Chapter 33
Accountability for Federated Clouds

Thiago Gomes Rodrigues, Patricia Takako Endo, David W.S.C. Beserra, Djamel Sadok, and Judith Kelner

33.1 Introduction

The computational service delivery has evolved according to users' necessities, increasing the rigorousness of the requirements along the years; it can be divided into the following distinct eras (Fig. 33.1): monolithic, client-server, web, Service-Oriented Architecture (SOA), and cloud computing [3, 8, 9]. Focusing on the last era, cloud computing has brought many advantages for both provider and customer. From the provider perspective, clouds facilitate the infrastructure management, providing resource control mechanisms (dynamic allocation, elasticity) and at the same time, minimizing the costs to a new infrastructure investment [15]. On the other hand, from the customer perspective, cloud computing represents a good and easy model to rent computational resources, offering on-demand self-service with a pay-as-you-go model.

However, cloud computing suffers from many weaknesses, and according to the NIST, *"security, interoperability, and portability [. . .] are the major barriers to broader adoption"* of the clouds [18]. Beyond that, evidence distributed among different machines with different hardware architectures, Operational Systems (OS), and infrastructures also increases the complexity in performing accountability properly.

T.G. Rodrigues (✉) • D. Sadok • J. Kelner
Federal University of Pernambuco, Recife, Brazil
e-mail: trodrigues@gprt.ufpe.br; jamel@gprt.ufpe.br; jk@gprt.ufpe.br

P.T. Endo
University of Pernambuco, Caruaru, Brazil
e-mail: patricia.endo@upe.br

D.W.S.C. Beserra
Centre de Recherche en Informatique, Université Paris 1 Panthéon-Sorbonne, Paris, France
e-mail: David.Beserra@malix.univ-paris1.fr

© Springer International Publishing AG 2018
K. Daimi (ed.), *Computer and Network Security Essentials*,
DOI 10.1007/978-3-319-58424-9_33

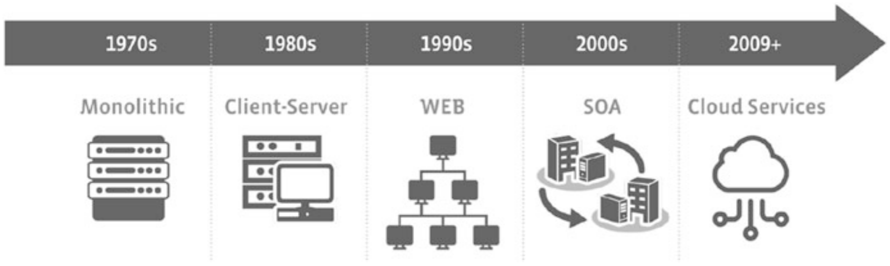


Fig. 33.1 Computational service delivery eras

Considering clouds interconnection, some authors named it as cloud federation or inter-cloud. The first uses the providers' interface, while inter-cloud is based on standards and open interfaces [23]. Nevertheless, both approaches aim to obtain interoperability [3] by offering new services that combine components from different clouds; in this way, we decided to use cloud federation and inter-cloud nomination interchangeably in this work.

Cloud federation provides scalability, hardware heterogeneity [5], more availability when compared against traditional clouds, geographic distribution to deal with disaster recovery and low latency access, avoiding vendor lock-in, as well as cost efficiency and energy savings [1, 3, 6]. However, interoperability between different clouds may be affected by integration problems or security breaches. In addition to the security concerns inherent to virtualized environments, cloud interoperation raises more security challenges, such as trust, authorization and identity management, and policy and interoperability control [23].

Furthermore, in federated scenarios, it is necessary to provide transparency between operations in order to guarantee that applications are fulfilling the Service Level Agreements (SLAs), that services are being billed properly and that security routines are being applied. For that, performing accountability is crucial but hard at the same time, because the evidences are spread across different servers, infrastructures, and applications, and the federation members may have different infrastructures with their own and specific security procedures, systems, and controls.

Considering accountability as *"the acknowledgement or assumption of responsibility for actions and decisions of persons or organizations that affects others"* [16], detail "when, where, what, who, and how" is one of the key functions for a proper accountability mechanism. The other key function of a proper accountability mechanism is store and retrieve the evidences, and increase the liability, transparency, responsiveness, responsibility, and controllability. Therefore, if the cloud computing environment allows accountable routines, it will be considered more trustworthy and, consequently, it will attract more customers.

Nonetheless, the current approaches do not support the audit process, infrastructure management, planning and billing at the same time. To properly support these four procedures, the information from infrastructure, virtualization and application layers must be collected. The related works of the state of the art that provide accountability routines [4, 6, 19, 20] collect information at one layer, supporting only billing process. They do not consider legal aspects related to the registry's safeguard, neither are able to provide different configurations according to the user needs. They also do not provide alerts routine when detect some SLA metric violation, contractual clauses, or nonstandard activities.

