Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 286860, 9 pages
http://dx.doi.org/10.1155/2015/286860

Research Article

Towards the Integration between IoT and Cloud Computing: An Approach for the Secure Self-Configuration of Embedded Devices

Antonio Puliafito, Antonio Celesti, Massimo Villari, and Maria Fazio

DICIEAMA, University of Messina, Contrada di Dio, Sant'Agata, 98166 Messina, Italy

Correspondence should be addressed to Antonio Celesti; acelesti@unime.it

Received 17 April 2015; Accepted 2 September 2015

Academic Editor: Melike Erol-Kantarci

The secure boot-up and setup of Internet of Things (IoT) devices connected over the Cloud represent a challenging open issue. This paper deals with the automatic configuration of IoT devices in a secure way through the Cloud, in order to provide new added-value services. After a discussion on the limits of current IoT and Cloud solutions in terms of secure self-configuration, we present a Cloud-based architecture that allows IoT devices to interact with several federated Cloud providers. In particular, we present two possible scenarios, that is, single Cloud and a federated Cloud environments, interacting with IoT devices and we address specific issues of both. Moreover, we present several design highlights on how to operate considering real open hardware and software products already available in the market.

1. Introduction

Internet of Things (IoT) is the next step evolution of Internet, where any physical object/thing having/equipped with computation and communication capabilities could be seamlessly integrated, at different levels, into the Internet. The exploitation of Cloud computing technologies is challenging to support the development of IoT systems, because it guarantees high scalability and reliability of the available services. Thus, IoT and Cloud computing offer new possibilities for sharing data and services through the Internet, by introducing a dynamic global network system with self-configuring capabilities based on standard and interoperable communication protocols [1].

As highlighted in the *Digital Agenda for Europe [2]*, one of the key challenges for the European Commission is to have a globally competitive Cloud infrastructure for the "Internet of Services" interconnected with "Things" distributed over remote areas. IoT is currently applied in many application fields, such as in buildings construction, car traffic monitoring, environments analysis, health-care assistance, weather forecast, and video surveillances. As a consequence, IoT will offer new services for making cities "Smarter" and it will improve the interaction of people and IoT devices/services with the surrounding environments, increasing the Citizens' quality of life. In these scenarios, security is one of the major factors hampering the rapid and large scale adoption and deployment of IoT and Cloud computing.

There is no limit to the possible scenarios that can be accomplished putting together IoT and Cloud computing and several architectures have been proposed up till now [3]. In our opinion, IoT can appear as a natural extension of Cloud computing, in which the Cloud allows us to access IoT-based resources and capabilities, to manage intelligent pervasive environments. In addition, Cloud computing can support the delivery of IoT services. Thus, an IoT service can be considered as an on-demand Sensing and Actuation as a Service (SAaaS). One of the main problems in deploying IoT devices is the self-configuration of such devices that is necessary to interconnect them over the Cloud.

In our vision, an IoT device should be able to configure itself to interact with the Cloud in a secure way and should automatically customize its behavior by downloading required features from the Cloud. From the user point of view,

when the user turns on his/her IoT device and connect it via WiFi (or other communication technologies), he/she just has to wait for the self-configuration of the device and, then, can start to use it.

In order to self-configure IoT devices in a secure way and allow them to interact over the Cloud, devices should be equipped with capabilities including security keys, cryptographic algorithms, and hidden IDs. This approach is already a reality. An example is represented by http://www.my-devices.net/, that is, a Cloud provider delivering secure remote access services to embedded devices via HTTP(S) or other TCP-based protocols. As well, *Temboo* (https://www.temboo.com/) provides innovative commercial solutions to interconnect IoT devices with Cloud services (e.g., Storage, Processing, and Messaging) using simple Application Program Interfaces (APIs). These examples represent only "a few drops in the ocean of IoT and Cloud computing," due to the great interest of research and business companies in this application field.

In our previous work [4], we analyzed current issues on the self-configuration of IoT devices connected over the Internet to Cloud providers. Specifically, we proposed a secure approach for the boot-up and setup of embedded devices. In this paper, we discuss how to apply our solution in real environments, presenting two possible scenarios: a Single-Cloud and a federated Cloud environment interacting with IoT devices. Moreover, we present several design highlights, discussing how to operate considering real open hardware and software products already available in the market. The solution proposed hereby is aimed at IoT enterprises that consider Cloud computing strategic to improve their business.

