Chapter 19
Biometric Authentication and Data Security in Cloud Computing

Giovanni L. Masala, Pietro Ruiu, and Enrico Grosso

19.1 Introduction

The migration, from local to web applications, is probably one of the most significant advances of the recent years in the arena of the application software: sharing critical data and resources and giving support to multi-user/multi-tenancy scenarios. The development of service-oriented architectures (SOA) and WEB services are key issues in all frameworks. SOAs support designing and developing in terms of services with distributed capabilities, which can be under the control of different ownership domains. These architectures are essentially a collection of services or, in different terms, entities performing single or a limited number of repeatable activities and communicating with each other by simple data passing. Service consumers view a service provider as a communication endpoint supporting a particular request format or contract; this request format (or interface) is always separated from the service implementation.

As a matter of course, security breaches on web applications are a major concern because they can involve both enterprise and private customer data: protecting these assets is then an important part of any web application development. This process usually includes authentication and authorization steps, asset handling, activity logging, auditing. A variety of protection mechanisms has been developed,

G.L. Masala (✉)
School of Computing, Electronics and Mathematics, Plymouth University, Plymouth, UK
e-mail: giovanni.masala@plymouth.ac.uk

P. Ruiu
Istituto Superiore Mario Boella (ISMB), Torino, Italy
e-mail: ruiu@ismb.it

E. Grosso
Department POLCOMING, University of Sassari, Sassari, Italy
e-mail: grosso@uniss.it

© Springer International Publishing AG 2018

K. Daimi (ed.), *Computer and Network Security Essentials*,
DOI 10.1007/978-3-319-58424-9_19

for this purpose, like: password management, encryption, intrusion prevention, and vulnerability analysis. The extension of the web application paradigm to the cloud computing model is denoted as software as a service (SaaS). The adoption of cloud computing, in particular leveraging on the public and hybrid models [1], involves many advantages in terms of flexibility, scalability, and reliability, but also implies new challenges on security, data privacy, and protection of personal data.

Literature is vast on this topic, and different risks and vulnerabilities have been extensively studied and highlighted [2, 3]. Attacks to cloud systems are becoming more targeted and sophisticated [4], since attackers know that cloud storage is becoming one of the most adopted ways to archive and share personal information. Incidents of data leakage from the cloud are increasingly frequent and affect also big players like Apple, PlayStation, and others [5–7]. These vulnerabilities are accompanied by collateral legal and reputational risks that should be regulated by national governments. The USA and European Union have enacted regulatory requirements applicable to data stored by cloud providers [8]. The security specific risks of the cloud are primarily derived from the complexity of the architecture, which includes different models of services and distribution. Furthermore there are risks related to the characteristics of multi-tenancy and resource sharing, allowing to allocate the same resources in different times to different users [9].

A first element of risk is related to the failure of the isolation systems for storage and computational resources. When data reside on the same physical infrastructure, a failure of the isolation systems can compromise machines hosted through guest-hopping, SQL injection, and side channel attacks [10]. Individuals and organizations may have different interests and requirements, or even conflicting/competing objectives. To this concern, it is necessary to protect data and systems using methods that guarantee the physical and logical separation of resources and data flows [11]. Moreover, being the cloud a distributed architecture, this implies an increased use of networks and data communication flows, compared to traditional architectures. For example, data must be transferred to the synchronization of images, of the same virtual machine, among various and distributed hardware infrastructures. Or else, simple storage operations can involve communication between central systems and cloud remote clients. Risks are, therefore, those of incurring on sniffing, spoofing, man-in-the-middle, and side channel attacks. An additional element of risk is related to the cloud model adopted. In fact, some cloud models require the user to transfer part of the control over his own data to the service provider. In this case, not only the data are allocated on the provider's servers, but also the user cannot apply specific protection mechanisms like encryption or access control, as the service provider is the sole subject having total control of the cloud resources. Finally, some key roles for managing the cloud infrastructure, such as system administrators and managers of security systems, must be considered. These actors usually have the power to perform all types of activities, within the system, and this would potentially break safety requirements imposed by corporate policies. Yet, the assessment of this kind of fraudulent actions is very complex and there is a lack of certification agencies internationally recognized for the independent evaluation of cloud security.

The “remote user authentication” or “logical access control” is one of the fundamental steps in protecting data and IT infrastructures. Authentication protocols allow to verify that each of the participants in the electronic communication is really who he claims to be. This task is commonly demanded to a specialized architecture denoted as the authentication server (AS). The AS preserves and manages the access keys to the various subsystems. In order to access private services or data, each authorized person must first establish a connection with the AS, declare and prove his own identity, and obtain a session key useful to require further services. Currently, the most common authentication mechanisms of the ASs make use of passwords and private tokens. Passwords are subject to various security threats; for example, they can be easily stolen or intercepted and used fraudulently. Tokens are more difficult to be reproduced and for this reason they are often used in banking services. However, being more expensive and difficult to manage, they are far to be an optimal solution. Moreover, they are usually based on the possession of a physical card or device that can be easily shared with different people.

As reported in the scientific literature [12, 13], the efficient use of multiple biometric features for identity verification is still an open and attracting scientific problem; biometric physical access systems are perceived as reliable [12], then minimizing the typical risks of traditional authentication systems in applications that require a high level of security like border control. On the other hand, the use of biometric data for the logical access to IT services is a more challenging and still unsolved problem. Certainly, the use of biometric techniques can be considered as one way to ensure a significant increase of security in the authentication protocols managed by modern authentication servers.