Considering the aforementioned motivations, the main objective of this chapter is to present relevant concepts about security and accountability in federated cloud, discuss main challenges in this research area, and propose a Cloud-based Accountability Framework for Federated Clouds, named CloudAcc, capable to enable audit process, infrastructure management, planning, and billing collecting the evidences dispersed through whole infrastructure.

This chapter is organized as follows: Sect. 33.2 discusses about cloud environments. Section 33.3 presents why accountability is important for federated clouds. After that, Sect. 33.4 describes the CloudAcc Framework thoroughly each framework module. Section 33.5 presents the CloudAcc implementation in a real federated cloud infrastructure. Finally, Sect. 33.6 will present the final considerations and future remarks.

33.2 Cloud Environments

Cloud computing has changed the way IT services are consumed. The computing utilities become consumed like other utility services available in the contemporaneity society. The main idea behind the IT services provided by a cloud provider is that the users must to pay only by the consumed resources. Furthermore, consumers should not expend money building a new and complex IT infrastructure to support their business needs. Currently, in order to expand the IT resources and provide more availability for their costumers, some cloud providers are operating in cooperation with other providers, composing what is called as cloud federation.

According to [23], cloud federation is the practice of interconnecting different cloud infrastructures, in order to provide scalability and hardware heterogeneity [6]; other key features include availability and disaster recovery, geographic distribution and low latency access, interoperability and avoiding vendor lock-in, legal issues and meeting regulations, as well as cost efficiency and energy savings [1, 3].

Cloud interoperability can be obtained by interface standardization or brokering. In an interface standardization approach all providers adopt the same interface. Nevertheless, developing a set of standards is difficult and hard to be adopted by all providers. The broker approach uses a service broker to translate messages among cloud interfaces making them interoperate with each other. A combination of these two approaches aforementioned often occurs in practice.

Despite the advantages provided in cloud federation considering the security aspects, the federation increases the security concern. Different access and authorization policies should compromise the access control management and communication problems may affect the services' availability. In addition, problems related to identity and access management, data security and trust and assurance are closely linked to these type of environment.

These security issues are even more critical when we consider other current scenarios integrated with cloud computing, such as fog computing and Internet of Things (IoT) communication. Both approaches have as main goal the idea to bring the computing more close to the users, when the process at the cloud is not viable (because it can take a long time to get the process result, for instance). In this case, due to several and different components existing in the environment, the accountability mechanism performs a crucial role, but even more complex too.

33.2.1 Accountability for Federated Clouds

Accountability is a concept that has different definitions. In [13], the author proposes five important aspects of accountability: transparency, liability, controllability, responsibility, and responsiveness. The author considers transparency and liability as the main important foundations of the accountability's concept. Transparency provides the way needed to assess organizational performance results. Liability means that organizational members should be held liable for their actions.

Another accountability concept that is closely connected to cloud computing environments is presented in [24]: *"accountability is a concept to make the system accountable and trustworthy by biding each activity to the identity of its actor. Such biding should be achieved under circumstance that all actors within the system are semi-trusted."*

In [7], the European Union Agency for Network and Information Security (ENISA) defines that accountability offers three capabilities: validation, attribution, and evidence. Validation allows users to verify if everything works as expected. Attribution allows, in the case of fault, that the responsibility can be assigned. Evidence is an artifact used to convince when a dispute arises. They identified *"falsifying record collection," "tampering with records,"* and *"destroying or suppressing the transmission of records"* as main threats against integrity of the collection, storage, and transmission of accountability registers.

These threats are also present in cloud computing and federated cloud environments, and compromise the forensic investigation [10]. Furthermore, an accountable infrastructure (Fig. 33.2) must provide sufficient evidences for: an audit process, infrastructure management, planning, and billing. Users should be able to trust that their contract will be fulfilled following the SLA, once their records can be checked.

Mixing the concepts aforementioned, we consider accountability as a mechanism that provides more transparency, responsiveness, responsibility, controllability, and liability through attribution or responsibility, evidence collection and validation, giving support for auditing, billing, management, and planning process.