The rest of the paper is organized as follows. Section 2 discusses related works. Section 3 presents two challenging scenarios integrating IoT and Cloud computing. In Section 4, we discuss the main factors involved for a secure self-identification of IoT devices. In Section 5, we propose an IoT Cloud-based architecture. In Section 6, we discuss how IoT devices joining the Cloud system can self-register themselves to perform a self-configuration process. Section 7 concludes the paper.

2. Related Works

Security in IoT and Cloud computing is a widely discussed topic that hardly influences the rapid and large scale adoption and deployment of such technologies [5, 6]. In [7], the authors investigate security issues and challenges on IoT-based Smart Grids (SG) and define the major security services that should be considered when dealing with SG security. An approach to simultaneously scan several IoT objects in a short time is presented in [8]. The authors present the notion of Probabilistic Yoking Proofs (PYP) and introduce three main criteria to assess related performance: cost, security, and fairness. The proposal combines the message structure of classical grouping proof constructions with an iterative Poisson sampling process where the probability that each object is sampled varies over time. A key distribution approach for secure e-health applications in IoT is presented in [9], where the authors conduct a formal validation of security properties. A secure mutual authentication scheme for an RFID implant system is presented in [10]. The authors propose a scheme that relies on elliptic curve cryptography and the D-Quark lightweight hash design. The D-Quark lightweight hash design is tailored for resource constrained pervasive devices, considering costs and performance. The computational performance analysis shows that the proposed solution has 48% less communication overhead compared to existing similar schemes. In [11], the authors propose a secure and scalable IoT storage system based on revised secret sharing scheme with support of scalability, flexibility, and reliability at both data and system levels. Shamir's secret sharing scheme is applied to achieve data security without complex key management associated with traditional cryptographic algorithms. The original secret sharing scheme is revised to utilize all the coefficients in polynomials for larger data capacity at data level. In [12], the authors propose an approach to provide secure IoT services using the Datagram Transport Layer Security (DTLS) as the de facto security protocol. In particular, they examined problems in applying the DTLS protocol to IoT, which comprises constrained devices and constrained networks. To solve such problems, they separate the DTLS protocol into the *handshake phase* (i.e., establishment phase) and the *encryption phase* (i.e., transmission phase). An overview of the main security challenges in IoT-aided robotics applications is presented in [13] that is specifically focused on network security. In [14], the authors investigate the possibility to unify resilient Cloud computing and secure IoT in Smart Cities scenarios. Considering the self-configuration issue of IoT devices in a Cloud computing scenario, in [15], the authors present an interesting IoT Cloud architecture exploiting Arduino devices, whereas, in [16], the authors propose an IoT service provisioning using a Cloud computing system. However, both [15, 16] lack secure self-configuration mechanisms during the boot up phase. In fact, they require human interactions and an a priori configuration of devices. In this paper, we try to overcome this gap.

3. Single and Multicloud Scenarios for IoT

In this section, we present two challenging scenarios that we, respectively, identify as "Single-Cloud" and "Multicloud" [17] (see Figures 1 and 2). Both scenarios include different users holding several IoT embedded devices connected to Internet (e.g., through a domestic WiFi network). Each device is able to automatically configure itself downloading its configuration from a given Cloud provider. As shown in Figure 1, in the Single-Cloud scenario, several datacenters belonging to a Cloud operator are spread over the world. For example, datacenter A is placed in USA, datacenter B is located in Europe, and datacenter C is placed in Asia. Each datacenter collects data coming from IoT embedded devices connected in the geographical area that it serves.

The Multicloud scenario shown in Figure 2 is much more challenging than the previous one, because datacenters belong to different cooperating Cloud providers [18]. In the example, Cloud B is a device manufacturer, whereas Clouds A