One of the criticisms of some biometric approach is related to privacy risks. In particular, this has to do with the storage of images or other biometric features in the database of the authentication server, in order to be compared during the recognition phase. These images are considered as sensitive data and should be protected with high secure systems [14]. Hence, according to privacy regulations, it is not possible to outsource these data to cloud services. Authors use often techniques to overcome this problem, as fuzzy biometric templates, based on the fuzzy vault of Jules and Sudan [15], for instance, the Biometric Encryption scheme by Soutar et al. [16], Cancelable Biometrics by Ratha et al. [17], robust bit extraction schemes based on quantization, e.g., of Linnartz and Tuyls [18], of Chang et al. [19], and of Chen et al. [20], and applications of the fuzzy commitment scheme of Juels and Wattenberg [21] to biometric templates, e.g., the constructions of Martini and Beinlich [22] for fingerprints. Authors in [23] propose a solution, using a compact representation of the biometric feature, converted using Scale Invariant Feature Transform (SIFT) representation: only this model is used to recognize the user and stored in the cloud; thus, it is not required to protect sensible data.

In this chapter, we present an example of cloud system [23, 24] that uses biometric authentication based on fingerprints [25]. This advanced access control is combined with a very peculiar fragmentation technique guaranteeing the security of the data residing on the cloud architecture. In Sect. 19.2 some preliminary considerations concerning the cloud platform are introduced while in Sect. 19.3 an example of cloud system is described in detail and the main results on the cloud security are discussed. Section 19.4 draws some conclusions, pointing out issues and problems that will be faced in the near future.

19.2 Preliminaries

19.2.1 Cloud Platform

OpenStack [26] is an open source project that many identify as the first true cloud operating system. OpenStack has to be considered as a basic technology rather than a solution; by analogy is often associated with the Linux kernel.

The example of project [23, 24] described in this chapter has the primary goal of supporting basic web applications shared by small and medium companies; candidate platforms for cloud computing should be, therefore, oriented to scalability, to be implemented according to the public or private cloud models. In this respect, OpenStack has many interesting features; it allows a prompt and elastic control of computing resources such as CPUs, storage, and networks, and includes features for general system management, process automation, and security.

OpenStack consists of several individual sub-components. This modular design improves flexibility because each component may be used alone or in combination with others. Some of these modules marked as cores (such as compute, storage, and networking) represent the essential parts of the platform. Other modules are initially placed in an incubator from which they come only if needed.

The main modules of OpenStack, fully distributable and replicable, are the following: computing (Nova), networking (Neutron), image templates (Glance), block (Cinder) and object storage (Swift), graphical interface platform accessible via the web (Horizon), authentication, the native orchestration module (Heat), and accounting (Keystone). The architecture is based on the concept of "sharing nothing" that makes components independent and self-sufficient, avoiding the sharing of memory or storage. Communications between the different modules are asynchronous and are managed by queue managers (message brokers) that implement the Advanced Message Queuing Protocol (AMQP). The various services communicate with each other through specific Application Programming Interfaces (APIs) that implement the REST model. All these features make OpenStack an ideal tool to be deployed on commodity hardware, with consequent economic benefits and flexibility.

Virtualization is an important element of cloud computing because it guarantees the required elasticity in resource allocation. Virtualization is a technique that allows to run multiple virtual machines on a single physical server and to optimize the available resources. It is possible to provide different levels of abstraction that make the operating system do not see the physical hardware but the virtual hardware.

This abstraction is achieved by a software layer, called *hypervisor,* which is usually integrated into the operating system kernel and it is loaded at system startup. The *hypervisor* does not offer any management capabilities to virtual machines. Like many of the cloud computing platforms also OpenStack is not released with a specific *hypervisor*; the system administrator can choose among a set of supported hypervisors like VMware, Hyper-V, Xen, and KVM. In this project the Kernel-based Virtual Machine (KVM) is used; it is one of the most supported and popular among scientific developers. KVM is a Linux kernel module that allows a user program to use hardware virtualization capabilities of various processors. It supports in particular processors from AMD® and Intel® (x86 and x86_64) having these features (Intel VT or AMD-V). From the point of view of the operating system each virtual machine is seen as a regular Linux process that can use the hardware resources according to what established by the scheduler. A normal Linux process has two execution modes: kernel and user. KVM adds a third mode, a guest mode that has its own kernel and user modes. The main benefit of KVM is that being integrated into the kernel improves performance and reduces the impact on existing Linux systems.

19.2.2 Data Security

A possible solution, to guarantee the security of data residing on distributed cloud infrastructure, is the use of systems for the fragmentation and distribution of data, which allow to split the data into fragments and disperse them on all machines available to the cloud. In this way the recovery and the use of the data is very complex for an unauthorized user. By using fragmentation techniques, it is possible to distribute data on platforms of different providers, and to problems arising from the lack of trust in the service provider. However, in order to achieve a proper fragmentation and distribution of the data in the network, it is necessary to develop support tools to ensure the prompt availability and integrity of these data, without increasing the complexity of the system. In fact, an excessive consumption of resources or performance degradation related to procedures of information retrieval would compromise this approach.

The use of fragmentation techniques to protect outsourced data is not a novel approach in literature. Different solutions have been proposed; however, the most prominent ones use cryptography to obfuscate data [27, 28] and traditional relational databases [29, 30], exploiting sharding functionalities. The approach proposed in this paper is completely different and original, since disclaims these two elements. The solution proposed avoids cryptography, seen as an excessive overhead to data retrieval processes, since encryption makes it not always possible to efficiently execute queries and evaluate conditions over the data. Moreover, another innovative aspect regarding the use of modern database platforms, which embracing the NoSQL paradigm, is characterized by highly scalable distributed architectures.