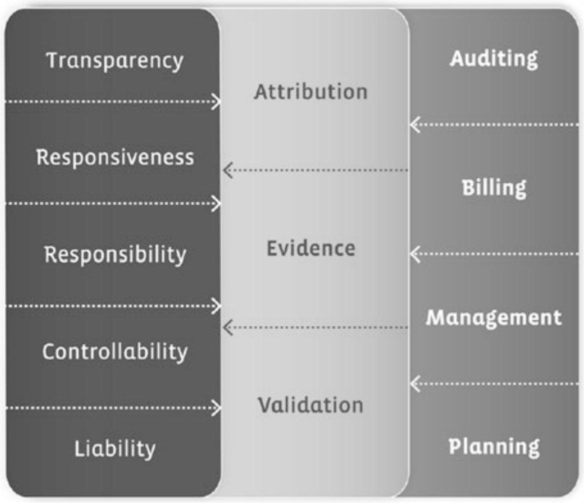


Fig. 33.2 Accountable infrastructure

33.3 Why Accountability Is Important for Federated Clouds?

Despite existing effective countermeasures that mitigate and/or solve cloud security threats, there are many barriers to adopt cloud computing as deployment model. Why cloud computing and inter-cloud computing are not being used doubtless as infrastructure to support new applications and services? The answer for this question is that stakeholders do not trust that the cloud providers apply properly the control needed to provide security. Then we can conclude that applying security controls is not enough if the stakeholders do not trust that they are effectively applied.

Trust in cloud is a term that involves the confidence about confidentiality, integrity, availability, accountability, and auditability routines provided by a cloud provider [10]. This is the main concern considered in the decision to adopt or not a cloud computing environment. In this case, cloud consumers must trust in cloud providers to grant proper control mechanisms in order to avoid losing data confidentiality, integrity, and availability with accuracy in accounting [2]. A proper accountability routine increases the trust because it allows correctness billing, transparency and tractability, and increments the auditability support.

In a federated scenario, cloud providers must trust each other. In addition, the reputation of each federated member directly affects the global reputation [11, 23]. The establishment of trust in interconnected clouds is a complex and non-trivial activity because each federation member has its own security policies and procedures [12]. Federation exposes users' assets to new security concerns that were avoided when it was in an internal infrastructure. These risks include users' rights, transitive trust issues, and different system security requirements [15].

Public-key Infrastructure (PKI) is the common trust model adopted in cloud environments to provide user access. In federated cloud environments, the trustworthiness is established by adding the Root Certified Authority (CA) certificate in chain-of-trust certificate for all federation members.

In federated clouds, despite the aforementioned security concerns, accountability is used as detective control, increasing the infrastructure security and trust, since the actions can be properly identified and logged. Therefore, strong actions identification and proper log generation and storage are the key requirements for a suitable accounting system.

An appropriate accountability mechanism in a federated scenario must be capable to increase security and confidence among the members of the federation. An accountable infrastructure must provide sufficient evidences for an audit process, infrastructure management, planning, and billing.

The existing projects that try to solve interconnection problems did not consider accountability in whole federated platform. Furthermore, an improper evidence collection affects the infrastructure trustworthiness, transparency, and liability. This work proposes a Cloud-based Accountability Framework for Federated Clouds, named **CloudAcc**, that provides cloud microservices to enable audit process, infrastructure management, planning, and billing collecting the evidences dispersed through whole infrastructure, increasing the trustworthiness, transparency, and liability. CloudAcc considers a multilayer evidence collection performing copy of logs to another infrastructure that will process and store them. Why the log processing is done in another infrastructure? The answer for this question is, because we want to avoid *"falsifying record collection," "tampering with records,"* and *"destroying or suppressing the transmission of records"* problems. In addition, the collection of evidence distributed for whole infrastructure may provide sufficient information, e.g., to detect malicious activities, misconfigurations, and correctness in billing process and SLA fulfillment.

Table 33.1 summarizes the acting areas for existing works focused in cloud federation that collect some accountability information with CloudAcc framework acting areas.

Considering the acting areas summarized by Table 33.1, we recognized that VM resources, services, and infrastructures must be accountable to increase the trustworthiness and provide proper evidence collection for a whole platform.

Table 33.1 Comparison with existing accountability frameworks

Framework	VM resources	Service	Infrastructure	Legal
RESERVOIR	x			
Cloudbus InterCloud	x			
OpenCirrus	x			
STRATOS		x		
CloudAcc	x	x	x	x

Considering the previous motivations, we propose the CloudAcc framework to increase the trustworthiness, transparency, and liability in cloud federation environments, improving the audit process, infrastructure management, planning and billing. In addition, the CloudAcc framework provides the security mechanisms needed to properly manage and store the collected evidences, solving or mitigating the security concerns related to accountability in federated environments.

33.4 CloudAcc Framework

Frequently, the integration between different clouds occurs through cross-signing the root certificate. This solution avoids the need of creating a new public-key hierarchy. In addition to this, the use of proxies is commonly adopted to provide locality transparency. In this work, we consider the use of proxies to provide cloud interconnection, as illustrated in Fig. 33.3.

Figure 33.3 depicts the peer-to-peer federation or interconnection between clouds C1 and C2. In this approach each federation member interconnects its own proxies through a secure tunnel (such as IPSec tunnel) that redirects the incoming requests to appropriate service.

