First, cloud computing should guarantee that user data stored in the cloud can be immediately available whenever retrieved. In particular, it is important to assure the availability of data services and hence business continuity (BC) of cloud users in case of temporarily or permanently cloud outage. In the real life, the catastrophic events are more likely to happen due the outage of cloud such as communication outage, power outage, bankrupt of the cloud service provider, etc.

Second, cloud computing should provide the agreed service quality to cloud users. For example, for redundant data storage the cloud user may need to store k physical replicas in the cloud. In this case, it is important to guarantee that the k replicas are indeed available in the cloud. This is also true when cloud users need to storage data replicas on geographically distributed cloud servers for quality of service. In these cases, the data are available but the quality of service would be degraded when the cloud service provider is not following the agreement. Special attention should be paid to the case of long term data storage. In such a scenario, it is important to assure that the cloud service provider does not break the service agreement by secretly moving less frequently accessed data from onsite storage to secondary storage. Such a break of the service agreement is usually easy to be ignored but will potentially degrade the service quality for cloud users. In many application scenarios, the quality of data service such as the speed of data access is very critical to the business success. It is important to promise data availability at every aspect of data services as discussed.

In order for providing dependable and trustworthy cloud data service to cloud users, appropriate mechanism(s) should be in place for cloud users to efficiently verify the availability of their data. For the first case, the common practice is to let cloud users store multiple replicas of data on distributed cloud servers or in multiple clouds. For assurance of data availability, we just need to provide a way for cloud users to make sure that the multiple copies of data do exist on designated clouds or cloud servers, which is one of the goals for the second case. For the second case, the essential issue is how to establish the trust between the cloud service provider and cloud users in the sense that the specific share of data does exist on designated storage sites/regions. Service level agreements (SLAs) can be adopted to achieve this goal as is used in many application systems. For assuring appropriate use of SLAs and avoiding possible disputes, effective verification mechanism(s) should be in place. Along this direction, the literature has proposed several cryptographic mechanisms to provide strong security protection on data availability in cloud computing. Among these solutions are two most promising ones: “provable data possession (PDP)” [20] and “proof of retrievability (PoR)” [42, 51].

In all these works great efforts are made to design solutions that meet various requirements: high scheme efficiency, stateless verification, unbounded use of queries and retrievability of data, etc. Considering the role of the verifier in the model, existing PDP or PoR schemes fall into two categories: private verifiability and public verifiability. While schemes with private verifiability can achieve higher scheme efficiency, public verifiability allows anyone, not just the client (data owner), to challenge the cloud server for correctness of data storage while keeping no private information. Clients are then able to delegate the evaluation of the service performance to an independent third party verifier, without devotion of their computation resources. Pubic verifiability is very important because cloud clients vary tremendously regarding to their computing capabilities. Many of them, such as individuals, may not be able to afford the overhead of performing frequent integrity checks by themselves. It seems more rational to equip the verification protocol with public verifiability, which is expected to play a more important role in achieving economies of scale for Cloud Computing. Moreover, for efficiency consideration, download of the original outsourced data should not be required by the verifier for the verification purpose.

Another major concern among previous designs is on how to support dynamic data operation for cloud data storage applications. That is, the data availability protocols should not only consider static data remotely stored in the cloud, but also data that may be updated, e.g., through block modification, deletion and insertion. The state-of-the-art in the context of remote data storage mainly focus on static data files. Efficient protocol for verifying the availability of data with dynamic updates is needed. The following is the brief summary of existing solutions.

Ateniese et al. [20] are the first to consider public data verification in their defined “provable data possession (PDP) model for ensuring possession of data files on untrusted storages. Their scheme utilizes the RSA-based homomorphic linear authenticators for auditing outsourced data and suggests randomly sampling a few blocks of the file. The public data verification in their scheme demands the linear combination of sampled blocks exposed to external auditor. Juels et al. [42] describe a “proof of retrievability (PoR) model, where spot-checking and error-correcting codes are used to ensure both possession and retrievability of data files on remote archive service systems. The number of verification challenges a user can perform in this scheme is fixed a priori. Public data verification is not supported in their main scheme. Public PoR is supported in their extended solution based on the technique of Merkle-tree. But this approach only works with encrypted data. Dodis et al. [35] give a study on different variants of PoR with private data verification. Shacham et al. [51] design an improved PoR scheme built from BLS signatures [28] with full proofs of security in the security model defined in [42]. Similar to the construction in [20], they use publicly verifiable homomorphic linear authenticators that are built from provably secure BLS signatures. Based on the elegant BLS construction, a compact and public verifiable scheme is obtained. Shah et al. [52, 53] propose allowing a TPA to keep online storage honest by first encrypting the data then sending a number of pre-computed symmetric-keyed hashes over the encrypted data to the verifier. The verifier checks both the integrity of the data file and the servers possession of a previously committed decryption key. This scheme only works for encrypted files, and it suffers from the auditor statefulness and bounded usage, which may potentially bring in online burden to users when the keyed hashes are used up.