These platforms include also native management features (redundancy, fault tolerance, high availability) which permit to design simple fragmentation systems without the burden of having to implement these complex control systems.

19.3 An Example of Cloud Platform

19.3.1 General Implementation of the Cloud System

The meaning of the term "node" usually relates to individual machines running the functions of the cloud. In some cases a node corresponds to a physical machine, in other cases it corresponds to an instance of a virtual machine (VM). OpenStack has a distributed nature; therefore, during installation it is necessary to take account of the number of nodes required for the installation of the platform. From the official documentation of OpenStack, the minimum number of nodes to be used in a stable installation is five, at least one for each of the following functions: Horizon, Keystone, Neutron, Nova, and Swift. In particular:

- Neutron is the system that allows to manage the network connectivity and to address the VMs in the cloud. It includes some features of type "networking as a service" that support the use of advanced networking.
- Swift is a distributed storage system that can accommodate data of users of the platform or VMs. It allows to manage the policies of replication and consistency ensuring the integrity, safety, and protection of distributed data in the cloud.
- Keystone manages all security policies, privileges, and authorization for user access to the platform. It provides API client authentication, service discovery, and distributed multi-tenant authorization.
- Horizon is a graphical interface platform accessible via the web, for easy and intuitive management of the cloud.
- Nova is designed to provide power massively scalable, on demand, self-service access to compute resources. It is developed to manage and automate computer resources and can work with several virtualization technologies.
- Glance is the Virtual Machine Image Repository or a catalog of images of the operating system that users can use to instantiate VMs.
- Cinder allows to provide storage that can be used by Nova to serve the VMs. Storage is provided in the form of block storage device and may be required as a service without reference to the real physical allocation.
- Heat is the native orchestration module of processes of the cloud.

In the considered system [23, 24] one module of OpenStack is not installed: Ceilometer, which allows monitoring and billing use of cloud resources. Figure 19.1 (top) highlights the distribution of modules in the nodes; the network configuration of the platform is illustrated in Fig. 19.1 (bottom).

The architecture is divided into two different Italian data centers located in Alghero and Turin. Each server stands on a virtual private LAN: we have a server

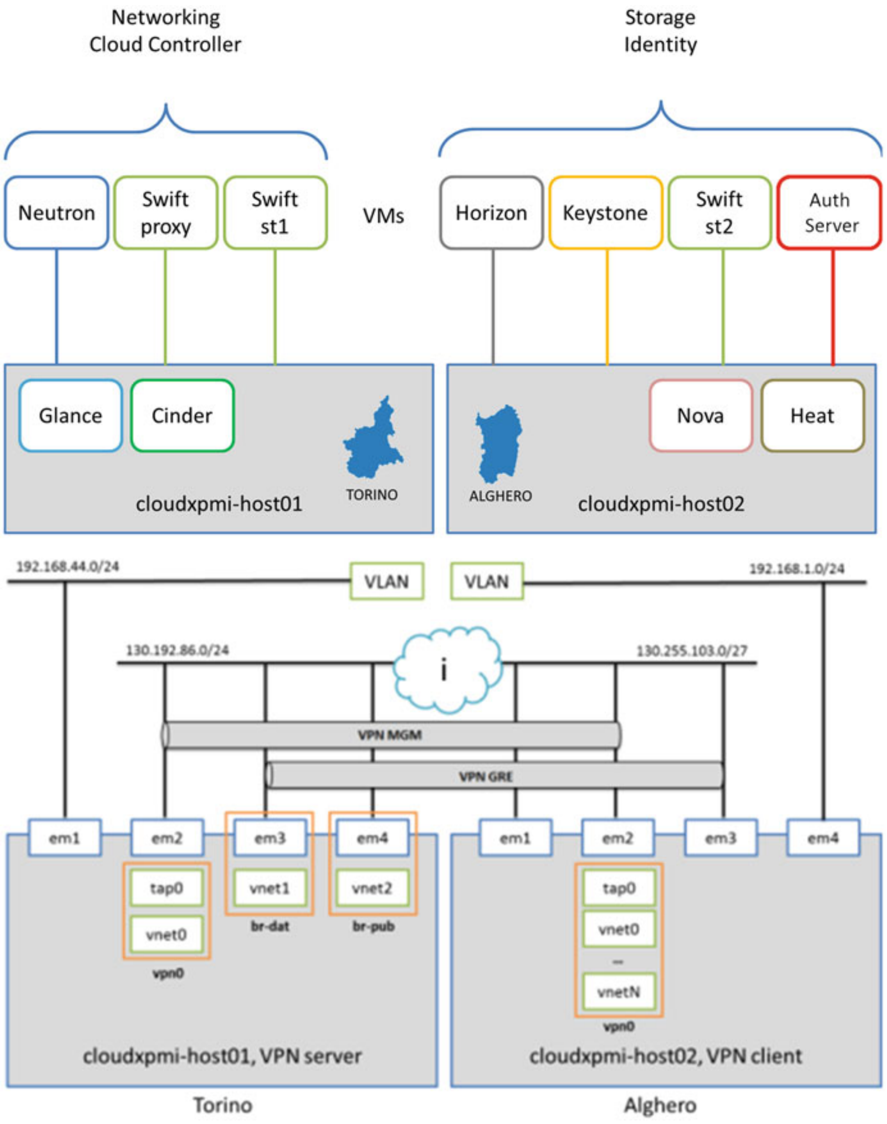

Fig. 19.1 (*Top*) The subdivision of OpenStack functions between our two Italian data centers of Alghero and Turin: services Nova and Heat have a physical machine on the server of Turin and all other services are arranged on virtual nodes. (*Bottom*) The general network configuration of the cloud platform