The CloudAcc framework was designed to provide less impact in federation functioning or compromising the available resources from each federation member. In this way, CloudAcc can also be implemented in other scenarios, such as fog computing and IoT devices. Considering this requirement, it is divided into **CloudAcc Agent** and **CloudAcc Core**. Considering the federated scenario depicted in Fig. 33.3, CloudAcc Agent collects the evidences from the federation members C1 and C2, and sends them to another cloud that runs the CloudAcc Core.

The CloudAcc Agent collects the evidences in each federation member (clouds C1 and C2) and sends them through a secure channel for our accountability cloud that runs the CloudAcc Core. The evidences are collected considering three layers: (1) the infrastructure layer; (2) the virtualization layer; and (3) the system layer. The evidences in infrastructure layer can be collected using the Simple Network Management Protocol (SNMP). The administrator must configure the Agent with the credentials in order to properly collect the evidences in infrastructure layer. In virtualization layer, the CloudAcc Agent connects on hypervisor and collects the evidences in this layer. Lastly, in the system layer the CloudAcc Agent collects the system and applications logs.

On the other hand, the CloudAcc Core runs in a different cloud from the accounted federation. The main idea behind this is that providing all processing routines in another infrastructure (accountability cloud), we do not consume the available resources for each federation member. Beyond that, security problems, such as falsifying record collection, tampering with records and destroying or suppressing the transmission of records, can be avoided because we send all logs to other infrastructure, preventing that anybody modifies logs. This requirement is even more relevant if we consider fog computing scenario or IoT communication,

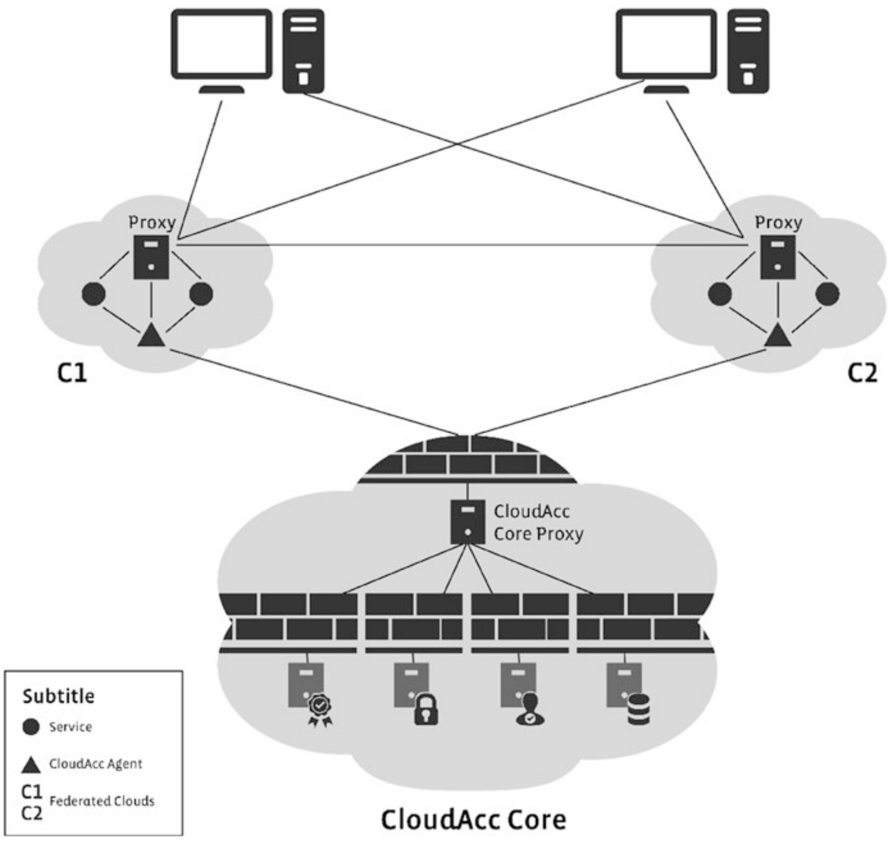


Fig. 33.3 Federated scenario overview

since the components, such as sensors, have less available resources. In addition, the registries stored in a trusted third-party increase the difficult of attacks against accountability because the attacker must corrupt the registries in federation and the registries in accountability cloud.

The CloudAcc Core was designed considering microservices architecture, that supports the security requirements, accountability routines, and legal issues. Moreover, each functionality should be configured according to the users' need. Each microservice has a local firewall that accepts only connections from other CloudAcc microservice or from the CloudAcc Core proxy. The CloudAcc Core is composed of two layers (Fig. 33.4): (1) Supporting Layer, and (2) Service Layer.

