Future Generation Computer Systems 90 (2019) 1–19

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An approach for the secure management of hybrid cloud–edge environments

Antonio Celesti [a,*], Maria Fazio [a], Antonino Galletta [a], Lorenzo Carnevale [a], Jiafu Wan [b], Massimo Villari [a]

[a] Department of Engineering, University of Messina, Messina, Italy
[b] School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China

HIGHLIGHTS

- CoT implies that applications must be moved from the center to the edge of network.
- Secure instant-messaging solutions for Cloud-to-Edge systems are required.
- We overcome such a gap following the Cloud Security Alliance (CSA) guidance.
- We improve confidentiality, integrity, authenticity and non-repudiation.
- Experiments show that security does not negatively affect the overall performances.

ARTICLE INFO

Article history:
Received 10 January 2018
Received in revised form 21 May 2018
Accepted 24 June 2018
Available online 19 July 2018

Keywords:
Cloud computing
Edge computing
Management
Communication
Security

ABSTRACT

The Cloud-of-Things (CoT) paradigm is a challenging approach to manage IoT applications exploiting Cloud resources and services. In order to avoid latency in Cloud–IoT communications, the management of time-sensitive services has to be moved to the edge of the CoT. To this aim, a secure Cloud-to-Edge environment for seamless management of IoT applications is necessary. The realization of a performing and secure Cloud-to-Edge middleware solution is a very strategic goal for future business CoT services. Thus, it needs to be deeply investigated, as highlighted by the Cloud Security Alliance (CSA). A valuable approach to develop an efficient Cloud-to-Edge system is based on an instant-message communication solution. In current Cloud environments, a Message Oriented Middleware (MOM) based on an Instant Message Protocol (IMP) provides good performance, but overlook security requirements. In this paper, we aim at overcoming such a gap following the CSA guidelines. In particular, we discuss the involved issues for improving such a kind of Cloud-to-Edge system in order to achieve data confidentiality, integrity, authenticity and non-repudiation. Moreover, we analyze a real case of study considering a MOM architectural model. Experimental results performed on a real testbed show how the introduced secure capabilities do not affect the overall performances of the whole middleware.

1. Introduction

With the advent of Cloud-of-Things (CoT) paradigm, new challenges arose, such as the real-time communication of many smart devices with a central coordination unit. However, traditional computing systems based on the Cloud paradigm do not support them. In order to avoid latency in Cloud–IoT communications, the management of time-sensitive services has to be moved to the edge of the CoT. To this aim, a secure Cloud-to-Edge environment for seamless management of IoT applications is necessary [1,2]. The communication system of a Cloud-to-Edge middleware is quite complex because it is necessary to balance performance and security management, and this is not trivial at all. In fact, considering a worldwide CoT environment, several issues need to be addressed. On one hand, the Cloud-to-Edge middleware needs to quickly react to changes. For example, according to the popular phrase of Benjamin Franklin "the time is money", a Service Level Agreement (SLA) violation might cause loss of money for a Cloud provider. On the other hand, a security leak can imply the disclosure of private data or cyber attacks. This can have also economic implications for the CoT provider. For this reason the Cloud Security Alliance (CSA) [3] has picked out the critical aspects of Cloud security. According to the CSA guidance, security and privacy have to ensure

* Corresponding author.
E-mail addresses: acelesti@unime.it (A. Celesti), mfazio@unime.it (M. Fazio), angalletta@unime.it (A. Galletta), lcarnevale@unime.it (L. Carnevale), mejwan@scut.edu.cn (J. Wan), mvillari@unime.it (M. Villari).

https://doi.org/10.1016/j.future.2018.06.043
0167-739X/

the availability of services, resource access control, vulnerability mitigation, privacy of the audited user data. As CoT is an emerging paradigm, software architects have to deal with the lack of ad-hoc security standards and of a consolidated vulnerability model as point of reference.

How usually happens in emerging ICT technologies, software architects start to face new technology issues adapting existing systems and solutions to address the new requirements. The same thing is happening with Cloud and Edge computing, but such a strategy is not resulting effective. As a consequence, researchers and software designers are looking at new innovative approach. A promising approach is the adoption of a Message Oriented Middleware (MOM) for Cloud-to-Edge management. In particular, a MOM is based on an Instant Messaging Protocol (IMP) and can considerably simplify the deployment of CoT applications and services also involving devices at the edge, because it allows separating the communication and signaling system from the business logic. At present, to the best of our knowledge, existing IMPs allow achieving a high level of reactiveness and good communication performance, but they do not offer an adequate degree of security.

In our previous works [4,5], we analyzed the performance of secure communications for the management of federated Cloud and IoT environments. In this paper, we analyze the impact of the security of a MOM for Cloud-to-Edge systems based on an IMP. More specifically, we will discuss how the overall communication system can be secured considering both the digital signing and data encryption mechanisms. In order to plan a security strategy, we analyze both the intra-module and the inter-module communications of a MOM based on XMPP [6].

Regarding security, although XMPP supports the SASL/TLS technologies, it presents some limitations. In order to overcome such a gap we will consider the *XEP-0373* [7] specifications, which describe the use of the XEP protocol with the Open Pretty Good Privacy (OpenPGP) encryption standard [8] for the integration of authentication and data encryption functionalities.

The rest of the paper is organized as follows. Section 2 describes related works. More detailed motivations are discussed in Section 4.2. Details about the Security Model applied are provided in 4. In Section 5, we discuss how a MOM for Cloud/Edge computing can be secured, considering in particular the MOM4Cloud architectural model [9]. Description of secure communication techniques is discussed in Section 6. Considering the CLoud-Enabled Virtual EnviRonment (CLEVER) [10] as case study, that is an implementation of MOM4Cloud, a few implementation highlights regarding the development of specific security features are discussed in Section 7. Experiments evaluating the communication overhead for security management are discussed in Section 8. Section 9 concludes the paper.

2. Related works

Security in emerging hybrid Cloud/Edge environments is an emerging topic. In fact, the need to move resources and services from the Cloud to the Edge raises several technical issues, especially from a security point of view.

The influence and impact of mobile Edge computing on existing communication systems is discussed in [11]. In particular, by analyzing existing Cloud systems, it is highlighted how even though the bind among Cloud and Edge is not so evident, there are several important features, such as security and resilience that have to be investigated. Challenges regarding the interconnection between Cloud and Edge computing environments are discussed in [12]. In particular, it is underlined that, in order to achieve real benefits, it is necessary to achieve high throughput under high concurrent accesses, real-time processing performance, mobility support, and data persistency. The need to enhance traditional Cloud security solutions for Edge computing environments is discussed in [11]. Specifically, a holistic analysis shows the presence of new security challenges, threats, and mechanisms inherent to the Edge computing paradigm, but also the presence of potential synergies with other similar paradigms. A fuzzy-based security approach for mobile Fog and Edge computing to handle a scenario in which security services change according to mobile users is presented in [13]. In particular, a multi-criteria decision making methodology that is based on an innovative extension of the hesitant fuzzy rough set theory fused with a hesitant fuzzy soft set is presented. An alternative approach based on the fuzzy security service chaining concept for sustainable mobile Fog and Edge computing is discussed in [14]. The proposed architecture decouples the needed security mechanisms from physical resources. Moreover, a security proxy enables the compatibility with traditional security functions. A biometric security through visual encryption approach for Edge and Fog computing named Zero-watermarking is discussed in [15]. It is aimed at helping to protect the ownership of multimedia contents that are easy to copy and distribute. In particular, a biometric security solution able to analyze face images that does not impact the visual quality of images by using both visual cryptography and zero-watermarking is presented. A security analysis of mobile Edge computing consisting in the context of the European funded SESAME project is discussed in [16]. In particular, the security analysis of a piece of framework deployed in virtualised small cell networks extended in the broader of a 5G wireless network environment is discussed. Always in the domain of mobile Edge computing environment whose borders are interconnected with 5G wireless network, a study on security of dense small cells, which exploit Network Functions Virtualization (NFV) and that move the intelligence into the edges of the network is discussed in [17].

Most of the aforementioned solutions do not consider the presence of firewalls (whose ports have to be opened) that represent obstacles for the secure communication between the Cloud to Edge environments and vice-versa. Instead the last one, requires the creation of overlay networks by means of NFV mechanisms that have to be configured on network devices. In this paper, we propose an alternative communication system solution for the interconnection between the Cloud and the Edge that is able to work at web layer (on port 80) and that is capable of bypassing firewalls (because port 80 is commonly opened), guaranteeing at the same time security and privacy.

3. Motivation

CoT is having a continuous growth in terms of services availability and differentiation. However, it is still not mature in definition of worldwide recognized standards. In particular, the lack of security standards represents one of the major points of weakness of CoT solutions. In this area, CSA conducted a valuable work, and further security aspects that involve Clouds and their stakeholders have been discussed during the last years. In this section, we report and analyze some hints extrapolated from the CSA guidance [18], which promotes the following:

- a common level of understanding between the consumers and providers of Cloud Computing regarding the necessary security requirements and **attestation of assurance**;
- an independent research into Best Practices for Cloud Computing security;
- educational and awareness programs on the appropriate uses of Cloud Computing and Cloud solutions;
- the creation of wide consensus lists of issues and guidance for Cloud security assurance.

Our work arises from a critical investigation of security requirements and issues reported into the CSA guidance. Specifically, we explore the Governance and Operations Domains in the following.

3.1. Governance domains

Governance domains address strategic and policy issues within a cloud computing environment. They include:

- *Governance and Enterprise Risk Management*. Providers should:
  1. have regular third party risk assessments;
  2. require listings of all third party relationships of the Cloud provider;
  3. understand the Cloud providers' key risk and performance indicators and how these can be monitored and measured from a customer perspective;
  4. divulge all policies, procedures and processes comprising the Cloud providers' Information Security Management System (ISMS).
- *Legal Issues and Contracts*. Contracts are the key legal enforcement mechanism and must be negotiable to reflect any organizations' unique needs and the dynamic nature of Cloud computing.
- *Compliance and Audit*. Classify data and systems to understand compliance requirements and understand data locations, in particular the copies of data that are made and how they are controlled.
- *Information Management and Data Security*. Understand the logical segregation of information and protective control mechanisms.

3.2. Operational domains

Operational domains focus on more tactical security concerns and implementation within the architecture. They include:

- *Traditional Security, Business Continuity and Disaster Recovery*. Customers should perform on-site inspections of Cloud provider facilities whenever possible and should identify physical interdependencies in provider infrastructure.
- *Data Center Operation*. While the adopted technology and the reference Cloud architectures will differ, they must all be able to demonstrate comprehensive compartmentalization of systems, networks, management, provisioning and personnel.
- *Incident Response, Notification and Remediation*. Any data classified as private for the purpose of data breach regulations should always be encrypted to reduce the consequences of a breach incident.
- *Application Security*. IaaS, PaaS and SaaS create differing trust boundaries for the software development lifecycle, which must be accounted for during the development, testing and production deployment of applications. For IaaS, a key success factor is the presence of trusted virtual machine images. Securing inter-host communications must be the rule, there can be no assumption of a secure channel between hosts, whether existing in a common data center or even on the same hardware platform. Managing and protecting application "secret keys" is critical.
- *Encryption and Key Management*. From a risk management perspective, **unencrypted data existent in the Cloud may be considered "lost" by the customer**. Application providers who are not controlling backend systems should assure that data is encrypted when being stored. Use encryption to separate data holding from data usage. Segregate the key management from the Cloud provider hosting the data, creating a chain of separation.
- *Identity and Access Management*. The key critical success factor when managing identities is to have a robust federated identity management architecture and strategy internal to the organization.
- *Virtualization*. Virtualized operating systems should be augmented by third party security technology to provide layered security controls and reduce dependency on the platform provider alone. Administrative access and control of virtualized operating systems is crucial and should include strong authentication integrated with enterprise identity management, as well as tamper proof logging and integrity monitoring tools.
- *Security as a Service*. Security as a Service (SecaaS) should enable Cloud providers to deliver faster threat protection and better security.

3.3. Our characterization of cloud/edge security

According to the Governance and Operational Domains topics investigated by the CSA, we focus on securing a MOM for Cloud-to-Edge environments. The bullet list below represents the functionalities presented in the CSA guidance [3] that we specifically investigated:

1. In the Governance and Enterprise Risk Management there is the need to *"divulge policies, procedures and processes comprising the Cloud providers' Information Security Management System (ISMS)"*, knowing who makes what.
2. In the Information Management and Data Security it is necessary to *"assure that Cloud provider personnel controls are in place to provide a logical segregation of duties"*.
3. In Data Center Operation, *"Cloud providers must all be able to demonstrate comprehensive compartmentalization of systems, networks, management, provisioning and personnel"*.
4. In the Incident Response, Notification and Remediation, *"Cloud providers should construct a registry of application owners by application interface (URL, SOA service, etc.)"*.
5. In the Traditional Security, Business Continuity and Disaster Recovery, *"Customers should perform onsite inspections of Cloud provider facilities whenever possible"*.
6. In Encryption and Key Management, it is important to *"segregate the key management from the Cloud provider hosting the data, creating a chain of separation"*.