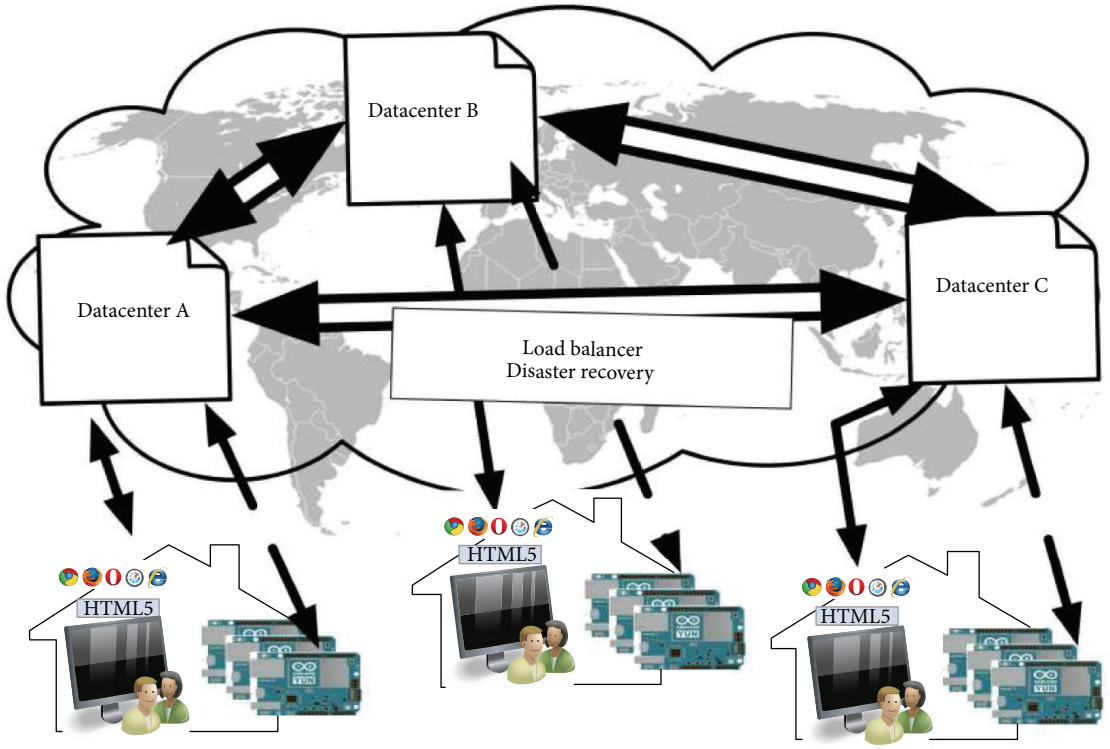

Figure 1: Single-Cloud scenario with one Cloud operator distributed among more sites, IoT devices, and customers.