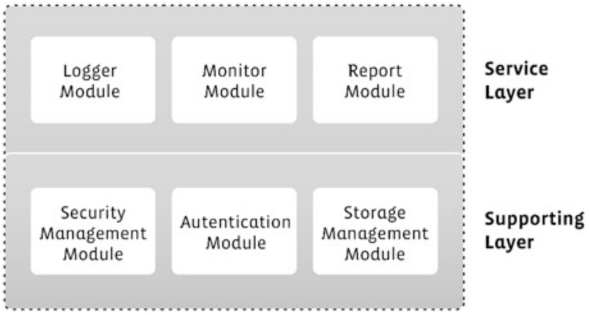


Fig. 33.4 General overview of CloudAcc Core

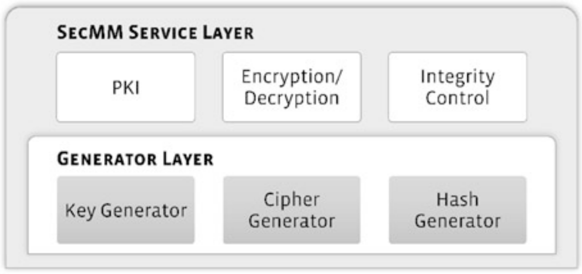


Fig. 33.5 SecMM conceptual model

33.4.1 Supporting Layer

The supporting layer is composed of three modules: Security Management Module (SecMM), Authentication Module (AM), and Storage Management Module (SMM). Each module is independent but can consume the services from other modules. In addition, the supporting layer is responsible to perform security and access control, and store and retrieve all collected evidences.

The SecMM is a set of APIs that implements security routines interfacing hardware for secure computing as Trusted Platform Module (TPM) and Smart Cards to provide an appropriate security management for service infrastructure and client's data. The SecMM implements services, such as Public-key Infrastructure, Encryption/Decryption, and Integrity Control to support the security requirements needed to provide strong communication, confidentiality, integrity, authentication, and non-repudiation. This module is responsible to manage the symmetric and asymmetric keys, hash functions, digital signature, and Message Authentication Code (MAC) to support the security procedures needed to securely manipulate and store the evidences (Fig. 33.5).

The AM provides authentication and access control. We consider two authentication routines: infrastructure authentication and user authentication. In both approaches strong authentication is a mandatory requirement. Infrastructure authentication occurs to grant that all CloudAcc members are properly identified and accountable. CloudAcc considers two strong authentication methods for users: (1) digital certificate and smart card, and (2) a pair username/password with 2-

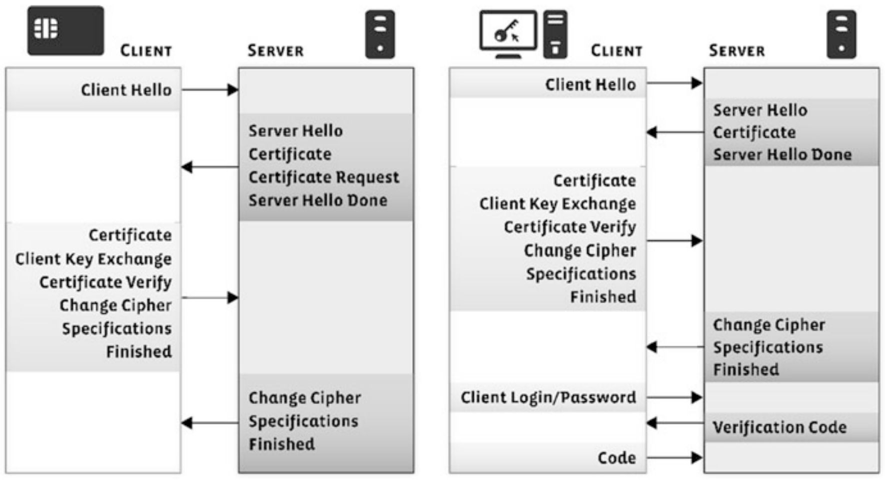
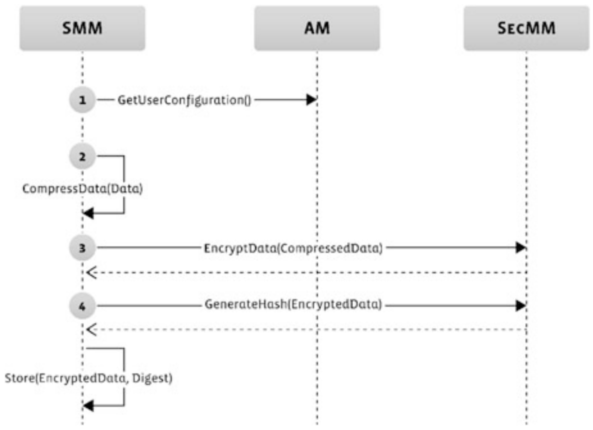


Fig. 33.6 Both client authentication approaches supported

Fig. 33.7 Secure storage sequence diagram


SMM
AM
SecMM
1 GetUserConfiguration()
2 CompressData(Data)
3 EncryptData(CompressedData)
4 GenerateHash(EncryptedData)
Store(EncryptedData, Digest)

step verification. Each authorized user can configure his/her profile with personal settings that will be used to define who will access the user's data, and which cryptography will be used to protect the data (Fig. 33.6).