All of the above functionalities can be addressed using a secure MOM, as we will explain in the next Sections.

4. Security model making a MOM for cloud/edge computing compliant with CSA requirements

In this Section, we describe how to make secure the interaction among the distributed software modules of a MOM for Cloud/Edge computing, according to the previously highlighted CSA requirements.

4.1. Preliminary considerations

Let us assume two distributed software modules *A* and *B*. If we adopt a Public Key Infrastructure (PKI), the public key of *A* is identified with PU_A, whereas the private key of A with PR_A (similar notation can be used for module *B*). Often, PKI keys are used

together with symmetric keys identified with K; in our notation, a secret K shared between A and B is recognized by $K_{(A,B)}$. All the keys are used for encrypting Plain text messages (P), which become Ciphered messages (C). The functions $e()$ and $d()$, used respectively for symmetric encryption and decryption, are used in Eqs. (1) and (2):

$$C = e(K_{(A,B)}; P) \tag{1}$$

$$P = d(K_{(A,B)}; C) \tag{2}$$

Algorithms used for performing the symmetric encryption/decryption might be AES [19], Twofish [20], 3DES [21], etc.

The functions $E()$ and $D()$, used respectively for asymmetric encryption/decryption operations, are used in Eqs. (3) and (4) involving only A:

$$C = E(PU_A; P) \tag{3}$$

$$P = D(PR_A; C) \tag{4}$$

The algorithm used for performing the asymmetric encryption/decryption might be RSA [22].

Another important operation used in securing systems is the hashing function $H()$. Given a variable-length message M, the Hash value h is the fixed-length hash achieved as follows:

$$h = H(M) \tag{5}$$

The property of $H()$ is its incontrovertible behavior. Having the h value, it is not possible to re-obtain M. The $H()$ function is defined one-way operation. All Unix based systems store all end-users' passwords as h values into the *passwd* file, thus to prevent any fraudulent use of cleartext passwords. Algorithms used for performing the hashing functions might be MD5 [23], SHA256 [24], HAVAL160 [25], etc.

Combining all the operations and keys described above, it is possible to accomplish many different security tasks for guaranteeing the following needs:

1. data confidentiality;
2. data integrity;
3. data authenticity;
4. data non-repudiation;

For example, using PR_A with the hashing of a transmitted Message M_t, it is possible to accomplish the Digital Signature (DS) of M_t (satisfying items 2 and 4 reported in the list above). This is achieved encrypting the hash of M_t with the PR_A as reported in (6):

$$DS_{M_t} = E(PR_A; H(M_t)) \tag{6}$$

The result of the Signature process DS_{M_t} is attached to the original message for attesting its integrity. The receiver r, when receives the message, compares the calculation of hashing (H(M_r)) against the decryption of DS_{M_t}. Hence *if*:

$$D(PU_A; DS_{M_t}) = H(M_r) \tag{7}$$

then

$$M_t = M_r = M \tag{8}$$

Eq. (8) means that the transmitted message M_t is equal to the received one M_r, and no message corruption occurred. So, we have a common message M exchanged. Implicitly, the success of the result in Eq. (8) also attests the correct identification of the message owner M_t, that is A.

A hybrid usage of asymmetric and symmetric keys allows us to guarantee high degree of security inside the Cloud. This is the typical use of protocols with security capabilities able to leverage SSL/TLS, that is http**S**, imap**S**, ftp**S**.

4.2. Securing the inter-module communication of a MOM for cloud/edge computing

In a complex Cloud to Edge system, we can identify many cooperating actors and software modules that need to exchange data in a secure way. We believe that in the MOMs of the future, all the stakeholders interacting with Clouds and Edge systems should use strong security mechanisms in order to know **Who** is **Doing What** and **When**. Our innovative solution is aimed to provide these capabilities with the following purposes:

- *tracking all the operations inside a Cloud system;*
- *guaranteeing dynamical setup of strong security environments when needed.*

For tracking all operations, we adopt the Digital Signature in conjunction with the Timestamp (T) recording. To setup a strong security communications system we adopt a hybrid model that uses both asymmetric and symmetric keys.

Let us assume a transmitter entity x and a receiver entity y with the modules i and j respectively. The module x_i wants to send the $n-th$ generic command message $CM_{n,y,j}$ to the y_j module. Our architecture has already an internal PKI, hence all the entities have their couple of keys (PU and PR). In particular, the entities can access the *PUs* Directory, for example with a Lightweight Directory Access Protocol (LDAP) service, for getting the PU of each entity compounding our complex system. In order to meet the CSA requirements discussed in Section , the following mechanisms have to be accomplished:

- mechanism (a). All the communications must be tracked. This is possible by enabling x to sign CM_n using the DS technique, and sending the CM_n to y. In addition, x includes the Timestamp (T) within the message. In this way, we can know what and when x requested y. In the same way, y can do the same when it send a response message back to x.
- mechanism (b). Communications between two parties must be cyphered. In this case, the software modules assume the following configuration: x encrypts CM_n using PU_y and the *RSA* algorithm, and sends the CM_n to y. Only y with its PR_y key can see the message (satisfying the requirements 1,2,3 and 4 reported in the list above). x might also include the Timestamp (T) into the message for tracking the date and time.
- mechanism (c). If an entity x wants to communicate in total secrecy with the w, z and y entities, x becomes a *moderator* of the communication and setups the whole security environment. In particular, x has to generate a Random Session Key K_{rx}, (e.g. a symmetric key having 256 bits), and send it to all the parties included in the secret communication, using the technique described in *mechanism (b)*. In this case x sends three different encrypted messages to w, z and y software modules respectively containing the encrypted K_{rx}. The K_{rx} will be used for a while and renewed again by x. All entities during their communication will use the operations highlighted in Eqs. (3) and (4) (satisfying the needs 1,2,3 and 4 reported in the list above). Moreover, in this case, x might also include the Timestamp (T) into the message for tracking the date and time of each interaction.

$$C_{CM} = e(Kr_{(x,z,y,w)}; P_{CM}) \tag{9}$$

$$P_{CM} = d(Kr_{(x,z,y,w)}; C_{CM}) \tag{10}$$

The hybrid security model guarantees better performance and the easiest way for distributing session keys among all the parties. This approach consists in reusing consolidate security technologies useful for increasing the security level inside CoT providers. However, the overhead introduced by these mechanisms in complex CoT systems is unclear. For this reason, in the rest of the paper we will analyze the impact of such a mechanisms by extending a real MOM for Cloud-to-Edge systems, i.e., MOM4Cloud, that is based on one of the major IMPs, i.e., XMPP.

5. MOM for cloud/edge computing architecture

Considering the security requirements discussed in Section 4.2, in this Section, we analyze how a MOM for Cloud/Edge computing can be made secure. The inter-module communication of the MOM enables the communication among the distributed modules of the system, independently of their location and/or role. In order to facilitate our analysis, we consider MOM4Cloud [9], a piece middleware for Cloud-to-Edge systems that is based on a well known IMP such as XMPP. More specifically, starting from the basic MOM4Cloud architecture, we will analyze how to secure it according to the CSA requirements discussed in Section 3.3.

5.1. MOM4Cloud overview

MOM4Cloud belongs to the category of Virtual Infrastructure Manager (VIM) [26] systems. It allows creating Infrastructure as a Service (IaaS) instances. It is able to manage several datacenters and dynamically create, migrate and execute Virtual Machines (VMs) and Containers in multiple Cloud and Edge layers. As depicted in Fig. 1, MOM4Cloud includes several nodes, each one controlled by a management module, that is the Host Manager (HM), and only one node, that is the Cluster Manager (CM), that includes a cluster level management module. A CM is the interface between tenants and software agents running on the HMs. Tenants can be, for example, organization, governments, and society exploiting the Cloud. Each tenant can control the system by means of an administrator or by means of a third party system. In fact, MOM4Cloud provides an Administration Client (AC) that allows controlling the cluster either through a Command Line Interface (CLI) or by means of REST web services. HMs send commands to the CM that gives instructions to other HMs and sends back the results to the AC. The CM also manages Cloud resources and monitors the status of the whole cluster. Into the CM and HMs, several software modules, named *agents* are running, respectively agents of the CM and the HM are named CA (Cluster Agent) and HA Host Agent. Typically, each HA refers to a particular CA. Agents may have one or more plugins that allow them to have specific functionalities. As example, in Fig. 1, HA2, by means of a specific plugin based on jClouds (that we will describe in Section 5.3), is able to manage efficiently both Clouds and Edge Layers.

In order to track the status of the system, a specific CA interacts with the database system, that is SEDNA [27]. MOM4Cloud supports three types of communication:

1. Internal Remote Method Invoker (IRMI);
2. External Remote Method Invoker (ERMI);
3. Notification.

The Internal Remote Method Invoker or "intra-module" communication is the communication at the host level, it is based on the Inter Process Communications, it involves agents of the same module that communicate by means of D-Bus.

The External Remote Method Invoker or "iter-module" communication, allows CAs to communicate with HAs. Indeed, each CA knows all agents in HMs and it can also request for a reply from the HAs. HAs, instead, do not have any knowledge of the CAs. In order to allow communication from HAs to CAs, in MOM4Cloud a Notification system was implemented. HAs do not have to specify the recipient CA, indeed a specific module running inside the CM, the dispatcher, forward requests to the right CA. Communications starting from HAs do not require any reply. Both Notification system and ERMI are based on XMPP, for this reason the MOM4Cloud architecture needs an XMPP Server. MOM4Cloud is based on Ejabberd XMPP server [28] which guarantees fault tolerance in communications and allows system status recovery if a crash of a middleware module occurs.

5.2. XMPP overview

XMPP is an XML-based protocol developed by the Jabber open-source community. It has become very popular due to its flexibility. It provides:

- near-real-time communication;
- extensible instant messaging;
- presence information;
- contact list maintenance.

XMPP works on the HTTP port, thus it allows the server to accept connection in an inter-domain distributed environment even in presence of Firewalls. This feature is very important for MOM4Cloud, because it allows CM and HMs to join/leave the system without any specific configuration. Even though, XMPP uses a client–server architecture, unlike similar systems (such as AOL Instant Messenger or Windows Live Messenger), and the decentralization of communications is achieved interconnecting many instant messaging servers without the need of a central master server.

5.3. Cloud and edge layers management

In this Section, we describe the software agent that allows MOM4Cloud to manage efficiently the Clouds and Edge layers. The **Cloud/Edge Management Agent** belongs to the Host Manager Agent. Indeed, it is necessary an Agent for managing each host. In order to make the maintenance of the whole middleware easy, the Cloud/Edge Management Agent is provided by a specific plugin. The enabling technology of such plugin is jClouds [29]. It supports more than 30 providers, such as Amazon EC2, Rackspace, Microsoft Azure, and Application Program Interfaces (APIs) for the interaction with Virtual Infrastructure Managers (VIMs), such as OpenStack, and Container Engines, such as Docker.

Fig. 2 shows the sequence diagram for deploying resources into the Cloud and Edge Layers. We can observe three different layers:

1. **User** space, that shows the method invoked by the end user by means of the shell.
2. **Cluster Manager** that shows involved modules and method of the Cluster Manager.
3. **Host Manager** that shows involved modules and method of the Host Manager.

The workflow to deploy/migrate resources, that are Virtual Machines and/or Containers, among Cloud and Edge layers is the following:

1. the user invokes the *CreateRes* method from the shell. An XMPP message from the Admin Host is sent to the Cluster Manager;
2. the Virtualization Manager Agent catches the message and invokes the *calculatingUUIDRes* method that, starting from information about the user and resources, creates an UUID that denotes the resource;

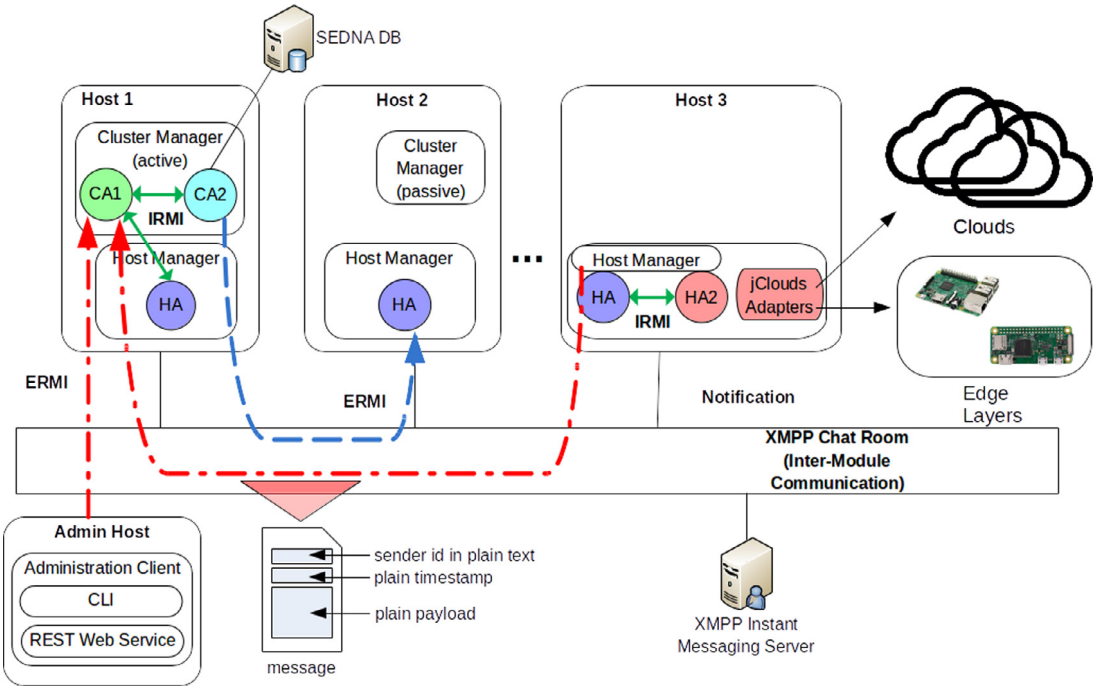


Fig. 1. Distributed organization of MOM4Cloud modules.