in Turin, which uses the *em1* interface, while another server, in Alghero, uses the interface *em4*. The other network adapters are used to configure the three networks necessary for the operation of OpenStack. The public network is used to allow the connection of the virtual machines to the outside (Internet). For this network it is necessary to configure a virtual interface for the Neutron node with a public IP address. This interface will then be used to configure the bridge virtual audience (*br-pub*) managed by Neutron. The *management network* interconnects physical hosts and virtual machines, which are the functional nodes of the cloud platforms. These nodes are equipped with the software modules of OpenStack, as described in the bottom part of Fig. 19.1. A Virtual Private Network (VPN) has been set up to ensure secure communication between these nodes (which manage all data transiting in the cloud). The Turin node has been configured as the VPN server, using the bridge *tap0*, attached to the interface *em2*. The host of Alghero and the nodes hosted in the same server connect to the VPN server through another bridge *tap0*, always on the respective interface *em2* (see Fig. 19.1).

The *data network* instead is the channel reserved for communication between virtual machines. OpenStack manages this kind of communication through the creation of ad hoc overlay network, which uses *Generic Routing Encapsulation (GRE) tunnels to encapsulate traffic*. A tunnel is established between the two hosts and the other two tunnels between the same host and the Neutron node.

Keystone provides authentication and accounting for the entire platform, and it is installed on a dedicated virtual machine, on the physical server of Alghero. This is necessary to facilitate its interface with a dedicated biometric authentication, via private network connection; the service is hosted in the authentication server (AS) of the data center, but externally with respect to the platform OpenStack.

19.3.2 Integration of Biometric Recognition with the Cloud Platform

The recognition system is implemented in an isolated authentication server (AS), which exposes the necessary API for ensuring interoperability with the rest of the system. The API includes a minimal set of functions providing registration (enrollment) of a new user in the system, identification of a user, cancellation of a registered user.

The authentication system is designed to be scalable in horizontal on multiple computing nodes and vertical optimizing the CPU performance, through the parallel computation inside the node, in which it operates. To improve processing time, at start-up of the computing node, the whole set of information related to the users is copied directly into RAM to cancel the disk access times. With the current service configuration (1 node with 4 vCPU) the total time of identification is calculable, on average, in 3/10 of a second per registered user.

A VPN is placed between the system and the desktop user application. When the VPN encrypted tunnels are enabled, the user starts the session simply touching the fingerprint scanner. This VPN selectively enables the services that can be accessed by the user: at the start of the process the user only sees the API server while, if authenticated, the system creates a route to the GUI. In this way, communications between the client and the API are always protected and the session ID is never transmitted in clear.

19.3.2.1 Biometric Recognition

The desktop application includes software modules both for the enrollment and the authentication of users. During enrollment, the new user's fingerprint is converted into a compact representation, called model; only this model will be used to recognize the user, thus it is not required to store the fingerprints in the AS database; only the models are recorded.

The features characterizing the model are obtained by using the Scale Invariant Feature Transform (SIFT) representation [31, 32]. Recently SIFT has emerged as a cutting-edge methodology in general object recognition as well as for other machine vision applications [31–35]. One of the interesting features of the SIFT approach is the capability to capture the main local patterns working on a scale-space decomposition of the image. In this respect, the SIFT approach is similar to the Local Binary Patterns method [36, 37], with the difference of producing a more robust view-invariant representation of the extracted 2D patterns.

The matching for the authentication application is performed considering the SIFT features located along a regular grid and matching overlapping patches; in particular, the approach subdivides the images in different sub-images, using a regular grid with a light overlap. The matching between two images is then performed by computing distances between all pairs of corresponding sub-images, and therefore averaging them [34]. A fusion module takes the final decision.

The fingerprint scanner used for the purpose of the project has a 1×1 inch sensor and is certified by FBI according to Personal Identity Verification (PIV) Image Quality Specifications. These technologies ensure a good quality and performance level, currently unreachable with most commercial devices.

19.3.2.2 Performance of the Authentication System

Our authentication system, based on SIFT [23, 24, 34], is tested on a subset of the Biosecure database [38]. More in detail, we used a subset including two different acquisitions (A and B) of the same fingerprint for 50 persons, randomly extracted from the original database of 400 subjects. The dataset contains features extracted in a realistic acquisition scenario, balanced gender, and population distributions. We made the comparison between each fingerprint A, against the fingerprints B of all 50 persons, for a total number of 2500 comparisons. We used normalized scores to express the similarity between two biometric patterns. The higher the score is, the higher the similarity between the biometric patterns.

The access to the system is granted only, if the score for a trained person (identification), or the person that the pattern is verified against (verification), is higher than a certain given threshold. Depending on the choice of the classification threshold, between all and none of the impostor, patterns will be erroneously accepted by the system. The threshold, dependent fraction of the falsely accepted patterns, divided by the number of all impostor patterns, is called False Acceptance Rate (FAR). Again depending on the value of the threshold, between none and all, also a varying number of genuine patterns will be falsely rejected. The fraction of