The SMM is responsible for providing secure storage of users' data based on settings established by themselves. It uses the services provided by SecMM and the configurations set in AM to execute security routines according to the user needs respecting the local legal issues, such as the Brazilian law #12965 or the National Security Systems (NSS) [17].

In Fig. 33.7, the sequence performed to grant the privacy in users' data is depicted. Firstly, the user configuration is retrieved from AM. The second step performs the data compression. After that, SMM uses SecMM to encrypt the compressed data following the configuration set by user. In the fourth step, to perform the integrity control, the encrypted data digest is generated. In the fifth step, SMM stores the data and its digest. The steps 1–4 are enough to grant the user's data privacy.

33.4.2 Service Layer

Service layer implements the core functionalities supported by CloudAcc. It consumes the services implemented in supporting layer to manipulate properly the users' data through interfaces and provides a set of functionalities that should be accessed through a RESTful interface. It is composed of Logger Module (LM), Monitor Module (MM), and Report Module (RM). This layer considers the configuration updated by each user in order to manipulate the evidences, monitoring the performance, and generating the reports according to the users' needs.

The LM is responsible for manipulating all distributed log collected on federated members executing merge/split operations. LM uses the modules presented in supporting layer (Sect. 33.4.1) to perform integrity control, confidentiality, authentication, and security storage.

The MM (Fig. 33.8) is responsible for checking if the SLA is being fulfilled, and if the security routines are in accordance with regulatory compliance. The RM (Fig. 33.9) provides reports about resource consumption, and alerts when the SLA is not fulfilled or some security configuration does not respect the regulatory compliance.

33.5 CloudAcc Implementation in a Real Federated Cloud

The CloudAcc was implemented in two distinct infrastructures, as shown in Fig. 33.10: at Google Cloud Platform (the CloudAcc Core) and at a real federated infrastructure (the CloudAcc Agent). In this example, C1 and C2 are the private clouds that federate their infrastructures.