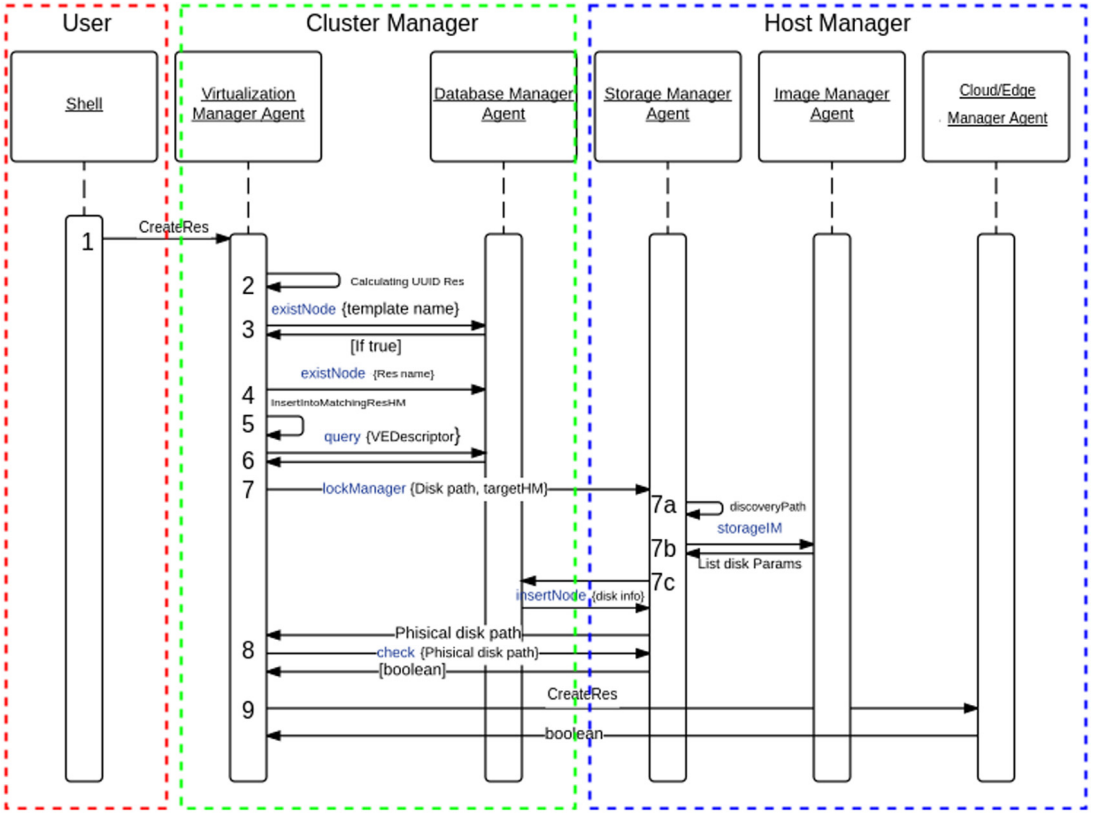


Fig. 2. Distributed organization of MOM4Cloud modules.

3. the Virtualization Manager Agent invokes the *existingNode* method of the Database Manager Agent in order to verify that the template data are stored into the database;
4. then, in case of positive result, the Virtualization Manager Agent invokes the *existingNode* method of the Database Manager Agent in order to verify that the resources data are stored into the database;
5. the Virtualization Manager Agent invokes the *insertIntoMatchingResHM* method that selects the Host Manager in which resources will be deployed;

6. the Virtualization Manager Agent invokes the *query* method of the Database Manager Agent in order to retrieve the information related to the VEDescriptor;
7. the Virtualization Manager Agent invokes the *lockManager* method of the Storage Manager Agent of the target Host Manager;
   (a) the Storage Manager Agent makes a query for the path to the resource by means of the *discoveryPath* method;
   (b) then, invoking the *storageIM* method of the Image Manager Agent, it stores the image into the target HM;
   (c) image information is stored into the db by means of the *insertNode* method of the Database Manager Agent;
8. the Virtualization Manager Agent invokes the *check* method of the Storage Manager Agent in order to verify that the image was correctly updated in the target HM. Finally;
9. the Virtualization Manager Agent invokes the *createRes* method of the Cloud/Edge Manager Agent that, by means of the jClouds plug-in, manages the Cloud or the Edge Layers.

We remark that communications among Client, Cluster Manager and Host Manager are done by XMPP.

6. Securing communications between cloud and edge layers

6.1. Required XMPP security extension

Considering the CSA requirements, the XMPP protocol must to be extended. More specifically, a MOM for Cloud/Edge computing based on this protocol should support the functionalities below:

- **Digital identity management** [30]. Each software module, during the in-band (i.e., an automatic XMPP client module registration on the instant messaging server) registration, asks for a X.509 digital certificate to a Public Certification Authority (PCA) through the Simple Certificate Enrollment Protocol (SCEP). The public key of each module is managed by a Lightweight Directory Access Protocol (LDAP) publisher server. Thus, when a module performs an in-band registration with the XMPP server, it verifies the corresponding digital certificate in the LDAP.
- **Signed message exchange**. Each module potentially can sign messages of other modules.
- **Encrypted message exchange**. Each module potentially can encipher the whole message or some of its parts.
- **Private chat room**. The communication system potentially allows authorized modules to communicate in a separated chat room.
- **Encrypted chat room**. The chat room above described can be encrypted by a "moderator" module through a shared key. Modules, that join the communication, ask the shared key to the "moderator" module, which will send it cyphered by means of their public key. The keys are distributed by means of the PKI.

The aim of this work is to build security functionality on top of the XMPP. XMPP, that supports both the SASL and the TLS technologies, presents several limitations in authentication and encryption: the protocol does not natively support digital signing and data encryption during the interaction between two entities, i.e., people or software modules. In fact, the security layer is demanded to developers because of the XMPP decentralized nature.

The protocol nature, which is flexible and extensible, allows integrating basic security mechanisms, improving the communication security level. The *XEP-0373* specification [7] describes how to include the Open Pretty Good Privacy methodologies (OpenPGP – RFC 4880 – [8]) into Jabber. Specifications for data communications, in terms of cryptographic privacy and authentication, are provided by OpenPGP. Currently, the XEP-0373 does not represent a standard for authentication and data encryption in XMPP communications, but it describes a possible solution. It defines how the information piece inside tags must be processed, by means of XML tags of specific *namespace*. However, the specification does not provide any rule about the public keys exchange. Indeed, this procedure is required to OpenPGP. Although, the chat messaging is related to the human interaction, it is able to be applied to the Cloud Computing systems in which distributed software modules need to interact in a secure real time way.

Given that an XMPP message has an XML encoding, adding an extension means adding a new tag element. In our opinion, in order to make the XMPP secure enough according to the security requirements previously discussed, four basic extensions are required: digital signature, data encryption, session key and timestamp. We briefly describe their features and how their implementation has been accomplished according to the XEP-0373 specification.

- **Digital Signature**. It is based on both a Public Key Infrastructure (PKI), using an asymmetric encryption algorithm, and on the Message Digest algorithm 5 (MD5). It computes the MD5 of the message and signs it by means of the key of the sender. Then, it attaches the MD5 to the message and sends both the message and the MD5 to the receiver. In such a way, the receiver, by means of the public certificate (retrieved by LDAP), can verify the authenticity of the message. The namespace jabber:x:signed identifies the signed extension. The above described process is represented in Fig. 3. In the XEP-0373 specification, the presence message in a chat room can be signed; in such a way a sender can sign its status. Listing 6.1 shows an XMPP message sent from the *HM* to the *CM* both acting in domain *example.org*.

```
<presence from='HM@example.org'
          to='CM@example.org'>
  <status>Online</status>
  <x xmlns='jabber:x:signed'>
      067A1704C780483726677B6C52860B6
      3F3CB87D27316205C72D5E6A680D320
      AFE1990E03C8D64151FBF7009738976
  </x>
</presence>
```

  The receiver (the *CM*), by means of the certificate, can verify the status of the *HM* in the example above.
- **Data Encryption**. Even this process is based on PKI. It provides a method to send confidential information to the receiver attached to the message body that presents TAGs with the namespace "jabber:x:encrypted". In order to send a ciphered message, the sender retrieves the certificate of the receiver from LDAP; then, it uses the certificate to cipher the message. Listing 6.1 shows a ciphered message sent from the *HM* to the *CM*. The receiver, by means of its private key, is able to open and to decipher the received message. Fig. 4 shows the process.

```
<message to='CM@example.org'
         from='HM@example.org'>
  <body>This message is encrypted.</body>
     <x xmlns='jabber:x:encrypted'>
        1496611EB2B6420C0D2F10B71FEF5554d
        AF41F64337240118A18E63EBD9DD44E7P
        dWpR0uQsuJe7+vh3NWn59/gTc5MDlX8dS
        BA69E48CBDBA41BE450615B1CBBB3ECBp
        904D626F378473429FE649724BCA4CF9
        0168F20FD5D2CBC2CFCE08B2880
     </x>
  </body>
</message>
```

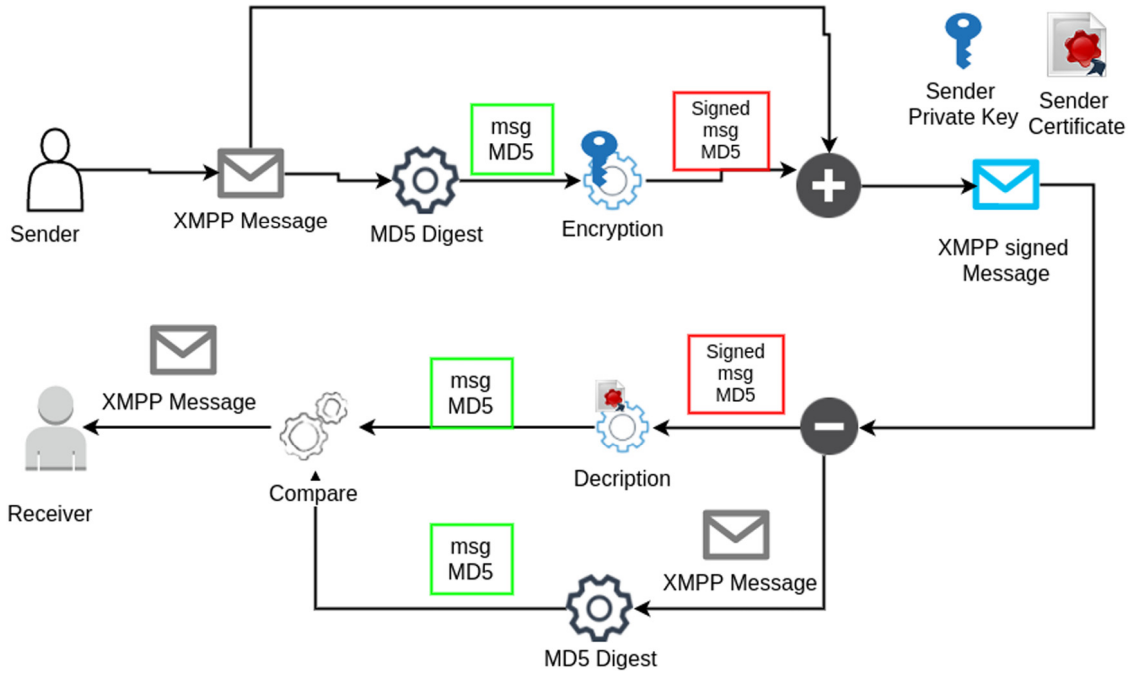


Fig. 3. XMPP message signing.