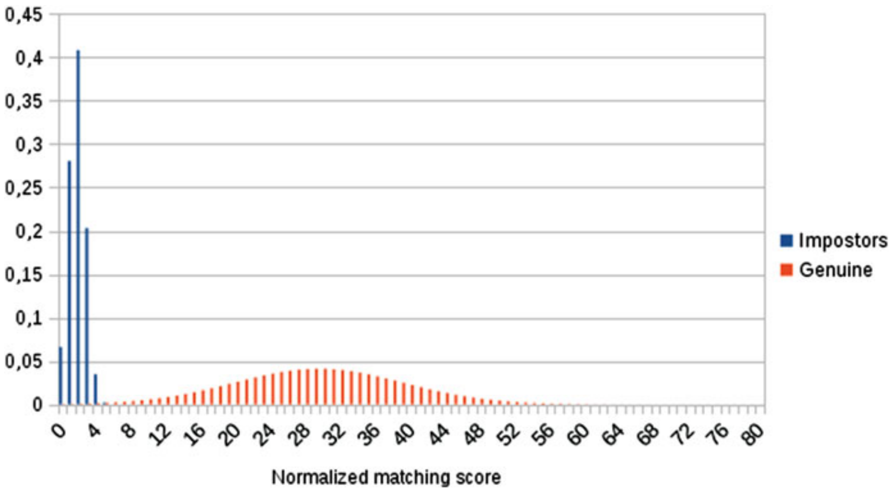

Fig. 19.2 Estimation of the normal distributions for genuine and impostors, in the dataset

Table 19.1 Normal distributions for genuine and impostors

	Mean	St. Dev.
Genuine	29.3	±9.7
Impostors	1.9	±1.0

Table 19.2 Estimations of FAR and FRR, varying the matching threshold

Matching threshold	FAR	FRR
6	4.060E-05	0.00012
7	2.760E-07	0.00016
8	6.430E-10	0.00023
9	5.140E-13	0.00032
10	1.410E-16	0.00044
11	1.320E-20	0.00059

the number of rejected genuine patterns, divided by the total number of genuine patterns, is called False Recognition Rate (FRR). The distributions of the genuine and the impostors scores sometimes overlap, and it is not easy for the choice of the threshold value.

To this purpose, the distributions for genuine users and impostors are estimated in Fig. 19.2. The threshold is tuned in such a way to give suitable FAR and FRR rates. In Table 19.1 are shown the estimated mean and standard deviation for our distributions, while in Table 19.2 are expressed the estimations of FAR and FRR, on such distributions are given, varying the matching threshold.

It is possible to note, in Table 19.2, that with a high threshold (e.g., 10) the FAR is virtually zero (no impostors enter into the system), without causing actual drop in FRR performance. In fact a FRR $= 0.00044$ corresponds to the above threshold value, which means that only in 44 cases over 100,000 the system rejects genuine fingerprints.

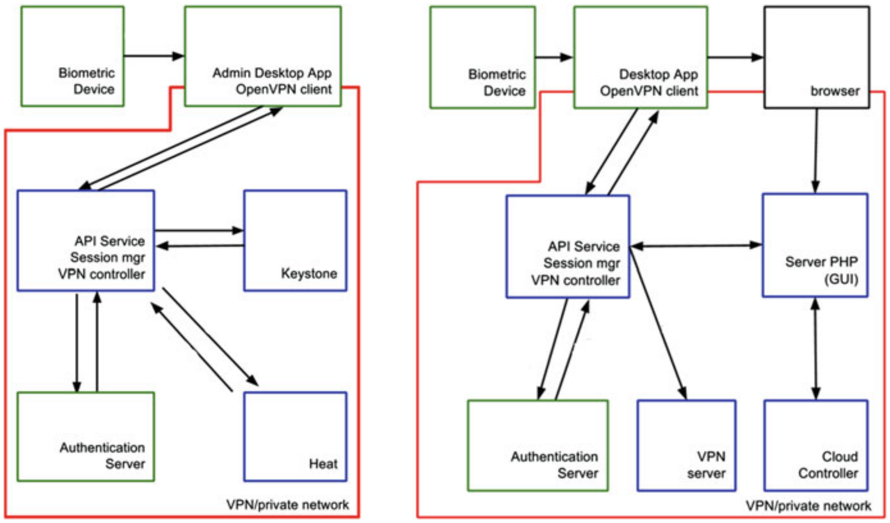

Fig. 19.3 The components and workflow of the registration procedure are shown in the *left* diagram, while figure on the *right* is related to the authentication procedure

19.3.2.3 Automation of the Biometric Access

Registration Process

Before registration process starts, a secure channel is created through a VPN. Next, the client sends new user's data to the system. The API service receives two files in JSON format, containing the meta-data that are generated during fingerprint acquisition. The JSON object also contains the user's company name. After receiving user data properly, the system initiates an automated procedure to set up the virtualized environment that will host the user's services. A general overview of the process is represented in Fig. 19.3.

During registration (Fig. 19.3, left), the service API does the following:

1. Add user to API service list;
2. Add user to AS;
3. Add user to OpenStack Keystone;
4. Add user to OpenVPN Server;
5. Create a new Stack with OpenStack HEAT.

At this stage, automated checks on each component are carried out. Given the complexity and heterogeneity of the resources involved, the system will conduct checks to prevent misalignment between authentication needed configuration service, Keystone, OpenVPN server, and API service. Therefore, the registration process ends when at least one of the operations listed above fails. Initially, the system calculates the new user's ID, password, and a network CIDR. The password is generated by random algorithm, and it is used by the API service to communicate with Keystone and manage cloud services. Therefore, in order to make the whole system even safer, no other component will possess credentials. Continuing, the API service sends the username and password to the AS that registers the new user. If the registration is not successful, the AS returns an error message, and the whole process stops. The API service requires a token in Keystone to create a new user. The interaction between Keystone and API service is done through the OpenStack API endpoint, called Identity.