In another related work, Ateniese et al. [22] propose a partially dynamic version of the prior PDP scheme, using only symmetric key cryptography but with a bounded number of audits. In [56], Wang et al. consider a similar support for partial dynamic data storage in a distributed scenario with additional feature of data error localization. In a subsequent work, Wang et al. [57] propose to combine BLS-based HLA with MHT to support both public data verification and full data dynamics. Almost simultaneously, Erway et al. [36] developed a skip lists based scheme to enable provable data possession with full dynamics support. The verification in these two protocols requires the linear combination of sampled blocks just as [20,51]. For these schemes, a promising future work is to consider batch verification, which can greatly reduce the computation cost on the third party verifier when coping with a large number of verification delegations.

2.4 Secure Data Access

In Cloud Computing, various sensitive data information pooled in the cloud demands the cloud data storage and sharing service to be responsible for secure, efficient and reliable distribution of data content to potentially large number of authorized users on behalf of data owners. To address this issue, one way is to rely on cloud servers and let them implement access control mechanisms such as Role-Based Access Control (RBAC) [6]. As access control mechanisms like RBAC are mature techniques with the capability to deal with fine-grained access control in large scale systems, the goal of data access control can be effectively achieved. The main issue with this solution exists in two folds: Firstly, in access control mechanisms like RBAC, the server need to have full access to all the user data when fulfilling their tasks. This requires that cloud users should fully trust the cloud servers, and hence the Cloud Service Provider (or even their employees). Secondly, due to the existence of serious outsider attacks at different layer of the system stack, e.g., cross VM attacks and bluepilling/subverting hypervisor attacks, it requires that cloud servers fulfilling the access control tasks should be well protected at every layer. In practice, this could be a challenging task considering the fact that cloud servers reside in such an open Internet.

An alternative way to provide secure data access service is based on cryptographic methods. In this type of solutions, the data owner (cloud user) encrypts data before storing them in the cloud and retain the secret key to himself/herself. Data access is granted by distributing the data decryption key to the authorized users. In this way, we achieve "end-to-end" security without disclosing data content to cloud servers. Different from the first method, this type of solutions do not demand the cloud users to fully trust the cloud server (and hence the Cloud Service Provider and its employees). However, cloud server can still take full charge of the management of the outsourced encrypted data since they are not able to compromise the data confidentiality. What makes the problem challenging is the enforcement of fine-grained authorization policies, the support of policy updates in dynamic scenarios, and the system scalability, while maintaining low level complexity of key management and data encryption. Thus, the main research work along this direction is to simultaneously achieve fine-grainedness, scalability and data confidentiality of data access control in cloud computing, without introducing significant computation burden on the data owner.

In the literature, existing related mechanisms [34, 38, 43] can be found in the areas of shared cryptographic file systems and access control of outsourced data, which have drawn great attention recently. These application scenarios share the similar characteristic with Cloud Computing in terms of untrustworthiness of data storage, which makes these proposed mechanisms potentially applicable to Cloud Computing. In [43], Kallahalla et al. proposed Plutus as a cryptographic file system to secure file storage on untrusted servers. Plutus groups a set of files with similar sharing attributes as a file-group and associates each file-group with a symmetric lockbox-key. Each file is encrypted using a unique file- blcok key which is further encrypted with the lockbox-key of the file-group to which the file belongs. If the owner wants to share a file-group, he just delivers the corresponding lockbox-key to users. One nice property of Plutus is its simplicity in key management since just a small number keys are involved, which makes the system very efficient. Plutus is particularly applicable to the case of coarse-grained access control in which data files/users can be categorized into a small number of groups. But it is not suitable for the case of fine-grained access control. This is because the complexity of key management is proportional to the total number of file-groups which could be huge in the case of fine-grained access control.