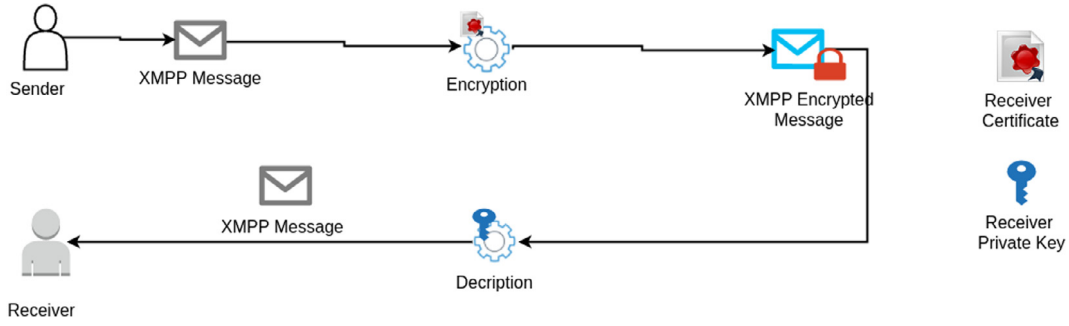


Fig. 4. XMPP message encryption.

Typically, this extension is used to encrypt small pieces of data, e.g., for delivering a session key (also called symmetric key or shared key) to different modules, encrypted with their public keys.

- **Session Key**. Typically, a message payload can hold a considerable data size. For this kind of data, an asymmetric encryption algorithm is faster than a symmetric one. For this reason, to setup the inter-module communication over a chat room, the system uses a hybrid cryptographic scheme. In particular, when different modules have to communicate in a chat room, the module which starts the communication sends the other modules a session key encrypted with their public keys using the encryption extension. Such a session key is set with a given Time To Live (TTL) and is periodically regenerated. Thus, when all the modules obtain the session key, they can encrypt the message using the Session Key extension defined in the message by the XML name space jabber:x:sessionkey (*<x xmlns='jabber:x:sessionkey'>*). SSL/TLS protocols use a similar mechanism. This cryptographic schema presents two advantages: processing times for the encryption/decryption of the message very fast and secure session key distribution. Such a mechanism can be used when CM starts a communication with different HMs.
- Timestamp. Signed MD5 is attached to the message, in such a way, the module that receives the message can verify when it was created and which module has created the message.

6.2. Securing the inter-module communication of MOM4Cloud

The inter-module communication is based on both ERMI and Notification and it enables the interaction among different distributed MOM4Cloud modules via the XMPP. More specifically, it allows:

- CAtoCM communication for sending request to MOM4Cloud
- CMtoHM communication for cluster administration tasks.
- HMtoHM communication for specific task, e.g., a VM migration.

An inter-module communication is triggered

- when a request task is sent by the administration client, the CM, and the HM.
- when an event occurs.
- according to a regular scheduled task, for example for data synchronization.

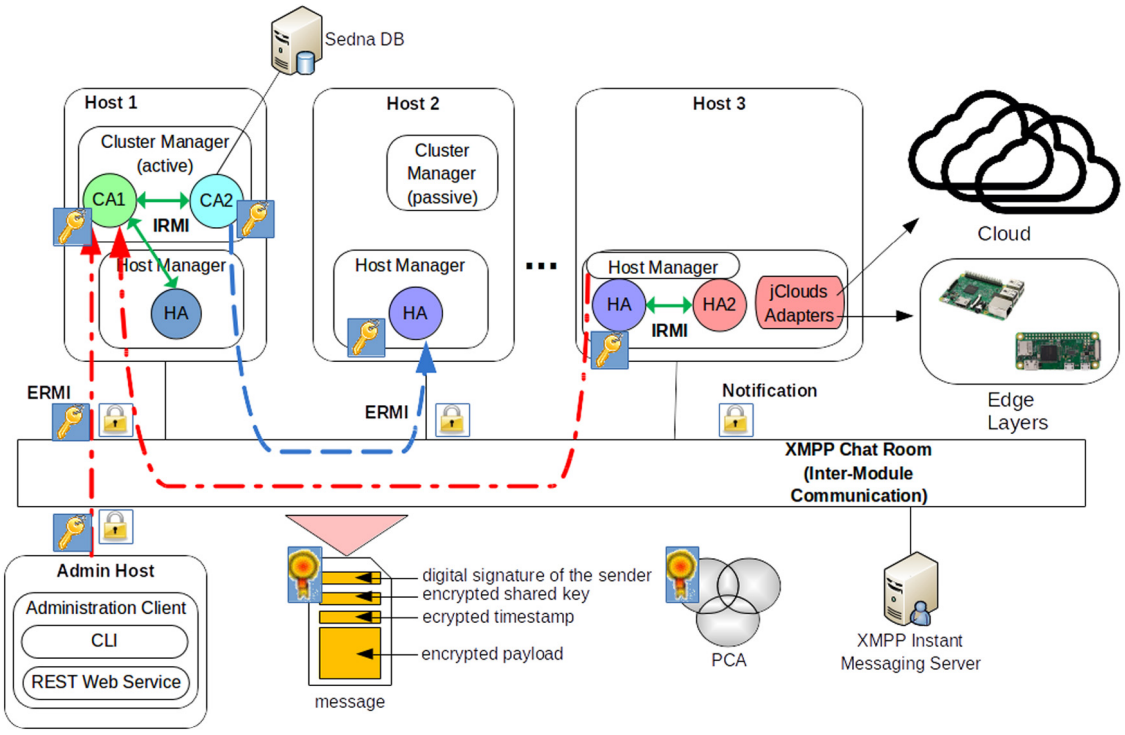


Fig. 5. Distributed organization of MOM4Cloud modules.

In order to secure each MOM4Cloud message by means of the XMPP security extensions, i.e., digital signature, encryption, session key, and timestamp, as shown in Fig. 5 a PCA trusted third party is required.

In the following, we describe how we have extended both the CM and the HM modules in order to include the XMPP extensions. The activity diagram of the CMtoHM without any authentication and encryption is shown in Fig. 6. When one of the aforementioned situations occurs, a MOM4Cloud message is formatted by the CM and sent to a destination HM via XMPP on a specific chat room. At destination, the HM interprets the message, executes the task, and, if a response is required, it formats and sends a response message to the CM

MOM4Cloud, by means of the integration of PKI, SCEP, CA and LDAP, is able to provide a digital identity to each MOM4Cloud element, and to secure the inter-module communication. By means of the SCEP, MOM4Cloud modules are able to obtain their key pair from the PCA. Then, a local KeyStore by each module is created. Each KeyStore contains the certificate in PKCS# format and the private key; KeyStores are protected by password. MOM4Cloud modules publish their certificates on the LDAP server. MOM4Cloud modules in order to establish a secure inter-module communication need the access to LDAP to retrieve certificates and their digital identities. In such a way, modules are able both to sign messages by means of their private key and to cipher the message by means of the certificate of the target module. Receivers, in order to verify the signature, interrogate LDAP for retrieving the public key of the senders. To share a symmetric key, modules use PKI again. The CM-HM inter-module communication in plan activity diagram is shown in Fig. 6, whereas the activity diagram of the version with authentication/encryption is shown in Fig. 7. The authentication/encryption version presents two more steps: the ciphering of the message before sending it, and the deciphering of the message after its reception.

Figs. 8 and 9 highlight the software modules that have been secured in the HM and in the CM respectively with thick borders. As depicted in Fig. 8, a *SCEP Client* has been integrated within the *Initiator* module. It is responsible for digital identity initialization. The *SCEP Client* interacts with the PCA requiring X.509 certificates to perform both authentication and data encryption. This is made by integrating the *Encoder/Decoder* module, i.e., a secure inter-module communication interface that enables data integrity, authenticity, confidentiality, non repudiation. More specifically, this module enables mutual authentication and secure communication between the AC and the active CM, between the CM and the HMs, and among different clusters in a federated inter-domain scenario. Considering HM, the *Encoder/Decoder* module acts as a secure communication interface for the Initiator module and the HC module. In the first case, it allows establishing a secure communication through the Cluster and Utility rooms that are respectively used for basic cluster management and for value-added services. In the second case, the *Encoder/Decoder* module is used for securing the communications of the HC module respectively for the interaction with the CM and for performing a secure VM migration into another HM. As shown in Fig. 9, within the CM, the *Encoder/Decoder* module is used by the Cluster Coordinator module respectively for cluster administration communication, inter-domain communication with HMs, and external cluster communication.

6.3. CSA requirements & MOM4Cloud

In this subsection, we analyze the matching between CSA requirements presented before and MOM4Cloud.

With regard to the possibility to **sign exchanged messages**, by means of **Timestamps** and valid **Signatures**, it is possible to guarantee data integrity and data non-repudiation. According to item 2, segregation of duties is possible considering **chat rooms protected by password**. Only authorized modules can access the communication system enabling them to perform given tasks, e.g., only authorized administrators can access the Cluster room, only authorized HM can access a Utility room, and so on. Protected chat rooms allow carrying out a comprehensive compartmentalization of systems and management according to item 3. Onsite

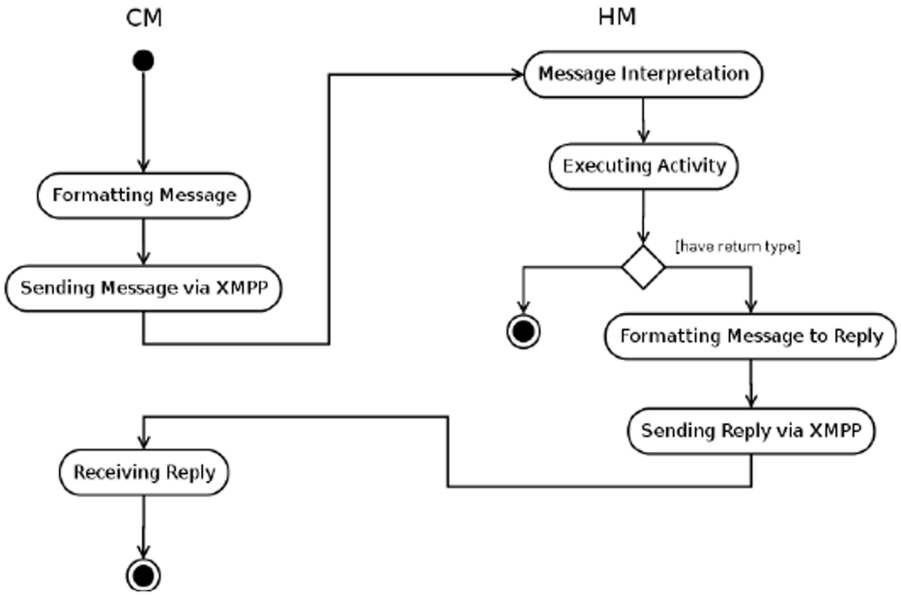


Fig. 6. Plain inter-module communication.

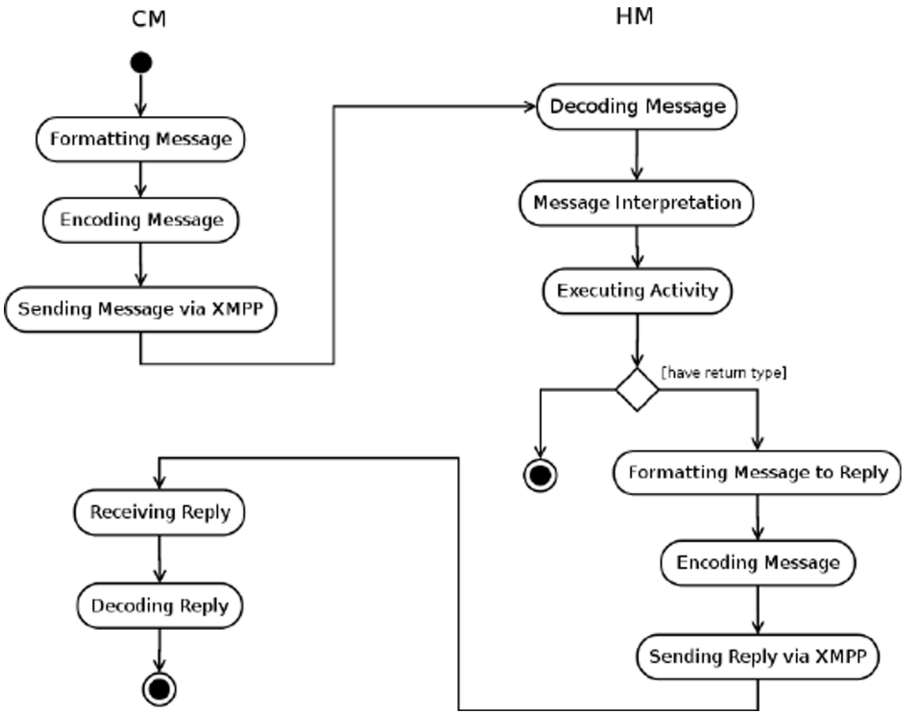


Fig. 7. Encrypted inter-module communication.

inspections from customers is guaranteed from **Sign exchanged messages** and **Timestamps**. The issue reported in bullet 5 is solved by means of the SEDNA database and logs of XMPP.