During the registration process, a VPN certificate is automatically generated and provided to the user. This should be used by the user every time a connection with the cloud services is established. In this phase, the automation is in charge of the OpenVPN server which accepts as input (communication done using a Rest API Interface) the user name, and the network's CIDR, and returns the OVPN certificate, ready to be used on the VPN client. The OpenVPN server setup correct routing rules that allow user access to the network and thus to its services. The rules will take effect only if user is authenticated successfully. OpenStack has images of virtual machines, pre-configured and ready to use. The API service sends a request to create a new stack. A stack consists of a set of virtual machines connected to a new network. The network is then connected to a virtual router. All these operations are carried out automatically by the machine-accessible orchestrator heat. This is the hardest operation of the entire process because it involved almost all OpenStack services: Nova for the creation of virtual machines, Neutron security groups, ports, subnets, and vRouter interfaces. Finally, API service has successfully completed all operations and returns to the desktop client, the Open VPN certificate.

Authentication Process

The procedure for authenticating the user is shown in Fig. 19.3 (right). Before performing any operation, the user connects to the system with its OpenVPN certificate. When the client desktop finishes to acquire and convert the fingerprint, sends the file to the API service through a VPN tunnel. The data is transmitted to the AS and if user is recognized, AS returns a pair of values (that are username and password). The credentials will also be used in this case (as happens in the registration process) by API service, GUI, and OpenStack. According to result of authentication stage, the API service creates or not a new session. When the user is correctly recognized the API service generates new PHP session by creating a session file in PHP session path containing usernames, passwords, token (Open Stack), and stack ID. The username and password parameters are supplied from the AS, while stack ID is obtained by consulting a list on the API service and a token is generated by an automatic procedure. The API service connects to Open Stack Keystone requesting the token that will be used to manage the virtual machines (start, stop, resume, etc.). Finally, the API service has completed its task and returns the generated session ID to desktop client.

At this stage, the user can access to services simply connecting to the URL via browser. When the user makes a request for service management, the GUI server interacts with Nova and other Open Stack services, through the REST API. All the automation layer is run with PHP with a light framework which is able to manage processes quickly. When the user leaves the GUI, the session is destroyed and all environmental variables used for service management are removed.

It is worth to highlight some aspects of the implemented security procedure:

- User and password to access the cloud are never transmitted out of the cloud itself.
- Web GUI, AS, and private cloud controller are not accessible outside the cloud.
- Sensitive data residing on the cloud (fingerprint model file) are compared inside the cloud.
- The data transfer is not related to the user (nobody outside the cloud can associate the model file with some user information).

19.3.3 Data security

Cloud computing services and applications must face various challenges, including latency, unreliability, malicious behavior; most of these challenges are related to the public shared environment in which cloud services are hosted. In particular, security of outsourced data is still one of the main obstacles to cloud computing adoption in public bodies and enterprises. The main reason is impossibility to trust the cloud provider due to the lack of control that the user has over the infrastructure, an issue intrinsic of the public cloud model. Algorithms have to be developed to cope with these challenges and innovative architectures.

In this work is proposed a solution to ensure the data security and high availability of the resources, using an innovative distributed cloud storage architecture. The solution is based on data chunking technique: The basic idea is to share data in small chunks and spread them on different VMs hosted on cloud computing. The complete control of the distributed storage system is delegated to the user who hosts the master node of the system, as shown in Fig. 19.4. The master node maintains the namespace tree and the mapping of blocks to the slaves nodes. Thus, only the user knows the location of the chunks needed to recompose the data. Even if a malicious user can access to one of the nodes which possess the chunks he cannot use it as the information is incomplete.

This solution is a viable countermeasure also for malicious behavior of the cloud provider. Some of the features of the proposed solution are:

- Distributed storage system implemented in cloud, with client–server architecture and partially trusted environment;
- Security granted by chunking data and spreading it on different nodes (VM) possibly hosted by different cloud providers;

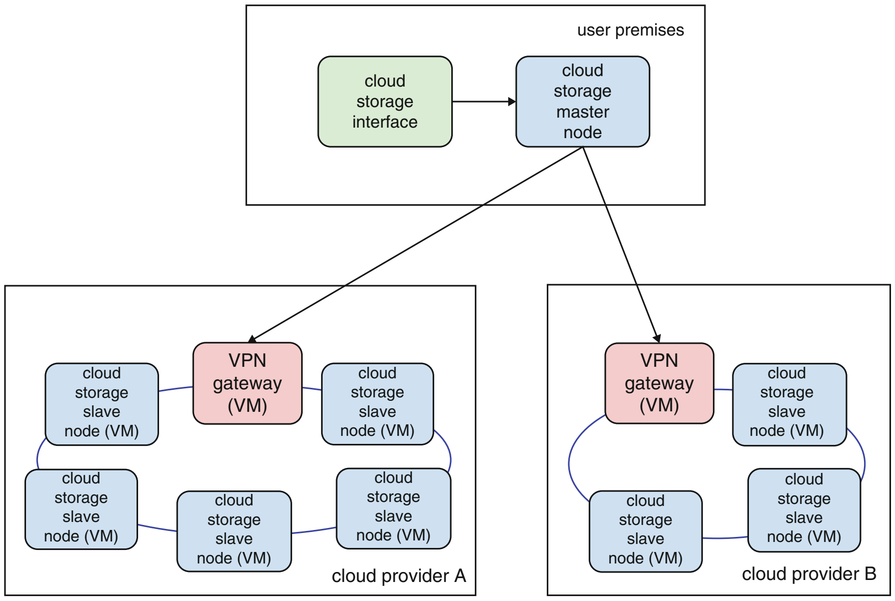
user premises
cloud storage interface
cloud storage master node
VPN gateway (VM)
cloud storage slave node (VM)
cloud storage slave node (VM)
cloud storage slave node (VM)
cloud storage slave node (VM)
cloud storage slave node (VM)
cloud provider A
VPN gateway (VM)
cloud storage slave node (VM)
cloud storage slave node (VM)
cloud storage slave node (VM)
cloud provider B

Fig. 19.4 Architecture of the distributed storage system

- Availability and resiliency ensured by the redundancy of nodes and replica of chunks;
- The possibility to use different cloud providers prevents also the so-called vendor "lock-in."