In [38], Goh et al. proposed SiRiUS which is layered over existing file systems such as NFS but provides end-to-end security. For the purpose of access control, SiRiUS attaches each file with a meta data file that contains the file's access control list (ACL). SiRiUS encrypts the file with a file encryption key (FEK) and then encrypts the FEK with each individual authorized user's public key for fine-grained data sharing. The extended version of SiRiUS uses NNL broadcast encryption algorithm [46] to encrypt the FEK of each file. One nice property of SiRiUS is that the number of keys that each user need to keep is minimal, which indicates a minimal key storage complexity as well as a minimal data decryption complexity at the user's side. Moreover, as this scheme is built on top of NNL broadcast encryption algorithm, it inherits all the security properties of NNL scheme such as efficient traitor tracing. When applied in large scale applications, SiRiUS requires that the number of unauthorized users should be relatively small as compared to the total number of users. This is because the data encryption complexity (and hence the ciphertext size) is proportional to the number of unauthorized users.

Ateniese et al. [21] proposed a secure distributed storage scheme based on proxy re-encryption. In this scheme, the data owner encrypts blocks of content with symmetric content keys. The content keys are all encrypted with a master public key, which can only be decrypted by the master private key kept by the data owner. The data owner uses his master private key and users public key to generate proxy re-encryption keys, with which the semi-trusted server

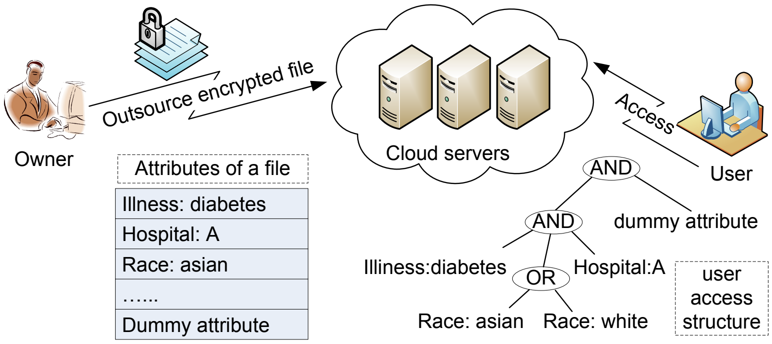

Fig. 2. An example for attribute-based encryption

can then convert the ciphertext into that for a specific granted user and fulfill the task of access control enforcement. One nice property of this solution is that it is very efficient in dealing with user management, especially for user revocation. Actually, to revoke users, the server just needs to keep a revoked user list and revoke a user by refusing to convert the ciphertext for him/her. When applying this scheme in Cloud Computing, it requires that there is no collusion between the server and any authorized but malicious user. This is because a single such collusion would expose decryption keys of all the encrypted data and compromise data security of the system completely. For this sake, this solution will be more applicable to application scenarios in which cloud servers can be trusted by the data owner (a cloud user). Different from the first solution, this scheme is vulnerable to outsider attacks since on the cloud servers just ciphetexts of data are stored. Data confidentiality can be protected even if the server is compromised by outsider attackers.

In [34], Vimercati et al. proposed a solution for securing data storage on untrusted servers based on key derivation methods. In this proposed scheme, each file is encrypted with a symmetric key and each user is assigned a secret key. To grant the access privilege for a user, the owner creates corresponding public tokens from which, together with his secret key, the user is able to derive decryption keys of desired files. The owner then transmits these public tokens to the semi-trusted server and delegates the task of token distribution to it. Just given these public tokens, the server is not able to derive the decryption key of any file. This solution introduces a minimal number of secret key per user and a minimal number of encryption key for each file.