7. Implementation

Hereby, we provide several implementation highlights related to the development of the XMPP security extensions in CLEVER, that is an implementation of the MOM4Cloud architectural model. In order to develop the Java code, the following tools have been adopted.

- Java API able to sign and validate XML documents by means of **XML Digital Signatures (XMLDsig)** [31];
- Java APIs for the administration of a CA by means of **EJBCA Web Service** [32];
- Public Key Infrastructure (PKI) by means of **Enterprise Java Bean Certification Authority (EJBCA)** [32];
- Simple Certificate Enrollment Protocol (SCEP) by means of **Java Simple Certificate Enrollment Protocol (JSCEP)** [33];
- Lightweight Directory Access Protocol (LDAP) by means of **OpenLDAP** [34];
- Java APIs for LDAP by means of **Java Naming and Directory Interface (JNDI)** [35];

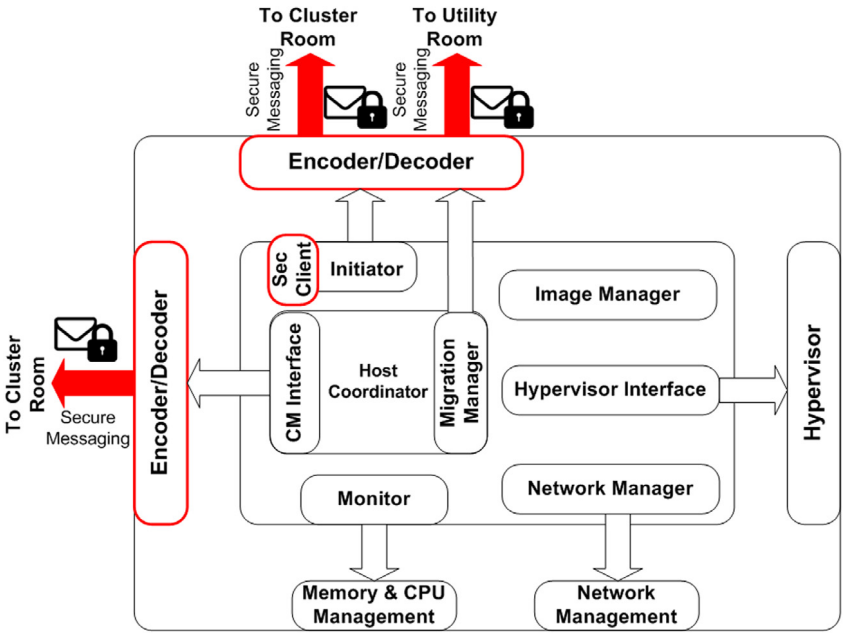


Fig. 8. Security integration in HM.

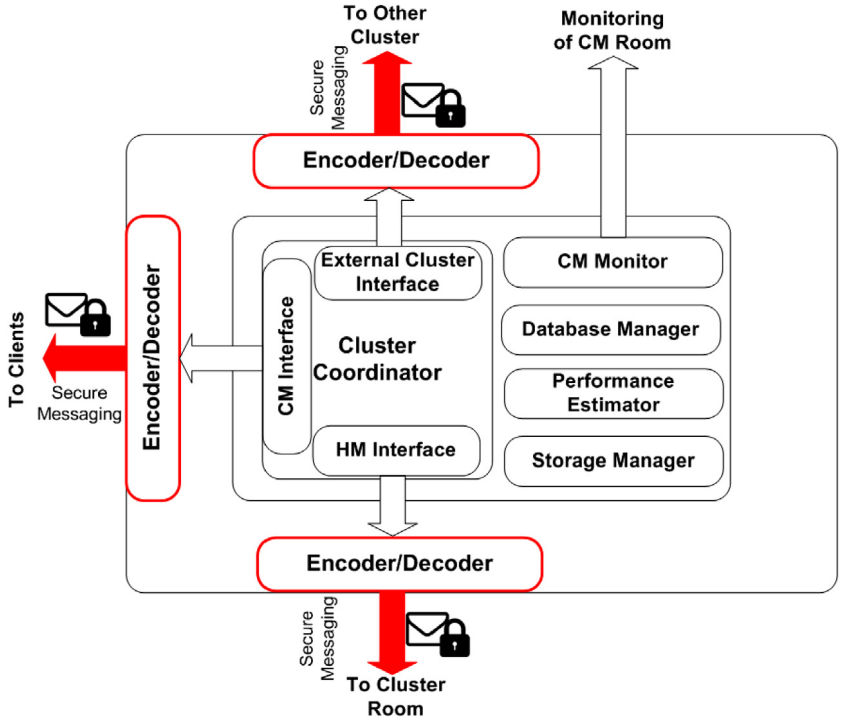


Fig. 9. Security integration in CM.

- Java cryptographic algorithms library by means of **Bouncy Castle** [36]. In our tests we adopted the following algorithms, according to the required functionalities:
  - Data encryption: AES, that is a symmetric-key algorithm based on the Rijndael cipher;
  - Signing: RSA-1024, that is an algorithm for public-key cryptography with keys of 1024 bits;
  - Digest generation: SHA-1, that is a cryptographic hash function able to produce a 160-bit message digest.

7.1. XMPP security extension design

The security extensions were developed through several Java classes implementing the *Smack PacketExtension*: *SecureExtension* and *Provider*. The former allows the definition of XMPP security extensions, while the latter allows the receiver to correctly process the different XMPP security extensions.

The *SecureExtension* class allows defining security extensions to MPP. It involves two attributes: *type* and *data*. The first indicates the type of extension and the corresponding namespace,

whereas the second represents the extension of the content. The Provider class performs the parsing of the XML messages at the receiver. In order to allow a correct parsing of the encrypted data, digital signature, and session keys, it has been further specialized in three classes: *EncryptedProvider*, *SignedProvider*, *SessionKeyProvider*. Each of these three specialized classes overrides the *parseExtension* method of the *SecureProvider* super class, that returns a PacketExtension object.

In order to integrate the security extensions in the XMPP, we developed the *ConnectorThread* class implementing the *Runnable* interface, representing an XMPP client. It registers the packet Listener and three Providers for the management of the security extensions respectively represented by the *EncryptedProvider*, *SignedProvider*, and *SessionKeyProvider* classes. The *ConnectorThread* allows an XMPP client to establish a connection with the XMPP server, register an account, setup the digital identity of the client. As soon as the client establishes the connection with the XMPP server, the SCEP client is called and it sends an *enrollment* request to the trusted PCA, that returns an X.509 digital certificate. After that, the XMPP client creates a *KeyStore* local object of *Map* type containing a private/public key pair. In order to improve the security of the whole system a password will protect the key pair. Once the client obtains its credentials, it waits for incoming requests. The XMPP client is able to create/join a chat room and send clear, encrypted, or signed messages on the chat room. The signed and encrypted messages are formatted using the methods of the *SecureExtension* class. Its main methods are:

- **encryptedMessage()**. It allows sending encrypted messages using the public key of the receiver XMPP client.
- **signedMessages()**. It allows sending signed messages by means of an attached signed message digest.
- **groupChatEncryptedMessage()**. It allows sending encrypted messages on a given chat room.

In order to manage the incoming XMPP messages, three Listener classes have been developed extending the PacketListener super class. Each specialized Listener class overrides the processPacket() method. The three Listener classes are:

- *MessagePacketListener*. It manages clear and encrypted messages.
- *PresenceListener*. It manages the presence messages.
- *GroupChatPacketListener*. It manages the presence messages in an encrypted chat room.

7.2. Integration of the XMPP security extensions in CLEVER

Hereby, we discuss how the previously discussed XMPP security extensions have been integrated in both CM, HM, and AC, in order to enable secure the inter-module communication. The integration of the security extension in CLEVER involved three main classes: *ConnectionXMPP*, *ClusterManagerAdministrationTools*, *RoomListener*, and *CleverChatListener*. The corresponding class diagrams are respectively depicted in Figs. 10–13; *ConnectionXMPP* is the class that manages the inter-module communication of CLEVER via the XMPP. It was enhanced with three attributes: *sessionKey*, for the management of the session key; *ldapClient* for requesting the LDAP server the digital certificate of the remote entity (e.g., the HM) involved in the secure inter-module communication. The access to the LDAP server is further protected by password. The *X509Utils* class supports the required cryptographic algorithms, allowing data encryption and digital signature.

ClusterManagerAdministrationTools is the class implementing several tools supporting the CLEVER administration command line interface client. In this class, we overloaded the *execAdminCommand()* and *execSygnAdminCommand* in order to support respectively the encryption and signature of the messages sent from the CLEVER administrator to the CM.

Each CLEVER entity manages the XMPP messages coming from other entities by means of two main classes, each implementing the *MessageListener* and the *PacketListener* interfaces. The first class is *RoomListener* that allows the CM to wait for messages coming from the administration client and other CMs on an administration chat room. The second class is *CleverChatListener* that allows both CMtoHM and HMtoHM communications on another chat room. As the secure extension integration is the same for both classes, in the following we will focus only on the first. The constructor of the *RoomListener* class requires the *CleverMessageHandler* object representing the handler of a received message, and the *ConnectionXMPP* object representing the XMPP connection on which the listener is connected waiting for messages. By means of the *get()* method of *ConnectionXMPP*, the listener can access the *ldapClient* and the *keyStore* object containing the session key of the entities joined to the chat. In order to allow the encrypted communication, the *processMessage()* method has been modified. Through the getExtension() method of the *Message* class it is possible to read the XMPP message extension. If the *SessionKey* extension is present, a message decryption is performed by the processing entity using its own private key. The decryption task is performed by means of the *decryptToString()* method to the *X509Utils* class provided by the Bouncy Castle project. After that, the decrypted session key is associated to the identifier of the sender and it is stored in the *sessionKey* object. If the *Encrypted* extension is present, it is needed to get the session key of the sender and to decrypt the message by means of the *sEnctypt()* method of the *X509Utils* class.

8. Experimental assessment and analysis

In this section, we perform scalability and efficiency tests in order to evaluate the overhead due to security mechanisms.

8.1. System settings

In our tests, we considered three nodes connected by means of a GB-LAN, Fig. 14 shows the test environment.

In our experiments, the nodes are 3 blade servers IBM X3630M3, characterized by the following hardware configuration: CPU Intel(R) Core(TM) i7-4790 K CPU @ 4.0 GHz with 4 cores and 4 threads, RAM 16GB, GFLOPS 181, OS: Ubuntu server 16.04 LTS 64 BIT.

From a logical point of view, a node works as AC at the Cloud layer, that is responsible to connect several tenants (e.g., administrators or third party software systems) to CLEVER.

We deployed CM and HM in the other two nodes that act respectively at the Cloud and Edge layers. Fig. 14 also shows the steps that different nodes execute during the time, that are described below:

- Step 1: the tenant sends to the AC the *listvms* command that queries all registered VM in the specific HM;
- Step 2: AC sends the corresponding XMPP message formatted following the CLEVER specifications to the CM;
- Step 3: CM sends the request to the specific HM;
- Step 4: HM processes the request and replies to the CM;
- Step 5: CM sends the answer to the AC;
- Step 6: AC displays information about VMs to the tenants.

In order to simulate concurrent requests for Cloud services in our experiments, a thread for each tenant in the AC node was instantiated. Each thread sends a command at a random time. In such a way a lot of messages in a short time interval are generated, and this causes high overhead in the AC for a consistent number

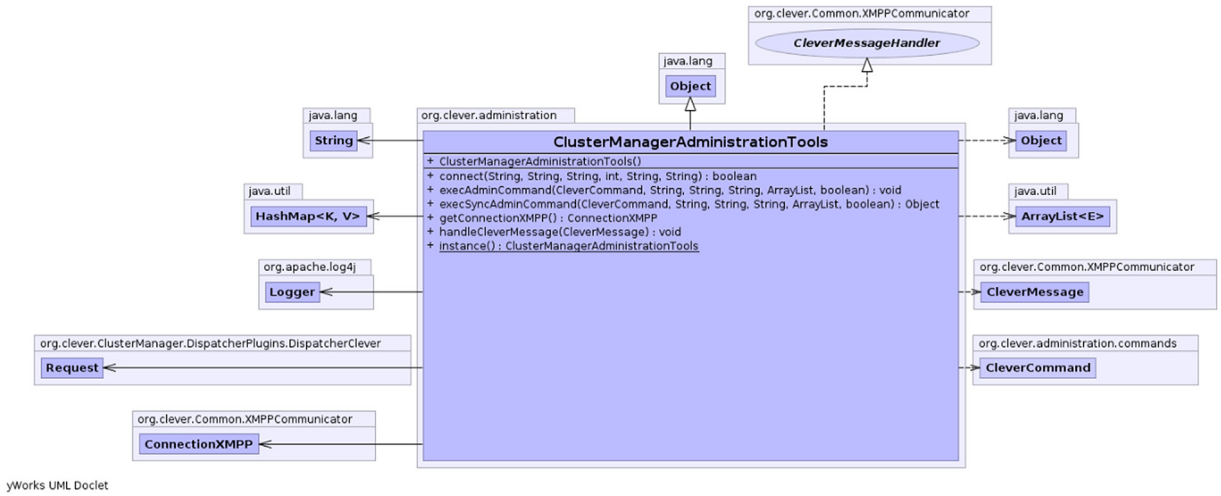


Fig. 10. Class diagram of the clustermanagerAdministrationTools class.