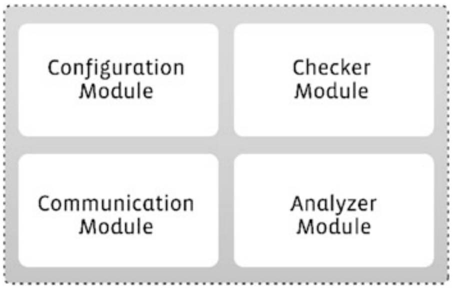


Fig. 33.8 Monitor Module conceptual model

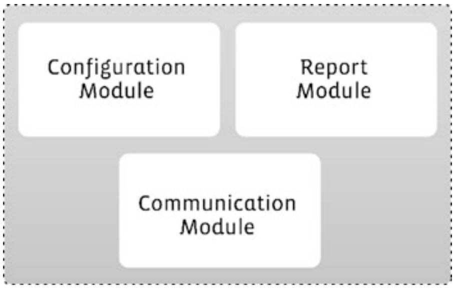


Fig. 33.9 Report Module conceptual model

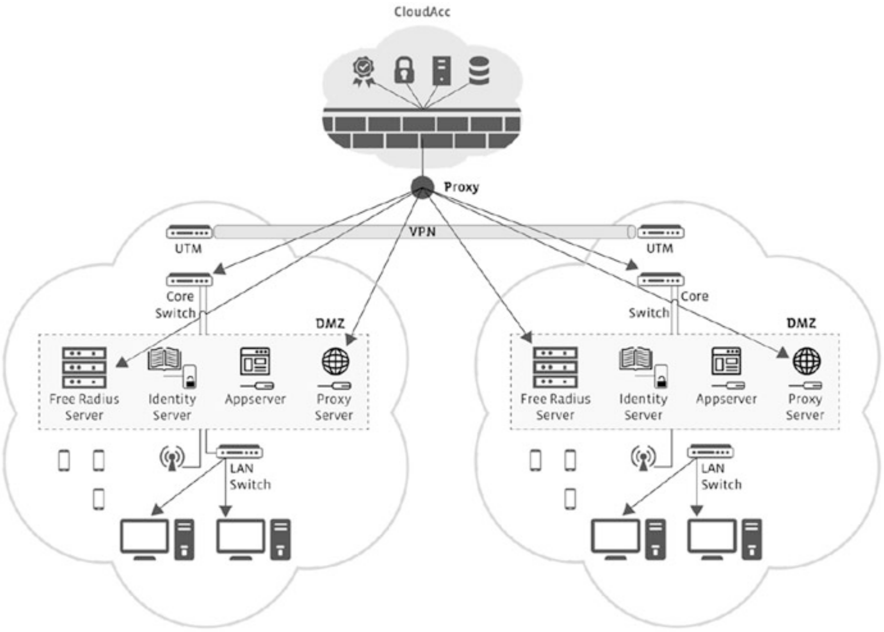


Fig. 33.10 CloudAcc implementation in a real scenario

The CloudAcc Core was implemented in the Google Cloud Platform, that was composed of two sets of machines. The first group is a Google f1-micro instance with one vCPU, 0.6 GB type RAM, 10 GB HD, Debian 8 and MySQL version 5.6.31. The second group is composed of Google g1-small instancies with one vCPU, 1.8 GB type RAM, 10 GB HD, Debian 8, Java SE Runtime Environment version 1.8.0, and Tomcat 8.

The CloudAcc agents were implemented in a real federated infrastructure. We will not provide infrastructure detail that we run the CloudAcc agent because we signed a Non-Disclosure Agreement (NDA). Considering this, Fig. 33.10 overviews our test scenario with generic elements. The black triangle in Fig. 33.10 represents a CloudAcc Agent running, collecting the evidences and sending them to the CloudAcc Core.

Agents collect evidences from three elements: the core switch, the web proxy, and the network authentication server. The web proxy is an instance with 4 vCPUs, 4 GB type RAM, 10 GB type HD, Debian 8, NGINX 1.9. The network authentication server is an instance with 2 vCPUs, 1 GB type RAM, 10 GB type HD, Ubuntu server 14.04, and Freeradius v3. The CloudAcc agent is a Java 1.8 implementation that includes the snmp4j version 2.4.3, jDOM version 2.0.6, libvit version 0.5, and JSON.simple API version 1.1.1.

We performed two types of experiments in our testbed: the performance and the proof-of-concept tests. The performance tests were executed in the Google Cloud Platform; and the second set of tests were performed in the real federated infrastructure. However, due to page limitation, we do not present these results here. For experimental results, please see [21].

33.5.1 CloudAcc Implementation Challenges

Despite the federation members still under the same timezone, we faced clock synchronization problems. In order to properly mount the users' activities across the servers the servers' clocks must still be synchronized. Furthermore, clock synchronization problems may compromise all accountability routines of CloudAcc. Clock synchronization may affect the CloudAcc agent, because the evidence collection in infrastructure and virtualization layers considers the local timestamp as key index of the collected evidence. To overcome these problems, we run a local Network Time Protocol (NTP) server instance that synchronized all the servers' clocks. Despite the security problems involving NTP, the time is the fundamental part for applications that use cryptographic routines (DNSSec, bitcoin, Time Stamping Authority (TSA), etc.) and we decided to adopt NTP in our framework considering the security concerns listed in [14] and their countermeasures.

Regarding the implementation challenges, we consider the CloudAcc core functionalities in microservices, and supporting the Brazilian "Marco Civil" requirements as the main challenging implementation problems. Modeling the CloudAcc in microservices architecture allowed scalability and reuse generated integration and communication problems. When the CloudAcc was deployed in Google Cloud platform, we faced several integration problems. As a workaround to solve the communication and integration problems, we configured internal static IPs for each machine running the CloudAcc microservices.

At last, there was the auditing process established in "Marco Civil" that specifies the log safeguard and the access. Considering the safeguard requirements, CloudAcc has encrypting routines to properly store the clients' information.

33.6 Final Considerations

A proper accountability system should support audit process, infrastructure management, planning, and billing. To support these functionalities, the accountability systems must collect evidences from physical, virtualization, and application layers. However, existing approaches in the literature do not support the four requirements, because they collect evidences in one layer only.

In this work, we proposed and implemented an accountability framework for federated cloud environment, named CloudAcc. Our main objective was implementing a framework to support audit process, infrastructure management, planning, and billing in federated environments according to the users' needs in compliance with the Brazilian "Marco Civil."

Another important aspect about security is its cost to the provider. According to [22], *"although deployment of such technologies may reduce security vulnerabilities and losses from security breaches, it is not clear to organizations how much they must invest in information security."* In this way, as a future work, we plan to analyze what is the cost to implement and maintain the CloudAcc framework and estimate the return of security by using the CloudAcc, as proposed by Sklavos and Souras, [22].