In order to achieve secure, scalable and fine-grained data sharing on outsourced data in the cloud, Yu et al. [60] proposed a scheme which exploits the following three advanced cryptographic techniques: attributed-based encryption (ABE) [41], proxy re-encryption (PRE) [25] and lazy re-encryption [43]. The proposed scheme is partially based on the fact that, in practical application scenarios each data file can be associated with a set of attributes which are meaningful in the context of interest. The access structure of each user can thus be defined as a unique logical expression over these attributes to reflect the scope of data files that the user is allowed to access. As the logical expression can represent any desired data file set, fine-grainedness of data access control is achieved. To enforce these access structures, [60] defines a public key for each attribute and encrypts files using the public keys corresponding to their attributes. User secret keys are defined to reflect their access structures so that a user is able to decrypt a ciphertext if and only if the data file attributes satisfy his access structure. Figure 2 is a toy example for the case of health record exchange and sharing. In this example, the health record is associated with a set of attributes (non-PII information). User access structure is defined in the way that just allows the user to access health records for asian or white patients treated by hospital A with diabetes. Such a definition of access privilege is in accordance with the descriptive way seen in our real life and could be very expressive. Different from traditional access control mechanisms such as RBAC, the access policy in this scheme is enforced merely by encryption/decryption. According to [60], such a design also brings about the efficiency benefit in that, 1) the complexity of data file encryption is just related the number of attributes associated to the file, which is independent to the number of users in the system, and 2) the functionality of this scheme is similar to RBAC or ABAC in the sense that it separates the process of data-attribution association from user-privilege binding. With such a design, the file creation/deletion and new user grant operations just affect current file/user without involving system-wide data file update or re-keying. To resolve the challenging issue of user revocation, the proposed scheme enables the data owner to delegate tasks of data file re-encryption and user secret key update to cloud servers without disclosing data file plaintexts or user access privilege information. They achieve these design goals by exploiting a novel cryptographic primitive, namely key policy attribute-based encryption (KP-ABE) [41], and combine it with the technique of proxy re- encryption (PRE) and lazy re-encryption.

2.5 Compliances and Regulations

For mission-critical applications, store and access of sensitive data is strictly regulated. Both the data owner and the cloud service provider should be aware of the underlying regulations/compliances before moving sensitive data into the cloud. Examples of these compliances are as follows.

- *Health Insurance Portability and Accountability Act (HIPAA)* The privacy rule of HIPAA [14] regulates the correct use and disclosure of private health information held by “covered entities” as defined HIPAA and the Department of Health and Human Services (HHS). In particular, it defines 18 types of Protected Health Information (PHI) held by a covered entity and sets up regulations for the appropriate use and disclosure of PHI. PHI usually refers to information that can be linked to an individual. But it is frequently interpreted broadly and can include all parts in an individual’s health record and/or payment history.

- *Federal Information Security Management Act (FISMA)* FISM [2] intends to regulate the information security for U.S. federal government agencies and/or their contractors. A security framework is defined for information security and must be followed by all the agent information systems. Under this framework, a suit of security measurements are required, such as information categorization, security control, risk management, etc.
- *SarbanesOxley (SOX)* SOX [11] was enacted for public companies with the primary goal of defending against corporate and accounting scandals in the financial market. 11 titles are include in this act which involve several aspects of financial information security such as integrity, accountability, secure audit, etc.
- *Statement on Auditing Standards No. 70 (SAS 70)* SAS 70 [7] aims to regulate the contracted internal controls for service organizations, including hosted data centers, insurance claims processing companies, credit information processors, etc. It defines a set of criteria for auditing standards that an auditor must employ.

These compliances impose various requirements on data security. In a cloud computing environment, following the compliances can be challenging due to the cloud characteristics, e.g., multi-tenancy, Internet-based services, etc. Certain security certification and/or accreditation of the cloud service provider can be required before the sensitive data can be stored in the cloud. Such a security certification usually involves comprehensive assessment on the service provider with regard to its operational and/or technical security controls. For example, FISMA requires such a certification/accreditation before the agents can utilize cloud services for data processing/storage. Whenever necessary, strong data protection such as cryptographic mechanism should be employed to provide information confidentiality, integrity, availability, and more. Data access should be audited to help detect inappropriate disclosure and/or modification of data. Attention should also be paid to the geographic location of data storage, which can be regulated by the export control regulations.

2.6 Audition

By moving into the cloud, cloud users lose physical control on their data. Cloud users have to rely on cloud service providers for security assurance and quality of service. In most cases, all the requirements from the cloud users will be defined as a Service Level Agreement with cloud service providers. However, it becomes difficult in the cloud for users, or their agents, to audit the services provided by the cloud service providers. This is because cloud service providers usually manage the whole system stack under the level of service provided. To abstract the underlying service, it is necessary for cloud service providers to hide all the lower level implementations and just expose necessary interfaces to cloud users. On the contrary to this, data audit requires that cloud service providers provide transparent services to cloud users. Given these two contradicting goals, it is difficult to audit all parts of the services provided by the cloud service providers.

