Biometric Authentication and Data Security in Cloud Computing

G.L.Masala[1], P. Ruiu[2], A. Brunetti[1], O.Terzo[2], E. Grosso[1],
[1]Department of Political Science, Communication, Engineering and Information Technologies,
Computer Vision Laboratory, University of Sassari, Sassari, ITALY.
[2] Istituto Superiore Mario Boella (ISMB), Turin, ITALY.

Abstract - *The paper presents a new Cloud platform designed to support basic web applications shared by small and medium companies. The platform guarantees secure access of multiple users and complete logical separation of computational and data resources related to different companies. A peculiar data fragmentation approach ensures a high-level of protection of the data stored in the Cloud.*

The platform is built using the OpenStack architecture, while the user authentication is based on an original biometric approach that easily integrates finger and face modalities. Details of the authentication process and of the service modules involved in the biometric authentication are given. The platform proved to be effective and it is currently under beta testing phase for a limited set of candidate companies.

Keywords: *Security of data residing ,data fragmentation, multimodal biometric authentication, Openstack.*

1 Introduction

The migration from local to web applications, sharing critical data and resources and giving support to multi-user/multi-tenancy scenarios, is probably one of the most significant advances of the recent years in the arena of the application software.

The development of service-oriented architectures (SOA) and WEB services are key issues in this framework. 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, repeatable activities that perform single or a few specialized operations and communicate 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. Traditional protection mechanisms, like password management, encryption, intrusion prevention and vulnerability analysis have been developed for this purpose.

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.

The security specific risks of the cloud are primarily derived from the complexity of the architecture (which includes different models of services and distribution) and its characteristics of multi-tenancy and resource sharing, allowing to allocate the same resources in different times to different users [2].

A first element of risk is related to the failure of the isolation systems for storage and computational resources. When data of individuals and organizations, who may have different interests and requirements or even conflicting/competing objectives, 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 [4]. 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 [3].

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 for 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 .

This paper deals with “remote user authentication” or “logical access control”, 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, every 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 [5-6], 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 [5], 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.

In this paper, we present a Cloud system that uses biometric authentication based on fingerprints [13]. This advanced access control is combined with a very peculiar fragmentation technique guaranteeing the security of the data residing on the cloud architecture. In chapter II we introduce some preliminary considerations concerning the cloud platform while in chapter III the realized Cloud system is described in detail and the main results on the cloud security are discussed. Section IV draws some conclusions, pointing out issues and problems that will be faced in the near future

2 Preliminaries

2.1 Cloud platform

OpenStack [7] 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 project described in this paper 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 many features for general system management, process automation and security.

OpenStack consists of several individual sub-components. This modular design facilitates great flexibility because each component may be used alone or in combination with others. Some of these modules, marked as cores (such as compute, storage, 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), authentication, and accounting (Keystone). The architecture is based on the concept of "sharing nothing" that make 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*, but it is the system administrator who chooses one of the supported: VMware, Hyper-V, Xen and KVM. In our project we use the Kernel-based Virtual Machine (KVM); 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: guest mode that allows to isolate processes that grain inside. The main benefit of KVM is that being integrated into the kernel improves performance and reduce the impact on existing Linux systems.

2.2 Security of data residing

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.

3 Main results

3.1 General implementation of the Cloud System

OpenStack has a distributed nature, therefore, during installation it was necessary to take account of the number of nodes required for the installation of the platform. 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). From the official documentation of OpenStack, the minimum number of nodes to be used in a stable installation is 5, at least one for each of the following functions: Horizon, Keystone, Neutron, Compute, 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 advanced 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 and authorization for user access to the platform, with different privileges. Moreover, it is the system that functions use to request services through a specific *Application Programming Interface (API).*
- Horizon is a graphical interface platform accessible via the web, for easy and intuitive management of the cloud.
- 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.