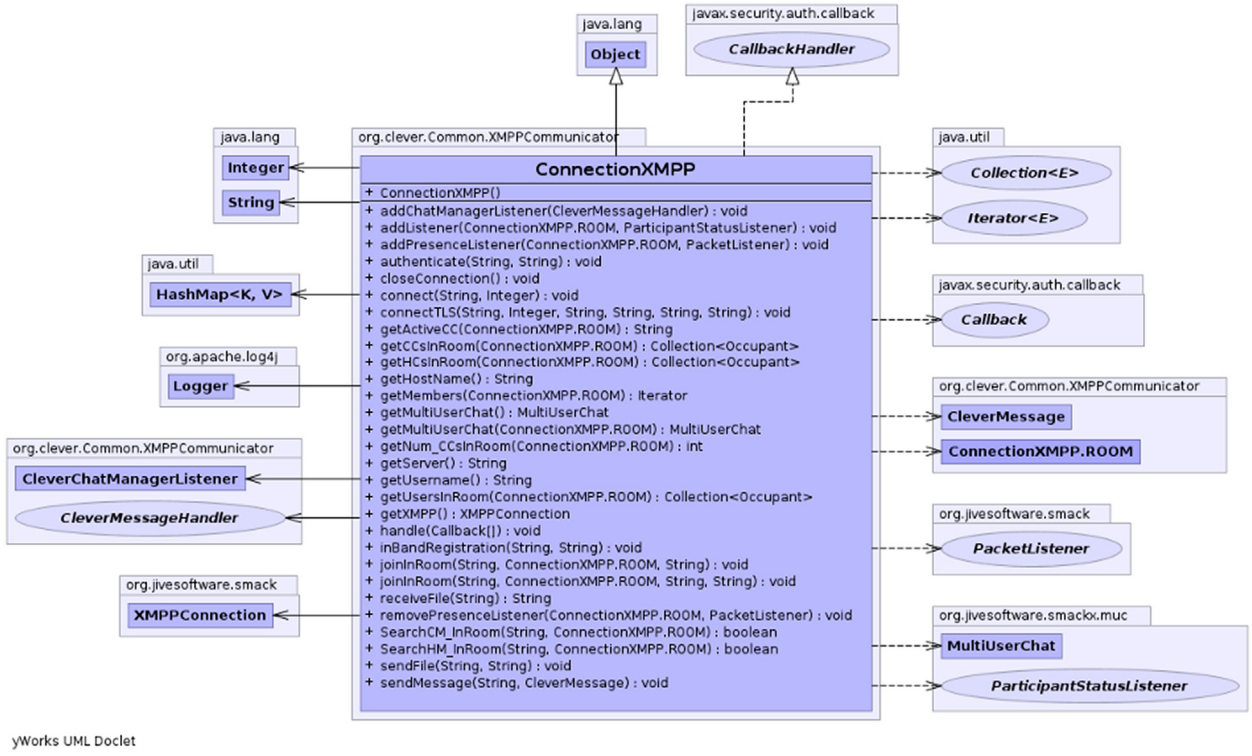


Fig. 11. Class diagram of the ConnectionXMPP class.

of tenants. In particular, we considered a range of 1–40 threads that send commands in a time interval of 1 s. This scenario well describes typical service providers (e.g., web portals) that works as collectors of tenant queries.

For the evaluation analysis we consider the following metrics:

- Transmission Time (T_{tran}:, the time spent to send a formatted message from a sender to a receiver. It provides information on the overloading of the system, buffers fillings and nodes congestion.
- Elaboration Time (T_{elab}), that measures the complexity of message management and provides the overhead due to data encryption and signing. We separately deal with the processing time in message transmission T_{elab_TX} and in message receiving T_{elab_RX}.

In the following, we provide the experimental results in terms of the above metrics assuming request/response messages. More specifically, we analyzed AC to CM and CM to HM communications for the request and HM to CM and CM to AC communications for the response. We performed 50 subsequent experiments in order to consider confidence intervals at 95% and average values.

8.2. Transmission time

We evaluate the overhead introduced by the security extensions in terms of delay for transferring XMPP messages. As we described previously, the testbed machines are interconnected by means of a GB LAN, for this reason we can assume that the network latency is negligible. In such a way, our result can be used in order to evaluate a scenario with a certain network latency.

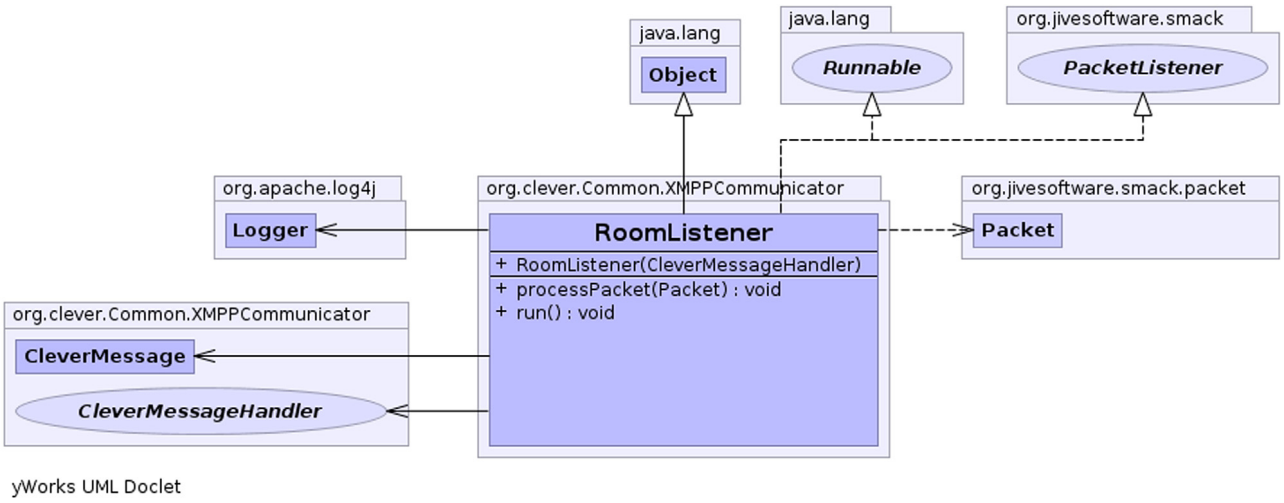


Fig. 12. Class diagram of the RoomListener class.

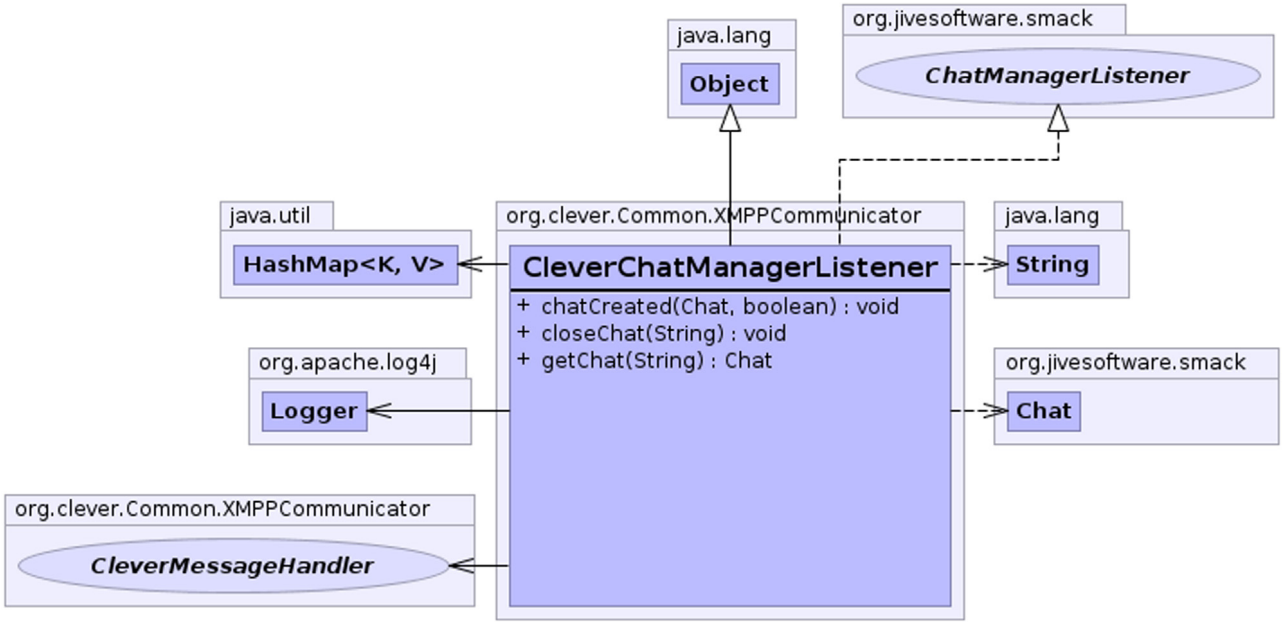


Fig. 13. Class diagram of the CleverChatManagerListener class.

Fig. 15(a) presents message exchanges between AC and CM. We remark that we have different threads in the AC that perform their queries in parallel. T_{tran} increases with the tenant number. Indeed, increasing the tenant number, a higher number of requests sent in a small time interval is stored in the transmission buffer. T_{tran} in the secure scenario is greater than in the plain one, due to the introduction of security mechanisms. Indeed the ciphering of data introduces a query to LDAP in order to retrieve the certificate. Thus, transmission buffers hold both data messages and requests for the LDAP server, resulting in a higher transmission delay. Transmission delay in the encrypted scenario increases with the number of the tenants from 0.16 to 2.56 s in average. Unfortunately, this behavior is not constant. In some cases, when the message size is bigger, it can increase more than one hundred times (from 0.10 to 10.09 s). SHA-1 produces a hash 160 bits long for any length of a generic message m. Furthermore, the RSA algorithm includes one block more of 128 bytes. The total message digest length is 148 bytes. In our experiments, we considered very small messages (about 1–2 KBytes), thus the digital signature increases the data size of about 10%, causing a fast filling of the buffers. Regarding the CM, the system behavior is a bit different, as shown in Fig. 15(b). In fact, unlike the AC, the CM manages messages go back, so that the message insertion rate in the buffers is slower than in the AC buffers. Specifically, T_{tran} measurements for uncoded messages are almost constant. T_{tran} curves for secure messaging are smoother than in Fig. 15(a), with a maximum increase of about 65% for encrypted messages and 270% for signed messages. The average T_{tran} for the CM (13.61 s) is higher than for the AC (0.97 s) since the CM is involved in communications also with the HM, thus its transmission suffers from two concurrent data flows. Such a stressed transmission overhead is proved by the results shown in Fig. 16.