19.3.3.1 Distributed Storage Systems

There are two main categories of distributed storage systems architectures: peer-to-peer and client–server [39]. The latter architecture has been chosen for the implementation because best fit the objectives of the proposed solution. A client–server-based architecture revolves around the server providing a service to requesting clients. The server is the central point, responsible for authentication, sharing, consistency, replication, backup, and servicing requesting clients. In our implementation the master node embraces the server's role and slave nodes the client's role. As slaves nodes are hosted on the cloud, the system operates in a partially trusted environment; users are exposed to a combination of trusted and untrusted nodes [39].

In Distributed Storage Systems data can be replicated across multiple geographical sites to improve redundancy, scalability, and data availability, as shown in Fig. 19.4.

Although these solutions provide the scalability and redundancy that many cloud applications require, they sometimes do not meet the concurrency and performance needs because of the latency due to the network [40]. Some examples of the most known distributed storage systems are HDFS, Ceph, MooseFS, mongoDB.

19.3.3.2 Architecture of the System

The architecture of the solution is comprised of interconnected nodes where files and directories reside. There are two types of nodes: the master node that manages the filesystem namespace and regulates client access to files, and the slave node that stores data as blocks within files. All nodes communicate with each other using TCP-based protocols. The mechanism of data protection does not rely on RAID approaches, but the file content is replicated on multiple slaves for reliability. Master and slave nodes can run in a decoupled manner across heterogeneous operating systems, and on different cloud providers. The complete control of the system is delegated to the master node, which maintains the namespace tree and the mapping of blocks to slave nodes. Slave nodes have little intelligence and not know the location of other slaves or chunks of data.

User applications access the system using a specific client, a library that exports the filesystem interface. When a user wants to perform a reading action on filesystem, the client first asks the master node for the list of *namenodes* that host the chunks of the file. After that, the client contacts a slave node directly and requests the transfer of the desired block. Instead, when a user wants to write on the filesystem, it first asks the master to choose slaves to host chunks of the file. All decisions concerning replication of the chunks are taken by the master node. This ensures the reliability of the data and the fault tolerance of the system.

19.4 Conclusion

A complete system for web applications, and data management over the Cloud, is presented and it is coupled with strong biometric authentication. The system guarantees the identity of the users and makes easy, and secure, the access to data and services. Moreover, the adoption of a data chunking solution is proposed, which is based on a distributed cloud storage architecture. This provides protection of data residing also from provider's administrators and hardware supervisors. A further improvement of the system will extend biometric access to multimodal techniques, thus including face and face + fingerprint authentication. The development of a web server application for the user side, aimed to avoid the installation of local software, will be also pursued.

References

1. Srinavasin, M. K., et al. (2012). State of the art cloud computing security taxonomies: A classification of security challenges in the present cloud computing environment. In *ICACCI 2012 proceedings of the international conference on advances in computing, communications and informatics* (pp. 470–476). ACM.
2. Zissis, D., & Lekkas, D. (2012). Addressing cloud computing security issues. *Future Generation Computer Systems, 28*(3), 583–592.
3. Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications, 34*(1), 1–11.
4. Nelson, C., & Teller, T. (2016). Cloud attacks illustrated: Insights from the cloud provider. In *RSA conference, February 29, 2016–March 4, 2016*. Moscone Center San Francisco.
5. Skokowski, P. (2014). Lessons from Apple iCloud Data Leak. *CSA–Cloud Security Alliance Industry Blog* [Online]. https://blog.cloudsecurityalliance.org/2014/11/19/lessons-from-apple-icloud-data-leak/
6. Gonsalves, A. (2013). Data leakage risk rises with cloud storage services. *Computer world Hong Kong* [Online]. http://cw.com.hk/news/data-leakage-risk-rises-cloud-storage-services
7. Konstantas, J. (2011). What does the Sony PlayStation network breach teach us about cloud security? *Security week* [Online]. http://www.securityweek.com/what-does-sony-playstation-network-breach-teach-us-about-cloud-security
8. Sotto, L. J., Treacy, B. C., & McLellan, M. L. (2010). Privacy and data security risks in cloud computing. *World Communications Regulation Report, 5*(2), 38.
9. European Commission (2012). Exploiting the potential of cloud computing in Europe, September 27, 2012 [Online]. Available: http://europa.eu/rapid/press-release_MEMO-12-713_it.htm
10. Yinqian Zhang, M. K. (2012). Cross-VM side channels and their use to extract private keys. In *CCS'12*. Raleigh, North Carolina, USA.
11. NIST (2013). NIST Cloud Computing Standards Roadmap. NIST
12. Ross, A. A., Nandakumar, K., & Jain, A. K. (2006). *Handbook of multibiometrics* (Vol. 6). Berlin: Springer.
13. Vielhauer, C. (2005). *Biometric user authentication for IT security: From fundamentals to handwriting (advances in information security)* (Vol. 18). New York: Springer.
14. Ratha, N. K., Connell, J. H., & Bolle, R. M. (2001). Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal, 40*(3), 614–634. Chicago.
15. Juels, A., & Sudan M. (2002). A fuzzy vault scheme. In *Proceedings of the 2002 IEEE international symposium on information theory* (p. 408). IEEE.
16. Soutar, C., Roberge, D., Stoianov, A., Gilroy, R., & Kumar, B. V. (1998). Biometric encryption using image processing. In van Renesse, R. L. (Ed.), *Proceedings of the SPIE, optical security and counterfeit deterrence techniques II* (Vol. 3314, p. 178U188).
17. Ratha, N. K., Connell, J. H., & Bolle, R. M. (2001). Enhancing security and privacy of biometric-based authentication systems. *IBM Systems Journal, 40*, 614–634.
18. Linnartz, J.-P., & Tuyls, P. (2003). New shielding functions to enhance privacy and prevent misuse of biometric templates. In *Proceedings of the 4th international conference on Audio- and video-based biometric person authentication (AVBPA'03)* (pp. 393–402). Springer.
19. Chang, Y., Zhang, W., & Chen, T. (2004). Biometrics-based cryptographic key generation. In *Proceedings of the IEEE international conference on multimedia and expo (ICME '04)* (pp. 2203–2206). IEEE Computer Society.
20. Chen, C., Veldhuis, R., Kevenaar, T., & Akkermans, A. (2007). Multibits biometric string generation based on the likelyhood ratio. In *Proceedings of the IEEE conference on biometrics: Theory, applications and systems (BTAS '07)* (pp. 1–6). IEEE Computer Society.
21. Juels, A., & Wattenberg, M. (1999). A fuzzy commitment scheme. In *Proceedings of the 6th ACM conference on computer and communication security* (pp. 28–36). ACM.