In our system two modules of OpenStack were not installed: Ceilometer, which allows monitoring and billing use of cloud resources, and Heat, which manages the orchestration of processes of the cloud.
Figure 1 highlights the distribution of modules in the nodes

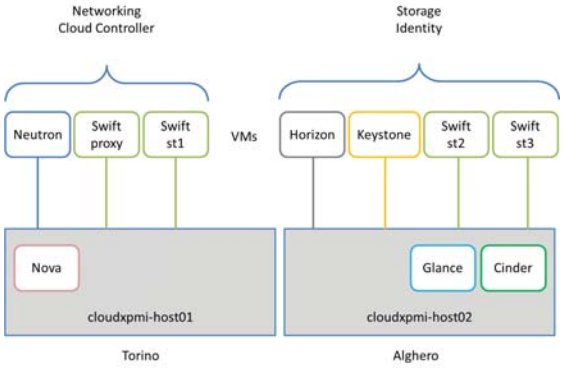

Fig.1 Subdivision of Open Stack functions between our two Italian data centers of Alghero and Turin. Services Nova and Heat have a physical machine on the server of Turin and all other services are arranged on virtual nodes

The network configuration of the platform is illustrated in Figure 2. We split the architecture in two different Italian data centers located in Alghero and Turin. Each server stands on a virtual private LAN: we have a server in Turin that 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 is used for the communication between hosts and virtual machines on which is installed the entire platform OpenStack. This network has been configured as VPN, so as to enable secure communication between the nodes. The server of Turin has been configured as the VPN server, the bridge *tap0*, using 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*.
The data network is used to allow communication between virtual machines. For this network OpenStack defines another VPN, through three tunnels type *Generic Routing Encapsulation (GRE).* A tunnel is established between the two hosts and the other two tunnels between the same host and the Neutron node.

The service Keystone provides authentication and accounting for the entire platform and it is installed on a dedicated virtual machine on the physical server of Alghero (hostname: cloudxpmi-host02). 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 of Alghero but externally with respect to the platform OpenStack.

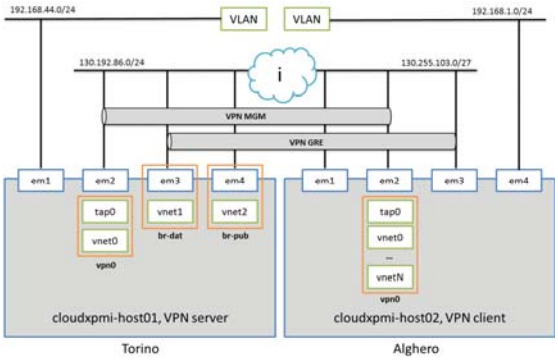

Fig.2 General network configuration of the Cloud platform

3.2 Integration of biometric recognition with cloud computing platform

Biometric authentication is proposed in order to access to the Cloud platform; a Client desktop application has then been implemented on the user side and a dedicated authentication server (AS) has been connected to the Keystone module.

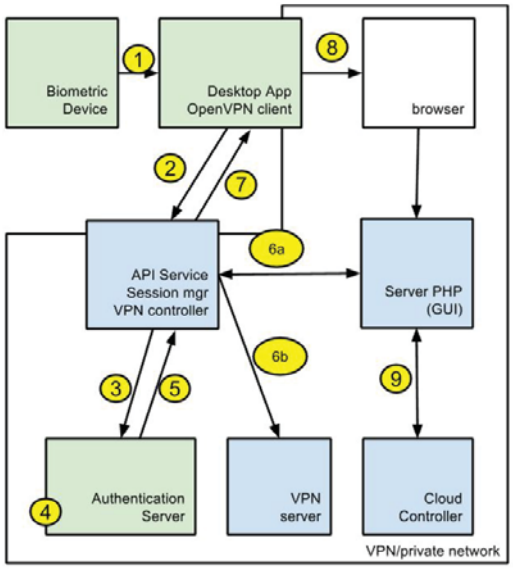

Fig.3 Authentication to the Cloud platform.

The authentication to the Cloud platform procedure, as shown in Figure 3, is based on the following steps :

1. The user interacts with the fingerprint scanner through a desktop client application. Such application produces a *model file* from the original fingerprints
2. The desktop client application contacts the API service, through a REST call (POST) and sends the *model file*. The call is asynchronous, so no one is waiting for a response. At the same instant a series of calls (GET) require the identifier of the user session to be created.

3. The service REST API connects to the Authentication Server (AS) asking for authentication and sending to this purpose the *model file* through the REST API (POST).
4. The AS performs the comparison of the *model file* with the content of its database. Once recognized, the user can retrieve the username and password associated with it.
5. If authentication is successful, the AS sends to the service API username and password (in response to the call POST)
6. Having username and password the API server :
   a)creates a new session on the web server;
   b)creates a new route on the VPN server that enables the user to access its subnet;
7. The API server responds to GET requests from the Client desktop application by sending the session ID.
8. The Client desktop application opens the browser with the address of the web graphical user interface (GUI) and the session ID.
9. The web server, knowing the username and password related to the session ID, can contact the cloud controller to manage cloud services.