We also want to support accounting in Network Function Virtualization (NFV), that is an initiative that aims to implement network functions supporting interoperation from different hardware vendors. Furthermore, NFV organizes the network functions into service boxes that can be connected to create novel services. The NFV functionalities increase the network capabilities because it enables the creation, for instance, of security services using firewalls, and antispam, reducing the costs of maintaining the infrastructure and improving the use of the resources.

References

1. Aoyama, T., & Sakai, H. (2011). Inter-cloud-computing. *Wirtschaftsinformatik, 53*(3), 171–175.
2. Ardagna, C. A., Asal, R., Damiani, E., & Vu, Q. H. (2015). From security to assurance in the cloud: A survey. *ACM Computing Surveys (CSUR), 48*(1), 2.
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM, 53*(4), 50–58.
4. Avetisyan, A. I., Campbell, R., Lai, K., Lyons, M., Milojicic, D. S., Lee, H. Y., Soh, Y. C., Ming, N. K., Luke, J. -Y., & Namgoong, H. et al. (2010). Open cirrus: A global cloud computing testbed. *IEE Computer Society, 43*(4), 35–43.
5. Barreto, L., Fraga, J., & Siqueira, F. (2015). Cloud federations and security attributes. In *2015 XXXIII Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)* (pp. 140–149). New York: IEEE.
6. Buyya, R., Ranjan, R., & Calheiros, R. N. (2010). Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Algorithms and architectures for parallel processing* (pp. 13–31). Heidelberg: Springer.

7. Castelluccia, C., Druschel, P., Hübner, S., Pasic, A., Preneel, B., & Tschofenig, H. (2011). Privacy, accountability and trust-challenges and opportunities. ENISA [Online]. Available: http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/pat-study/atdownload/fullReport.
8. Dagger, D., O'Connor, A., Lawless, S., Walsh, E., & Wade, V. P. (2007). Service-oriented e-learning platforms: From monolithic systems to flexible services. *Internet Computing, IEEE, 11*(3), 28–35.
9. Erl, T. (2008). *Soa: Principles of service design* (Vol. 1). Upper Saddle River: Prentice Hall.
10. Farina, J., Scanlon, M., Le-Khac, N. -A., Kechadi, M., et al. (2015). Overview of the forensic investigation of cloud services. In *2015 10th International Conference on Availability, Reliability and Security (ARES)* (pp. 556–565). New York: IEEE.
11. Fernandes, D. A .B., Soares, L. F. B., Gomes, J. V., Freire, M. M., Inácio, P. R. M. (2014). Security issues in cloud environments: A survey. *International Journal of Information Security, 13*(2), 113–170.
12. Fernandez, E. B., Monge, R., & Hashizume, K. (2016). Building a security reference architecture for cloud systems. *Requirements Engineering, 21*(2), 225–249.
13. Koppell, J. G. S. (2005). Pathologies of accountability: Icann and the challenge of "multiple accountabilities disorder". *Public Administration Review, 65*(1), 94–108.
14. Malhotra, A., Van Gundy, M., Varia, M., Kennedy, H., Gardner, J., & Goldberg, S. (2016). The security of NTP's datagram protocol. Cryptology ePrint Archive, Report 2016/055. http://eprint.iacr.org/2016/055.
15. Mell, P., & Grance, T. (2011). The NIST definition of cloud computing [online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.
16. Nakahara, S., & Ishimoto, H. (2010). A study on the requirements of accountable cloud services and log management. In *2010 8th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT)* (pp. 1–6). New York: IEEE.
17. National Security Agency and Central Security Service. (2016). Information assurance directorate. https://cryptome.org/2016/01/CNSA-Suite-and-Quantum-Computing-FAQ.pdf, Accessed: 2016-09-27.
18. NIST. (2010). Cloud computing. https://www.nist.gov/itl/cloud-computing. Accessed: 2016-05-27.
19. Pawluk, P., Simmons, B., Smit, M., Litoiu, M., & Mankovski, S. (2012). Introducing stratos: A cloud broker service. In *2012 IEEE Fifth International Conference on Cloud Computing* (pp. 891–898). New York: IEEE.
20. Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I. M., Montero, R., Wolfsthal, Y., Elmroth, E., Caceres, J., et al. (2009). The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development, 53*(4), 4–1.
21. Rodrigues, T. G. (2016). *Cloudacc: A cloud-based accountability frameworkfor federated cloud*. PhD Thesis.
22. Sklavos, N., & Souras, P. (2006). Economic models & approaches in information security for computer networks. *IJ Network Security, 2*(1), 14–20.
23. Toosi, A. N., Calheiros, R. N., Buyya R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR), 47*(1), 7.
24. Yao, J., Chen, S., Wang, C., Levy, D., & Zic, J. (2010). Accountability as a service for the cloud. In *2010 IEEE International Conference on Services Computing (SCC)* (pp. 81–88). New York: IEEE.