Notwithstanding the hardness of auditing the cloud services transparently, in practice it might be adequate to verify the services provided in a "black-box" manner in the sense that just the key properties of the services are checked during audit. In particular, for security services it would be enough to verify whether the cloud service providers meet the requirements of the security goals such as data confidentiality, integrity, availability as well as complying to the compliances. In practical systems, it is very possibly that, instead of auditing the services by themselves, cloud users would delegate data audit to third-party federated organization. Such a delegation will not only save the effort for cloud users, but also let them take advantage of expertise from the federated party, e.g., legal expertise. In some cases, it may be necessary to enable public auditing to facilitate the data auditing process. To enable public auditing, the whole service architecture design should not only be secure, but more importantly be practical from a systematic point of view. Keeping this in mind, we can briefly elaborate a set of suggested desirable properties below that satisfy such a design principle. Note that these requirements are ideal goals. They are not necessarily complete yet or even fully achievable with current technologies.

Minimize auditing overhead First and foremost, the overhead imposed on the cloud server by the auditing process must not outweigh its benefits. Such overhead may include both the I/O cost for data accessing and the bandwidth cost for data transferring. Besides, the extra online burden on data owner should also be as low as possible. Ideally, after auditing delegation, data owner should just enjoy the cloud storage service while be worry-free about the storage correctness auditing.

Protect data privacy Data privacy protection has always been an important aspect of service level agreement for cloud storage services. Thus, the implementation of public auditing protocol should not violate the owners data privacy. In other words, TPA should be able to efficiently audit the cloud data storage without demanding the local copy of data or even learning the data content.

Support data dynamics As cloud storage service is not just a data warehouse, owners are subject to dynamically update their data via various application purposes. The design of auditing protocol should incorporate this important feature of data dynamics in Cloud Computing.

Support batch auditing The prevalence of large scale cloud storage service further demands the auditing efficiency. When receiving multiple auditing tasks from different owners delegations, TPA should still be able to handle them in a fast yet cost-effective fashion. This property could essentially enable the scalability of public auditing service even under a storage cloud with large number of data owners.

To achieve these goals, strong cryptographic tools would be employed. For example, we can use the technique POR [42] to audit data availability and integrity in the cloud. However, it could be hard to achieve all these goals simultaneously using existing cryptographic tools. In most cases, we still have to rely on the cloud service provider to audit data.

3 Commercial and Organizational Practices

Although there is still no standardized draft available for data security in Cloud Computing (up to the time when this book was prepared), commercial parties and organizations in the community do provide various proposals. This section will give a brief overview of the security mechanisms adopted by the cloud computing service providers to ensure customers security and privacy.

Amazon Web Services (AWS) Amazon offers its cloud computing services using Amazon Web Services (AWS) [1] which provides a collection of remote computing services, including Amazon S3, EC2, Virtual Private Cloud (VPC), etc. AWS provides a suit of data security solutions at various levels of the system stack.

Amazon S3 provides both bucket- and object-level access control. By default only authorized access is allowed by the bucket and/or object creator (data owner). Access control lists (ACLs) at the bucket/object level are used to grant user access, which is authenticated via user's signature with his/her private key. Bucket and object level ACLs are independent. For example, an object does not automatically inherit ACLs from its bucket. S3 provides the following types of ACLs:

Authorized user with reference to Bucket:
- read: can list the file names, size, modified date
- write: can upload, delete files
- read access control policy(ACP): can check ACL of a bucket
- write ACP: can update ACL.

Authorized user with reference to object:
- read: can download the file.
- write: can replace or delete file
- read ACP: can list ACL of the file
- write ACP: can modify ACL of the file.

S3 defines four types of users who may be granted access permission:
- Owner (account holder)
- Amazon S3 users (by adding amazon.com email address)
- Authenticated User (sharing globally with all S3 users)
- Non Authenticated users (All Users)

In Amazon EC2 security protection is provided at multiple levels such as hypervisor, operating system, virtual machine instance, and API. For hypervisor, EC2 utilizes a highly customized version of the Xen hypervisor and provides four separate privilege modes for CPU. The host OS executes in most-privileged mode, the guest OS runs in a lesser-privileged mode, and applications are in the least privileged mode. Host OS can only be accessed by administrators of AWS. Customers have full access to their guest OS and can configure multi-factor authentication. AWS does not have any access privilege to customer instances and their guest OS. Different instances running on the same physical machine are isolated from each other via the Xen hypervisor. Calls to launch and terminate instances, change firewall parameters, and perform other functions are all signed by the customers Amazon Secret Access Key. Without access to the customers Secret Access Key, Amazon EC2 API calls cannot be made. In addition, API calls can be encrypted with SSL to maintain confidentiality. In addition, Amazon Virtual Private Cloud (VPC) enables to use isolated resources that one owns within the AWS cloud, and then connect those resources directly to your own datacenter using industry-standard encrypted IPsec VPN connections.