22. Martini, U., & Beinlich, S. (2003). Virtual PIN: Biometric encryption using coding theory. In Brömme, A., & Busch, C. (Eds.), *BIOSIG 2003: Biometrics and electronic signatures, ser. Lecture notes in informatics* (Vol. 31, pp. 91–99). Gesellschaft fur Informatik.
23. Masala, G. L, Ruiu P, Brunetti A, Terzo O, & Grosso E (2015). Biometric authentication and data security in cloud computing. In *Proceeding of the international conference on security and management (SAM). The Steering Committee of The World Congress in Computer Science* (p. 9). Computer Engineering and Applied Computing (WorldComp).
24. Ruiu, P., Caragnano, G., Masala, G. L., & Grosso, E. (2016). *Accessing cloud services through biometrics authentication on proceedings of the international conference on complex, intelligent, and software intensive systems (CISIS-2016), July 6–8, 2016*. Japan: Fukuoka Institute of Technology (FIT).
25. Maltoni, D., Maio, D., Jain, A., & Prabhakar, S. (2009). *Handbook of fingerprint recognition* (2nd ed.). Berlin: Springer.
26. OpenStack. OpenStack cloud administrator guide [Online]. Available http://docs.openstack.org/admin-guide-cloud/content/
27. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., & Xu, Y.. Two can keep a secret: A distributed architecture for secure database services. In: *Proceeding of the 2nd conference on innovative data systems research (CIDR)*. Asilomar, California, USA.
28. Ciriani, V., Di Vimercati, S. D. C., Foresti, S., Jajodia, S., Paraboschi, S., & Samarati, P. (2007). Fragmentation and encryption to enforce privacy in data storage. In *European symposium on research in computer security* (pp. 171–186). Berlin, Heidelberg: Springer.
29. Damiani, E., De Capitani, S., di Vimercati, S., Jajodia, S., Paraboschi, S., & Samarati, P. (2003). Balancing confidentiality and efficiency in untrusted relational DBMSs. In: *CCS03 proceeding of the 10th ACM conference on computer and communications security, Washington, DC, USA, October 2003*. New York: ACM Press.
30. Hacigümüs, H., Iyer, B., & Mehrotra, S. (2002). Providing database as a service. In *ICDE'02 proceedings of the 18th international conference on data engineering, San Jose, California, USA*. Los Alamitos, California: IEEE Computer Society.
31. Lowe, D. (1999). Object recognition from local scale-invariant features. In *International conference on computer vision and pattern recognition* (pp. 1150–1157).
32. Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*(2), 91–110.
33. Lowe, D. (2001). Local feature view clustering for 3d object recognition. In *IEEE conference on computer vision and pattern recognition* (pp. 682–688).
34. Bicego, M., Lagorio, A., Grosso, E., & Tistarelli, M. (2006). On the use of SIFT features for face authentication. *In CVPRW'06 Conference on computer vision and pattern recognition workshop* (pp. 35–35). IEEE.
35. Ke, Y., & Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. In *IEEE conference on computer vision and pattern recognition*.
36. Heusch, G., Rodriguez, Y., & Marcel, S. (2005). Local binary patterns as an image preprocessing for face authentication. *IDIAP-RR 76, IDIAP*.
37. Zhang, G., Huang, X., Li, S., Wang, Y., & Wu, X. (2004). Boosting local binary pattern (lbp)-based face recognition. In *L. 3338, SINOBIOMETRICS* (pp. 179–186). Springer.
38. Fierrez, J., Galbally, J., Ortega-Garcia, J., et al. (2010). BiosecurID: A multimodal biometric database. *Pattern Analysis and Applications, 13*, 235.
39. Placek, M., & Buyya, R. (2006). The University of Melbourne, a taxonomy of distributed storage systems. Reporte Técnico, Universidad de Melbourne, Laboratorio de Sistemas Distribuidos y Cómputo Grid.
40. Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., & Buyya, R. (2015). Big Data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing, 79*, 3–15.