Currently, a VPN is placed between the system and the user. 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. A detailed overview of communication services is also given in Figure 4.

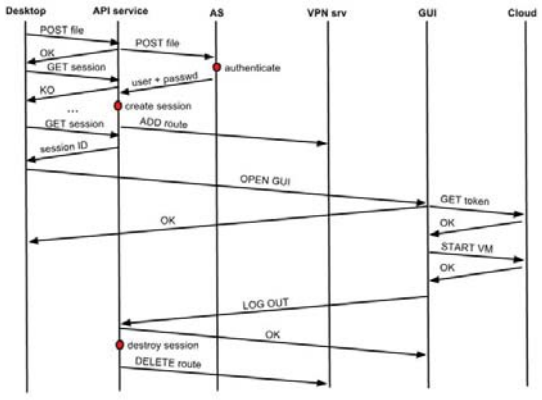

Fig.4 Communications between the authentication services of Cloud

With regard to the figures 3 and 4 it is worth to highlight some important 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).

3.3 Multimodal biometric recognition

The Client desktop application is composed by a software for the enrollment of new users and an authentication application.

During enrollment, the new user's fingerprint is converted into a compact representation, called *model*; this *model* will be used to recognize the user. It is not necessary to store the fingerprints in the AS database; only the *models* are recorded.

The features to produce the model are obtained by using the Scale Invariant Feature Transform (SIFT) representation [9-11]. Recently SIFT has emerged as a cutting-edge methodology in general object recognition as well as for other machine vision applications [8-12]. 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 [14-15], 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 [12]. A fusion module takes the final decision.

3.4 Security of data residing

Cloud computing services and applications are faced with many challenges, including latency, unreliability, malicious behavior, mostly related to the public shared environment in which 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. To cope with these challenges innovative architectures and algorithms have to be developed.

In this project a secure and high availability data chunking solution based on innovative distributed cloud storage

architecture is proposed. The basic idea is to shard data in small chunks and spread them on different VMs hosted on cloud computing. The complete control of the distribute storage system is delegated to the user who hosts the master node of the system, as shown in Fig.6. 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 behaviour of the cloud provider.

Some of the features of the proposed solution are:

1. distributed storage system implemented in cloud, with client-server architecture and partially trusted environment;

2. security granted by chunking data and spreading it on different nodes (virtual machines) possibly hosted by different cloud providers;

3. availability and resiliency ensured by the redundancy of nodes and replica of chunks;

4. the possibility to use different cloud providers prevent also the so-called vendor "lock-in".

3.4.1 Distributed Storage Systems

There are two main categories of distributed storage systems architectures: Peer-to-Peer and client-server [16]. 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 [16].

In Distributed Storage Systems data can be replicated across multiple geographical sites to improve redundancy, scalability, and data availability, as shown in figure 5.

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 [17]. Some examples of the most known distributed storage systems are HDFS[18], Ceph [19], MooseFS [20], mongoDB [21].

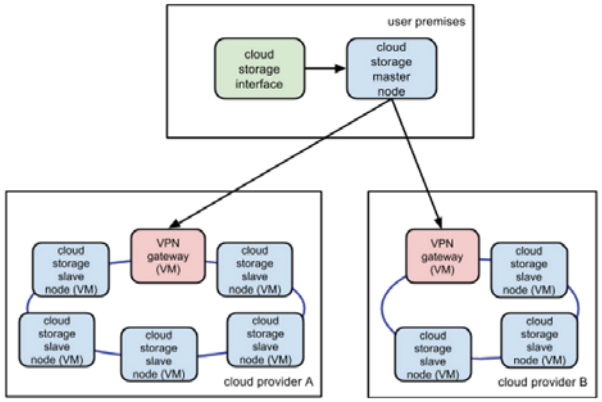

Fig. 5 – Architecture of the distributed storage system

3.4.2 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 file system namespace and regulates client access to files and the slave node which 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.