Microsoft Windows Azure Microsoft offers its cloud computing services using Windows Azure [5]. Windows Azure is a cloud services operating system that serves as the development, service hosting and service management environment for the Windows Azure platform. Windows Azure provides developers with on-demand compute and storage to host, scale, and manages web applications on the Internet through Microsoft datacenters. Windows Azure provides confidentiality, integrity, and availability of customer data, and will also provide transparent accountability to allow customers and their agents to track administration of applications and infrastructure, by themselves and by Microsoft.

Confidentiality Windows Azure provides data confidentiality via identity and access management, isolation, and encryption. The identity and access management mechanism adopts service management API (SMAPI) to provide web services via the Representational State Transfer (REST) protocol, which runs over SSL and is authenticated with a certificate and private key generated by the customer. In windows azure customers are not granted administrative access to their VMs, and customer software in Windows Azure is restricted to running under a low-privilege account by default. By this the level of attack will be reduced. Communication between Azure internal components are always protected with SSL and via mutual authentication. To assure data confidentiality, Azure provides isolation at different levels: hypervisor, root OS, guest VM, fabric controller. Customer access infrastructure is also logically isolated from customer applications and storage. Critical internal stored or transmitted data can be encrypted with the .NET Cryptographic Service Providers (CSPs) provided by the Azure SDK. Azure's Storage subsystem provides data deletion operations for customers. Successful execution of a delete operation removes all references to the associated data item and it cannot be accessed via the storage APIs.

Integrity The primary mechanism of integrity protection for customer data lies within the Fabric VM design itself. Each VM is connected to three local Virtual Hard Drives (VHDs): The D drive contains one of several versions of the Guest OS, kept up-to-date with relevant patches, selectable by the customer. The E drive contains an image constructed by the FC based on the package provided by the customer. The C drive contains configuration information, paging files, and other storage. The D and E virtual drives are effectively read-only and the access to the C drive is read/write. Only authorized customers accessing their Hosted Services via the Windows Azure Portal or SMAPI (as described earlier) can change the configuration file.

The monitoring Agent (MA) implemented by Azure gathers monitoring and log information from many places and writes it to the log files. It then pushes these log files into a pre-configured Windows azure storage account for audit. Optionally, the customers can also use Monitoring Data Analysis Service which will give the summary and analysis of the log files.

Google App Engine Google App Engine [3] is a platform for developing and hosting web applications in Google managed datacenters. Rather than segregating each customers data onto a single machine or set of machines, Google Apps data from all Google customers is distributed amongst a shared infrastructure composed of Googles many homogeneous machines and located across Googles many data centers. Google Apps uses a distributed file system designed to store large amounts of data across large numbers of computers. Structured data is then stored in a large distributed database built on top of the file system. Data is chunked and replicated over multiple systems such that no one system is a single point of failure. Data chunks are given random file names and are not stored in clear text so they are not humanly readable. Googles security vision is formed around a multi-layered security strategy that provides controls at multiple levels of data storage, access, and transfer. For example, Google provides service-to-service authentication based on X.509 certificates which are issued by Google-internal certificate authority. For deleted data, all the pointers to the data are deleted to make sure that the requested item is deleted from all the active servers. Google Apps also provides several additional security options that can be utilized by a customers domain administrators. The security options include single sign-on, administrator-based single sign-out, policy-enforced secure mail transfer, secure browser connections, etc.

4 Summary

Cloud computing is a promising computing model that has drawn extensive attention from both the industry and academy. Data security is a crucial issue for deploying applications into the cloud. In this chapter, we discussed the basic concepts behind the cloud and introduced the security issues underlying cloud computing. In particular, we defined the model for data security in cloud computing, which includes security services and adversary model. This chapter focuses on a set of crucial security issues pertaining to storing and accessing data in cloud computing: data confidentiality, integrity, availability, secure data access, regulations and compliances, and audition. We analyzed each issue with details and discussed the possible solutions based on existing techniques. During our discussion, we took into account strong attackers such as insiders and covered not only the regular application scenarios but also mission-critical ones. We need to point out that, cloud computing is still at its early stage and data security in cloud computing is an on-going research topic. Rather than offering a complete definition and/or solution for this challenging issue, this chapter aims at providing a reference for practical deployment and further research on this topic.