Fig. 16(a) shows that the mean T_{tran} measured at the CM is 1345 s, which is similar to the T_{tran} observed in transmissions from the CM to the AC, thus proving that transmissions in the CM are fairly divided between the two communication flows. On the contrary, the average T_{tran} measured at the HM and shown in Fig. 16(b) is very low, since the HM is involved only in transmissions with

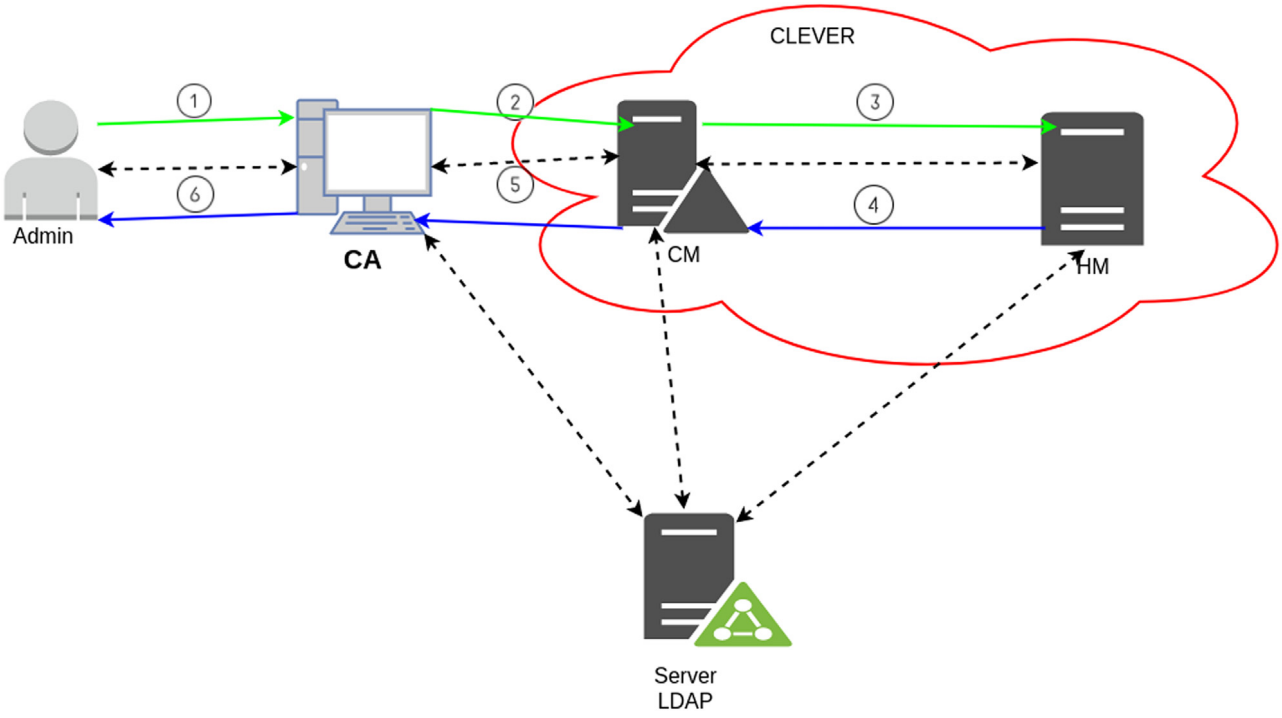


Fig. 14. Testbed configuration.

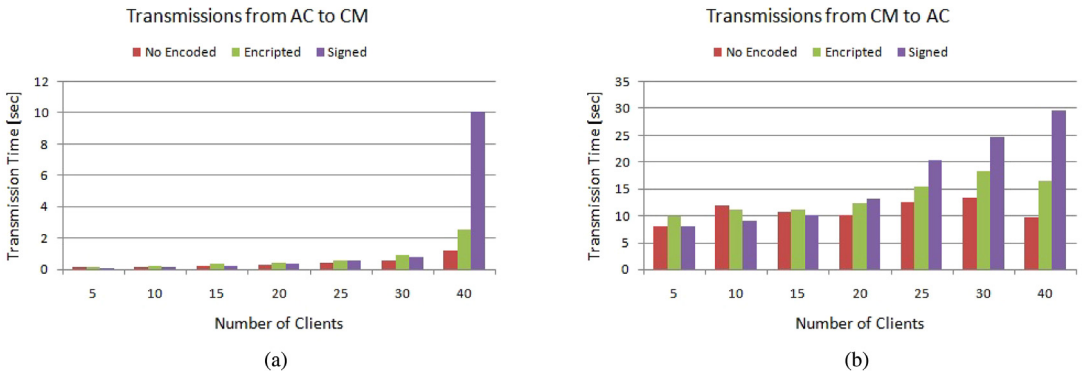


Fig. 15. Transmission time between AC and CM.

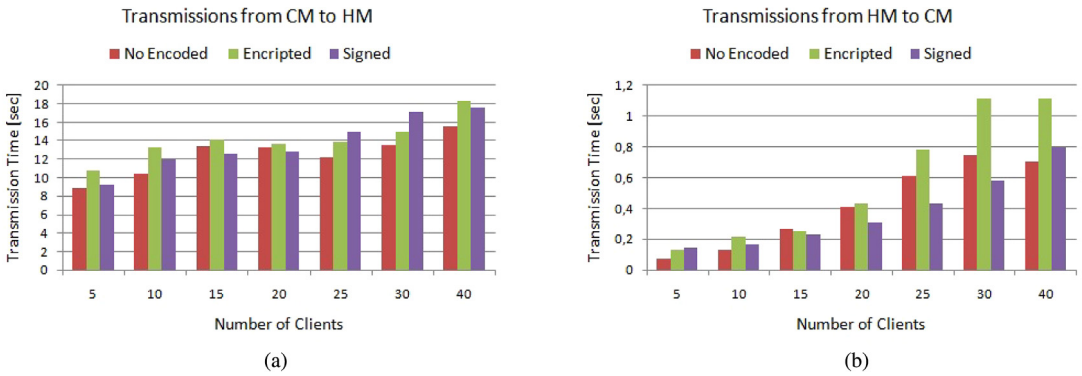
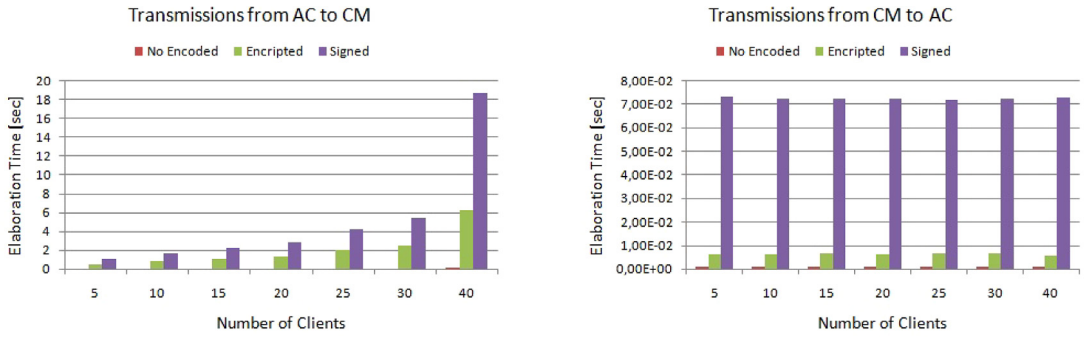


Fig. 16. Transmission time between CM and HM.



(a) (b)

Fig. 17. Elaboration time in transmission.

the CM. It is also lower than the average T_{tran} measured at the AC, since the HM elaborates one message at time, thus reducing transmission delay.

8.3. Elaboration time

$T_{elab-TX}$ evaluates the time all the tasks spend to manipulate a message before sending it to the destination. It includes:

- symmetric key management and message coding for data encryption;
- coding of message digest for data signature.

As shown in Fig. 17(a), $T_{elab-TX}$ is comparable to T_{tran} when measured at the AC. Otherwise, it is much lower than T_{tran}. This is the effect of the specific request generation mechanism implemented in the testbed, where concurrent threads work in the AC. Thus, the elaboration and the transmission tasks influence the effective delays measured at the AC node. This system behavior explains why $T_{elab-TX}$ increases by increasing the number of tenants. On the contrary, the progressive management of messages in the CM decouples the elaboration tasks from the transmission ones; the measured $T_{elab-TX}$ is not affected by node overloading and it remains constant, despite of the increasing in the number of tenants, as shown in Fig. 18(b). From the tenant point of view, the experienced $T_{elab-TX}$ is pretty much the delay due to activities being executed in the AC. The results shown in Figs. 17 and 18 prove that the message signing activity strongly impacts the elaboration load. This overhead is introduced by the asymmetric encryption. Indeed it is heavier than the symmetric.

Analyzing the results related to the elaboration time in the CM and HM shown in Fig. 18, we notice that $T_{elab-TX}$ for encrypted messages is higher when the communication occurs from CM to the HM (on average it is 10 msec) than from HM to CM (on average it is 7 ms). This result is due to the management of the symmetric keys mainly affecting the communications from the CM to the HM. To send a message to the HM, the CM has to share a symmetric key to encrypt the message. On the contrary, the HM sends messages to the CM using the same key previously shared. The transmission of signed messages is characterized by similar values of $T_{elab-TX}$ as shown in Fig. 18(a) and (b) (around 65 msec), since message signing is carried out only using the private key of the transmitter.

$T_{elab-RX}$ evaluates the time to execute for the tasks necessary to decode a message whenever it arrives at the destination. It includes:

- message decoding using the shared key for data encryption;
- recovery of the source's public key and decoding of message digest for data signing.

Fig. 19(a) shows that at the AC the time spent to elaborate received messages is drastically lower than the time experienced in transmission. This is a consequence of the serial message management of the CM, which implies that, usually, one tenant at a time receives a message and is involved in effective elaboration. Thus $T_{elab-RX}$ is not influenced by the overload of the node. The decoding of encrypted data is almost independent of the number of tenants, since it is carried out by using the already shared key. On the contrary, to decode signed messages, each tenant in AC has to recover the public key of the CM through a query to the LDAP server, so that $T_{elab-RX}$ raises with the number of tenants since the number of connections to the LDAP server grows. As depicted in Fig. 19(b), we can provide similar considerations with regard to $T_{elab-RX}$. When the CM receives a signed message, it has to retrieve the public transmitter key for the tenant sending the message and causes an overload of the CM with the number of tenants.

$T_{elab-RX}$ between the CM and the HM is plotted in Fig. 20. The decoding of encrypted messages using the shared key requires an elaboration time comparable to the time for elaborating uncoded messages (about 0.2 msec). With reference to signed messages in Fig. 20(a), $T_{elab-RX}$ is almost constant with the number of tenants since the CM makes just a query to the LDAP server for elaborating the first message of the HM. Then it uses the same key for all the other messages originated from the HM (even if they derive from different tenant requests). The same behavior characterizes the signed messages that are received and elaborated at the HM. As shown in Fig. 20(b), similar results can be appreciated considering $T_{elab-RX}$ during the transmission from HM to CM.

Encryption activities introduced in order to provide security features such as confidentiality, authenticity and non-repudiation of data, proposed in this scientific work cause delays. However, considering all the management activities of the Cloud middleware such delays are still acceptable.

9. Conclusions and remarks

With the advent of the CoT paradigm, needs of managing VMs and containers in both Cloud and Edge layers in a secure and flexible fashion became very compelling. In this paper, we investigated several of the major concerns described by the Cloud Security Alliance (CSA) guidance, i.e., (1) Governance and Enterprise Risk Management; (2) Information Management and Data Security; (3) Data Center Operation; (4) Incident Response, Notification and Remediation; (5) Traditional Security, Business Continuity and Disaster Recovery; (6) Encryption and Key Management.

In order to study the process required to enable Cloud/Edge MOM to address the aforementioned concerns, we extended security features in CLEVER, that is one of the reference implementations of the MOM4Cloud architectural model. More specifically,

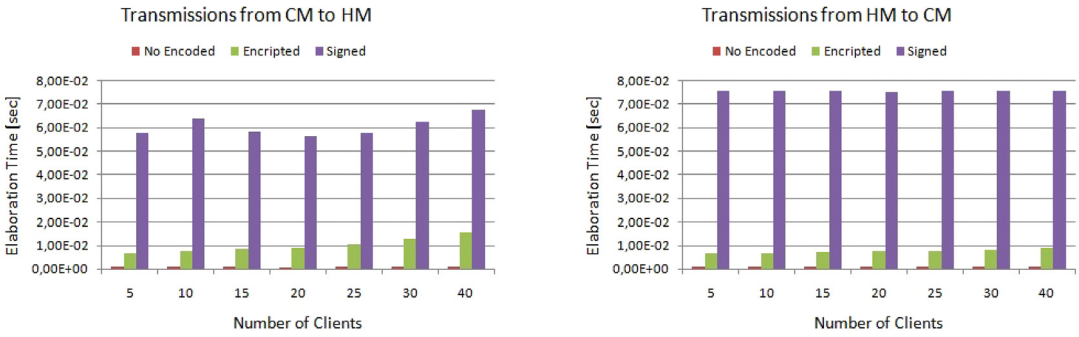


Fig. 18. Elaboration time in transmission.

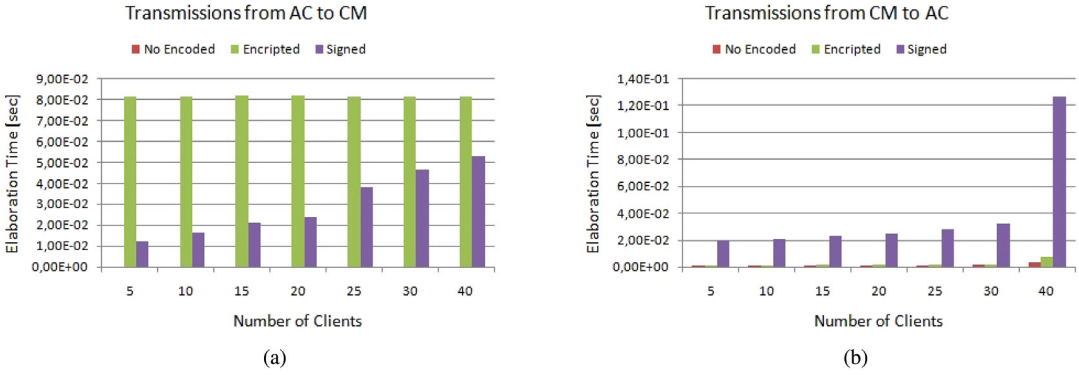


Fig. 19. Elaboration time in receiving.