3.4.3 Hardware and software

The server used for the Cloud Service is a Dell PowerEdge R620 with processor 2x Intel Xeon E5-2650 with high performances : 2GHz, 8C, cache 20MB, 8GT/s QPI, 95W, 256 RAM and 16 physical cores.

The AS is a Dell PowerEdge R210II with processor Intel Xeon E3-1220v2 Processor :3.1GHz, 4C/4T, 8M, Cache and 32GB RAM.

The fingerprints are acquired using the HI-SCAN PRO BIOMETRIKA scanner (FTIR, 500 dpi, 1" x 1").

The installed version of OpenStack is Icehouse 01.02.2014 issued on August 8, 2014. The installed operating system is a Linux distribution, free and open source: Ubuntu version 14:04 x86_64 server Long Term Support (LTS) with KVM (Kernel integrated) and included in the official repositories.

4 Conclusion

A complete system for web applications and data management over the Cloud, coupled with strong biometric authentication, is presented. 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 based on a distributed cloud storage architecture is proposed. 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.

5 Acknowledgment

This work was supported in part by Regione Autonoma Sardegna LR 7 2007, N.7: "Promozione della ricerca scientifica e dell'innovazione tecnologica in Sardegna", *Piattaforme di Cloud computing per le PMI*, Codice: CRP-61647.

6 References

[1] M.K Srinavasin at "'State of the art cloud computing security taxonomies: a classification of security challenges in the present cloud computing environment, ICACCI 2012 *Proceedings of the International Conference on Advances in Computing, Communications and Informatics* Pages 470-476, 2012.

[2] European Commission, "Exploiting the potential of cloud computing in Europe," 27 September 2012. [Online]. Available: http://europa.eu/rapid/press-release_MEMO-12-713_it.htm.

[3] NIST, «NIST Cloud Computing Standards Roadmap,» 2013.

[4] M. K. Yinqian Zhang, «Cross-VM Side Channels and Their Use to Extract Private Keys,» in CCS'12, Raleigh, North Carolina, USA, 2012.

[5] Ross, Arun A., Karthik Nandakumar, and Anil K. Jain. *Handbook of multibiometrics*. Vol. 6. Springer, 2006.

[6] Vielhauer, Claus. "Biometric user authentication for IT security: From fundamentals to handwriting (Advances in information security, Vol. 18)." 2005.

[7] OpenStack, «OpenStack Cloud Administrator Guide,» [Online]. Available: http://docs.openstack.org/admin-guide-cloud/content/.

[8] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In IEEE Conf. on Computer Vision and Pattern Recognition, 2004.

[9] D. Lowe. Object recognition from local scale-invariant features. In Int. Conf. on Computer Vision, pages 1150–1157, 1999.

[10] D. Lowe. Local feature view clustering for 3d object recognition. In IEEE Conf. on Computer Vision and Pattern Recognition, pages 682–688, 2001.

[11] D. Lowe. Distinctive image features from scale-invariant keypoints. Int. Journal of Computer Vision, 60(2):91–110, 2004.

[12] Bicego, M., Lagorio, A., Grosso, E., & Tistarelli, M. (2006, June). On the use of SIFT features for face authentication. In Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on (pp. 35-35). IEEE.

[13] Jain, A. K., Bolle, R., & Pankanti, S. (Eds.). (1999). Biometrics: personal identification in networked society. Springer Science & Business Media.

[14] G. Heusch, Y. Rodriguez, and S. Marcel. Local binary patterns as an image preprocessing for face authentication. IDIAP-RR 76, IDIAP, 2005.

[15] G. Zhang, X. Huang, S. Li, Y. Wang, and X. Wu. Boosting local binary pattern (lbp)-based face recognition. In L. 3338, editor, SINOBIOMETRICS 2004, pages 179–186. Springer Verlag, 2004

[16] M. Placek and R. Buyya, The University of Melbourne, A Taxonomy of Distributed Storage Systems, Reporte técnico, Universidad de Melbourne, Laboratorio de sistemas distribuidos y cómputo grid (2006).

[17] M. Assuncao, R. Calheirosb, S. Bianchia, M. Nettoa, R. Buyya, "Big Data Computing and Clouds: Challenges, Solutions, and Future Directions", Journal of Parallel and Distributed Computing, 2014

[18] https://hadoop.apache.org

[19] http://ceph.com/

[20] http://www.moosefs.org/

[21] https://www.mongodb.org/