Exercises

1. Explain the concept of Cloud Computing and list at least five of its most important characteristics.
2. What are the key enabling techniques of Cloud Computing?
3. What are the data security issues in Cloud Computing? What security services are needed to address these issues?
4. What are the issues for protecting data confidentiality with encryption? Explain existing solutions for these issues.
5. Why protecting data integrity and availability is challenging in Cloud Computing? List several existing solutions for this issue.
6. Why server mediated data access control is not enough for Cloud Computing? List several cryptography-based solutions and point out their pros and cons.
7. Describe several compliances and regulations pertaining to data security and privacy. Discuss the challenges to implement these compliances and regulations in the cloud.
8. Give examples of how existing cloud companies solve the issue of data security.
9. Section 2.2 presents a data integrity check mechanism for large data files based on the hash tree. In the hash tree, each leaf node represents a hash value computed for each data block in the file. The internal nodes are hashes of their children. The root node (a hash value) of the hash tree is stored by the data owner. While checking the integrity of a data block, the data owner asks the server to send the block along with the necessary internal hash values. The data owner computes the root hash with data received and compares it with the root hash stored. Equality of the two means that the data block is intact.
   Assume that each data block has the same probability p of being corrupted. The owner is actually able to detect any such corruption of the file with probability $1 - p$ by randomly checking n data blocks. Please determine n for $p = 0.1$. What is n if p is $0.01, 0.001$, and 0.0001 respectively?

References

1. Amazon aws, http://aws.amazon.com
2. Federal information security management act, http://csrc.nist.gov/drivers/documents/FISMA-final.pdf
3. Google app engine, http://code.google.com/appengine
4. The invisible things lab's blog, http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html
5. Microsoft azure, http://www.microsoft.com/windowsazure
6. Role based access control, http://csrc.nist.gov/rbac/rbac-std-ncits.pdf
7. Sas 70, http://sas70.com/sas70_overview.html
8. Wikipedia, http://en.wikipedia.org/wiki/Blue_Pill_(malware)
9. Wikipedia, http://en.wikipedia.org/wiki/Homomorphic_encryption
10. Wikipedia, http://en.wikipedia.org/wiki/Hash_tree
11. Wikipedia, http://en.wikipedia.org/wiki/SarbanesOxley_Act
12. The health insurance portability and accountability act of 1996 (hipaa) (1996), http://www.hhs.gov/ocr/privacy/
13. Emc, information-centric security (2007), http://www.idc.pt/resources/PPTs/2007/IT&Internet_Securit y/12.EMC.pdf
14. The hipaa privacy rule and electronic health information exchange in a networked environment: Individual choice (2008), http://www.hhs.gov/ocr/privacy/hipaa/understanding/special/healthit/individualchoice.pdf
15. The nist definition of cloud computing (2009), csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc
16. A. Machanavajjhala, D. Kifer, J.G., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: in TKDD (2007)
17. Agrawal, R., Srikant, R., Xu, Y.: Order-Preserving Encryption for Numeric Data. In: Proc. of ACM SIGMOD. Paris, France (Jun 2004)
18. Anderson, R.J., Kuhn, M.G.: Low Cost Attacks on Tamper Resistant Devices. In: Proc. of WSP. Paris, France (Apr 1997)
19. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A berkeley view of cloud computing. Tech. rep., University of California, Berkeley (Feb 2009)
20. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable Data Possession at Untrusted Stores. In: Proc. of ACM CCS. Alexandria, VA (Oct 2007)
21. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In: Proc. of NDSS (2005)
22. Ateniese, G., Pietro, R.D., Mancini, L.V., Tsudik, G.: Scalable and Efficient Provable Data Possession. In: Proc. of SECURECOMM. Istanbul, Turkey (Sept 2008)
23. Bellare, M., Boldyreva, A., ONeill, A.: Deterministic and efficiently searchable encryption. In: Proc. of CRYPTO (2007)
24. Bellare, M., Ristenpart, T., Rogaway, P., Stegers, T.: Format-Preserving Encryption. Lecture Notes in Computer Science 5867 (2009)
25. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Proc. of EUROCRYPT. Espoo, Finland (1998)
26. Boneh, D., Franklin, M.K.: An Efficient Public Key Traitor Tracing Scheme. In: Proc. of CRYPTO. Santa Barbara, California (Aug 1999)