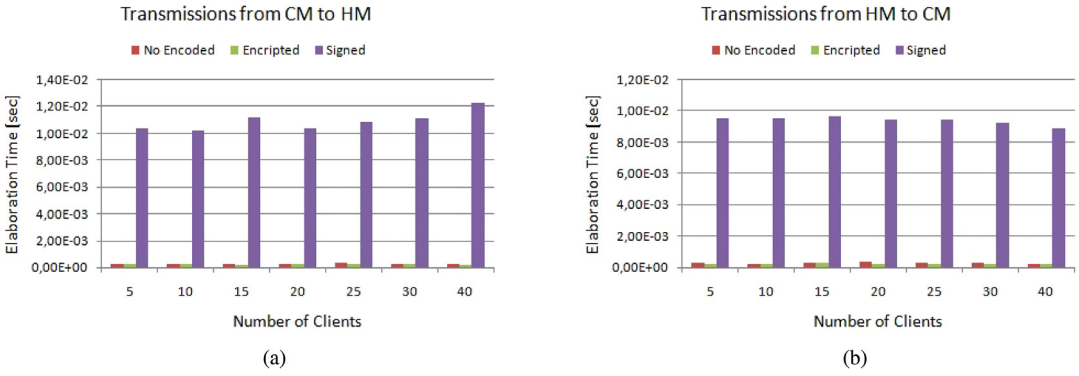


Fig. 20. Elaboration time in receiving.

we highlight the involved issues and discussed how to make its communication system secure.

Different modules of the CLEVER middleware communicate each other using XMPP. Even though XMPP presents several interesting capabilities in the context of Cloud computing, it does not support the security features required to address the concerns described in the CSA guidance. Thence, considering the CLEVER case of study, we specifically focused on how to make its XMPP communication system secure in order to meet the security requirements of Cloud/Edge environment for the management of CoT services deployed in virtualized environments.

The performance analysis on the developed security extensions shows the effectiveness of the proposed approach. The proposed work can be considered a landmark for software architects who want to make their Cloud/Edge systems compliant with the CSA guidance. For future works we plan to move toward the innovative Osmotic Computing paradigm [37] to manage IoT applications and services.

References

[1] G. Fortino, R. Gravina, W. Russo, C. Savaglio, Modeling and simulating internet-of-things systems: a hybrid agent-oriented approach, Comput. Sci. Eng. 19 (5) (2017) 68–76.

[2] Cloud-based activity-aaservice cyber-physical framework for human activity monitoring in mobility, Future Gener. Comput. Syst. 75 (2017) 158–171.

[3] Cloud Security Alliance. Security guidance for critical areas of focus in cloud computing, v3.0 https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf.

[4] A. Celesti, M. Fazio, M. Villari, SE CLEVER: A secure message oriented Middleware for Cloud federation, in: Proceedings - International Symposium on Computers and Communications, 2013, pp. 35–40.

[5] A. Celesti, M. Fazio, M. Villari, Enabling secure XMPP communications in federated IoT clouds through XEP 0027 and SAML/SASL SSO, Sensors (Switzerland) 17 (2) (2017) 1–21.
[6] Extensible messaging and presence protocol (XMPP), http://xmpp.org/.
[7] XEP-0373: Current Jabber OpenPGP Usage, 2016, https://xmpp.org/extensions/attic/xep-0373-0.1.2.html.
[8] OpenPGP Message Format, 2007, http://www.rfc-editor.org/info/rfc4880.
[9] M. Fazio, A. Celesti, A. Puliafito, M. Villari, A message oriented middleware for cloud computing to improve efficiency in risk management systems, Scalable Comput.: Pract. Exp. 14 (4) (2013) 201–213.
[10] A. Celesti, F. Tusa, M. Villari, A. Puliafito, Integration of CLEVER clouds with third party software systems through a REST web service interface, in: Proceedings - IEEE Symposium on Computers and Communications, 2012, pp. 827–832.
[11] S. Shirazi, A. Gouglidis, A. Farshad, D. Hutchison, The extended cloud: review and analysis of mobile edge computing and fog from a security and resilience perspective, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2586–2595.
[12] C. Esposito, A. Castiglione, F. Pop, K.K. Choo, Challenges of connecting edge and cloud computing: a security and forensic perspective, IEEE Cloud Comput. 4 (2) (2017) 13–17.
[13] S. Rathore, P. Sharma, A. Sangaiah, J. Park, A hesitant fuzzy based security approach for fog and mobile-edge computing, IEEE Access 6 (2017) 688–701.
[14] G. Li, H. Zhou, B. Feng, G. Li, T. Li, Q. Xu, W. Quan, Fuzzy theory based security service chaining for sustainable mobile-edge computing, Mob. Inf. Syst. 2017 (2017) 1–13.
[15] W. Abdul, Z. Ali, S. Ghouzali, B. Alfawaz, G. Muhammad, M. Hossain, Biometric security through visual encryption for fog edge computing, IEEE Access 5 (2017) 5531–5538.
[16] V. Vassilakis, I. Chochliouros, A. Spiliopoulou, E. Sfakianakis, M. Belesioti, N. Bompetsis, M. Wilson, C. Turyagyenda, A. Dardamanis, Security analysis of mobile edge computing in virtualized small cell networks, IFIP Adv. Inf. Commun. Technol. 475 (2016) 653–665.
[17] V. Vassilakis, E. Panaousis, H. Mouratidis, Security challenges of small cell as a service in virtualized mobile edge computing environments, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) vol. 9895, LNCS, 2016 pp.70–84.
[18] Cloud security alliance, http://www.cloudsecurityalliance.org/.
[19] AES https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development.
[20] Twofish web page, with full specifications, free source code, and other twofish resources, https://www.schneier.com/academic/twofish/.
[21] Triple DES encryption, https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.csfb400/gloss.htm#T.
[22] The Original RSA Patent as filed with the U.S. Patent Office by Rivest; Ronald L. (Belmont, MA), Shamir; Adi (Cambridge, MA), Adleman; Leonard M. (Arlington, MA), December 14, 1977, https://patents.google.com/patent/US4405829.
[23] RFC 1321, section 3.4, "Step 4. Process Message in 16-Word Blocks", page 5, https://tools.ietf.org/html/rfc1321.
[24] US secure hash algorithms (sha and hmac-sha), https://tools.ietf.org/html/rfc4634.
[25] Haval (the official haval page with the research paper on haval, the latest c source code and haval oids) https://web.archive.org/web/20150111210116/http://labs.calyptix.com/haval.php.
[26] B. Sotomayor, R. Montero, I. Llorente, I. Foster, Virtual Infrastructure Management in Private and Hybrid Clouds, IEEE Internet Comput. 13 (5) (2009) 14–22.
[27] Native Database XML System, http://www.sedna.org.
[28] Ejabberd, the Erlang Jabber/XMPP daemon, http://www.ejabberd.im/ Jan 2012.
[29] jClouds, "The Java Multi-Cloud Toolkit", https://jclouds.apache.org/.
[30] E. Bertino, F. Paci, R. Ferrini, N. Shang, Privacy-preserving digital identity management for cloud computing, Computer 32 (1) (2009) 21–27.
[31] XML digital signature API (XMLDsig), http://docs.oracle.com/javase/7/docs/technotes/guides/security/xmldsig/XMLDigitalSignature.html.
[32] EJBCA Enterprise PKI CA, http://www.ejbca.org.
[33] Java simple certificate enrollment protocol (JSCEP), https://code.google.com/p/jscep.
[34] OpenLDAP, http://www.openldap.org.
[35] Java naming and directory interface (JNDI), http://www.oracle.com/technetwork/java/jndi/index.html..
[36] The legion of the bouncy castle, http://www.bouncycastle.org.
[37] M. Villari, M. Fazio, S. Dustdar, O. Rana, R. Ranjan, Osmotic computing: a new paradigm for edge/cloud integration, IEEE Cloud Comput. 3 (6) (2016) 76–83.

Antonio Celesti received the Ph.D. in "Advanced Technology for Information Engineering" in 2012 at the University of Messina (Italy). From 2008 e is one of the members of the Mobile and Distributed Systems Laboratory (MDSLab) in Messina. He worked as collaborator in several International projects including EU FP7 RESERVOIR, EU FP7 VISION CLOUD, EU FP7 frontierCities and EU Horizon 2020 BEACON.

From December 2015 is technical-scientific fellow at the Scientific Research Organizative Unit, University of Messina. He is currently Adjunct Professor of Electronic and Computer Science Bioengineering and database II. His main research interests includes distributed systems, Cloud computing and IoT service federation and security.

Maria Fazio has been involved in many national and international projects, including the EU FP7 CloudWave Project (2013–2016) and EU Horizon 2020 BEACON (2015–17). She is Co-Editor-In-Chief of the EAI Endorsed Transactions on Cloud Systems, and member of the Editorial Board of international journals (e.g., the International Journal of Business Data Communications and Networking (IGI Publishing) and the EAI Endorsed Transactions on Smart Cities). She has been also editor of Special Issues in international journals (e.g., IGI Global-IJDST, IGI Global-IJBDCN, Scalable Computing: Practice and Experience). She acted as chair and organizer in international conferences and workshops (e.g., IEEE-ISCC, EAI-CN4IoT, IFIP-ESOCC,...). Her main research interests include distributed systems and wireless communications, especially with regard to the design and development of Cloud solutions for IoT services and applications.

Antonino Galletta graduated in 2016 at the University of Messina. Currently, he is attending a Ph.D. course in Distributed Systems at the University of Messina. He worked at University of Messina as software developer for three years. His main activities focus on Cloud technology development, especially with regard to Big Data, MongoDB, Smart Sensing. He is also working as consultant for the development of solutions based on the FIWARE technologies.

Lorenzo Carnevale received the master degree in Engineering and Computer Science from University of Messina, Italy, in 2016. From 2015 to 2016, he collaborated with IoT companies and associations in order to design and develop IoT applications for smart home. In the late 2016, he was member of the technical coaches group of the frontierCities initiative, maturing the experience about FIWARE technologies and EU projects. Currently, he is attending a Ph.D. course in Distributed Systems, at the University of Messina, with specialization in Serverless and Big Data technologies, focusing the activities on eHealth and Smart City domains.

Jiafu Wan has been a Professor in School of Mechanical & Automotive Engineering at South China University of Technology (SCUT) since Sep 2015. He joined in SCUT in May 2014. He received the Ph.D. degree in Mechatronic Engineering from SCUT in Jun 2008. From Jul 2003 to Apr 2014, he respectively served as an Assistant Lecturer, Lecturer, and Associate Professor at Guangdong Mechanical & Electrical College, Guangzhou, China. From Nov 2008 to Jun 2012, he was a Postdoctor of Computer Science and Engineering in SCUT. In 2010, he became an Associate Research Fellow and a Provincial Talent Cultivated by "Thousand-Hundred-Ten" Program of Guangdong Province, China. His research interests include Cyber–Physical Systems, Intelligent Manufacturing, Big Data Analytics, Industry 4.0, Smart Factory, Cloud Robotics and Internet of Vehicles.

Massimo Villari is Associate Professor in Computer Engineering at University of Messina (Italy). He is actively working as IT Security and Distributed Systems Analyst in Cloud Computing, virtualization and Storage. For the EU Projects "RESERVOIR" he leaded the IT security activities of the whole project. For the EU Project "VISION-CLOUD", he covered the role of architectural designer for UniME. He is currently Scientific ICT Responsible in the EU Project frontierCities, the Accelerator of FIWARE on Smart Cities — Smart Mobility. He is strongly involved in EU Future Internet initiatives, specifically Cloud Computing and Security in Distributed Systems. He is co-author of more of 130 scientific publications and patents in Cloud Computing (Cloud Federation), Distributed Systems, Wireless Network, Network Security, Cloud Security and Cloud and IoTs. He was General Chair of ESOCC 2015 and IEEE-ISCC 2016. Since 2011 he is a Fellow of IARIA, recognized as a Cloud Computing Expert, and since 2011 he is also involved in the activities of the FIArch, the EU Working Group on Future Internet Architecture. In 2014 was recognized by an independent assessment (IEEE Cloud Computing Transaction, Issue April 2014) as one of World-Wide active scientific researchers, top 27 classification, in Cloud Computing Area. He is General Chair of EAI-CN4IoT. He is Editor In Chief of EAI Endorsed Transactions on Smart Cities. Currently he is Scientific Responsible for UniME-IRCCSME Cloud initiative in eHealth: http://healthycloud.irccsme.it/