27. Boneh, D., Gentry, C., Waters, B.: Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In: Proc. of CRYPTO. Santa Barbara, California (Aug 2005)
28. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. Cryptology 17(4) (2004)
29. Boneh, D., Sahai, A., BrentWaters: Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In: Proc. of EUROCRYPT. Saint Petersburg, Russia (May-Jun 2006)
30. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Proc. of TCC. pp. 535–554 (2007)
31. Chor, B., Fiat, A., Naor, M.: Tracing Traitors. In: Proc. of CRYPTO. Santa Barbara, California (Aug 1994)
32. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: k-Anonymity. In: Yu, T., Jajodia, S. (eds.) Secure Data Management in Decentralized Systems. Springer-Verlag (2007)
33. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proc. of ACM CCS (2006)
34. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Over-encryption: Management of access control evolution on outsourced data. In: Proc. of VLDB. Vienna,Austria (Sept 2007)
35. Dodis, Y., Vadhan, S.P., Wichs, D.: Proofs of Retrievability via Hardness Amplification. In: Proc. of TCC (2009)
36. Erway, C., Kupcu, A., Papamanthou, C., Tamassia, R.: Dynamic Provable Data Possession. In: Proc. of CCS (2009)
37. Gentry, C.: A fully homomorphic encryption scheme (2009), phD Thesis
38. Goh, E., Shacham, H., Modadugu, N., Boneh, D.: SiRiUS: Securing Remote Untrusted Storage. In: Proc. of NDSS. San Diego, California (Feb 2003)
39. Goh, E.J.: Secure indexes. Cryptology ePrint Archive (2003), http://eprint.iacr.org/2003/216
40. Goodrich, M., Sun, J., Tamassia, R.: Efficient Tree-Based Revocation in Groups of Low-State Devices. In: Proc. of CRYPTO. Santa Barbara, California (Aug 2004)
41. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-Based Encryption for Fine-grained Access Control of Encrypted Data. In: Proc. of CCS. Alexandria, Virginia (Oct-Nov 2006)
42. Juels, A., Burton, J., Kaliski, S.: Pors: Proofs of Retrievability for Large Files. In: Proc. of CCS. Alexandria, VA (Oct 2007)
43. Kallahalla, M., Riedel, E., Swaminathan, R., Wang, Q., Fu, K.: Scalable secure file sharing on untrusted storage. In: Proc. of FAST. San Francisco, California (Mar-Apr 2003)
44. Kiayias, A., Yung, M.: Traitor Tracing with Constant Transmission Rate. In: Proc. of EUROCRYPT. Amsterdam, The Netherlands (Apr-May 2002)
45. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: Proc. of ICDE (2007)
46. Naor, D., Naor, M., Lotspiech, J.: Recovation and Tracing Schemes for Stateless Receivers. In: Proc. of CRYPTO. Santa Barbara, California (Aug 2001)
47. Parkhill, D.: The challenge of the computer utility. Addison-Wesley (1966)
48. Perrig, A., Song, D., Tygar, D.: Elk, A New Protocol for Efficient Large-Group Key Distribution. In: Proc. of IEEE S&P. Orkaland, CA (May 2001)

49. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proc. of CCS. Chicago, IL (Nov 2009)
50. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6) (November/December 2001)
51. Shacham, H., Waters, B.: Compact Proofs of Retrievability. In: Proc. of Asiacrypt. Melbourne, Australia (Dec 2008)
52. Shah, M.A., Baker, M., Mogul, J.C., Swaminathan, R.: Auditing to Keep Online Storage Services Honest. In: Proc. of HotOS (2007)
53. Shah, M.A., Swaminathan, R., Baker, M.: Privacy-Preserving Audit and Extraction of Digital Contents. Cryptology ePrint Archive (2008), http://eprint.iacr.org/2008/186
54. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proc. of S&P (2000)
55. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5), 557–570 (2002)
56. Wang, C., Wang, Q., Ren, K., Lou, W.: Ensuring Data Storage Security in Cloud Computing. In: Proc. of IWQOS (Jul 2009)
57. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. In: Proc. of ESORICS (Sep 2009)
58. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. In: Proc. of ESORICS. Saint Malo, France (Sep 2009)
59. Yu, S., Ren, K., Lou, W., Li, J.: Defending Against Key Abuse Attacks in KP-ABE Enabled Broadcast Systems. In: Proc. of SECURECOMM. Athens, Greece (Sep 2009)
60. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In: Proc. of INFOCOM. San Diego, California (Mar 2010)