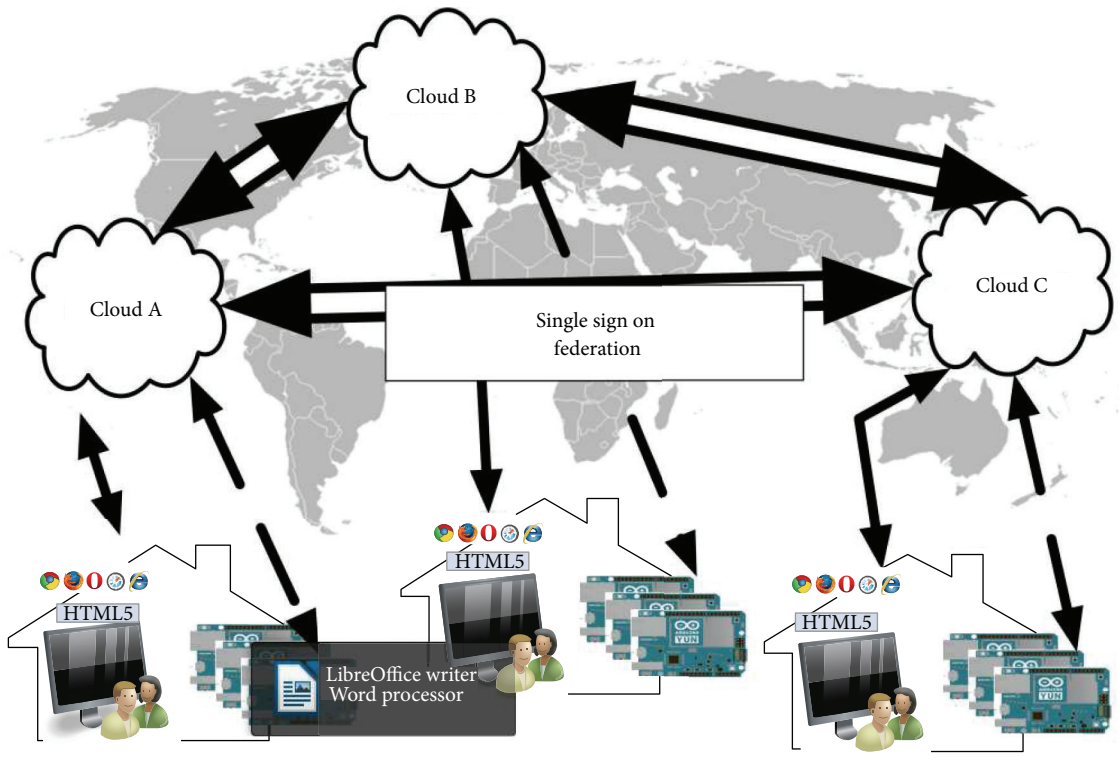

Figure 2: Multicloud scenario with more Cloud operators, IoT devices, and customers.

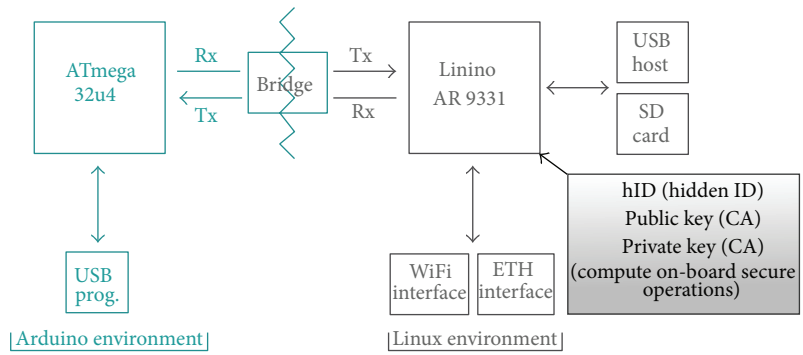

Figure 3: Arduino Yun extended with security capabilities.

and C are IoT service providers. Clouds A, B, and C establish a federation relationship with the objective to improve their business. An interesting question is how to establish agreements among these Clouds. Cloud B can provide different kinds of services to its customers but also to Clouds A and C that provide IoT services. In addition, Cloud B can be the third-party entity responsible for certifying the goodness and trustiness of its IoT devices. A similar situation already takes place in Trusted Computing considering the Trusted Platform Modules (TPMs) endorsed in motherboards by manufacturers. At the beginning, Clouds A and B make an agreement for a Single-Sign-On service. When an IoT embedded device wants to access Cloud A, it has to perform an authentication on Cloud B. If the authentication succeeds, Cloud A will trust Cloud B. Thus, Cloud A will complete the registration of the device. After that, Clouds A and B establish a federation relationship. To this aim, Cloud B (i.e., the device manufacturer) tracks each IoT embedded device in terms of firmware version, bug reporting, and so forth, thus being able to authenticate it without knowing nor the real location of the device neither its owner. Cloud A (i.e., IoT service provider) can be notified from Cloud B. Cloud B needs to identify the device without exchanging users' data and device Media Access Control (MAC) addresses.

In Figure 2, the federation agreement for SSO can be further extended implementing additional mechanisms for interoperability among Clouds. According to the reference scenario, each user gathers data coming from his/her embedded IoT devices by means of a Cloud's web portal and stores them into the Cloud. To manage such big data, different types of information from different providers have to be processed. For example, to provide a seamless service to the user, Clouds A and B should manage the following information:

(i) Devices information on Cloud B.

(ii) User authentication, authorization, and accounting information on Cloud B.

(iii) Devices configuration for initial setup on Cloud A.

(iv) Devices configuration for life-cycle operations on Cloud A.

(v) Devices sensed data on Cloud A.

To store and manage all these data in a scalable way, every Cloud should host a local distributed database. In a shared and opportunistic configuration of Clouds belonging to a federation, it is also possible to think about a global database distributed among different federated domains, which works as a global big catalogue holding information for users and devices (login, keys, code, IDs, MAC repository, etc.).

4. Towards Secure Self-Identification of IoT Devices

In this section, we discuss how existing IoT embedded devices might be extended and used to carry on self-identification in the scenarios previously described. Specifically, IoT devices should be on-boarded with security keys, cryptographic algorithms, and hidden IDs (hIDs). Thus, we analyze a well-known IoT platform, that is, Arduino Yun, in order to discuss the effectiveness of self-identification mechanisms.

4.1. Arduino Yun. The Arduino open hardware framework is a consolidated architecture able to fulfill IoT requirements especially for its cheapness and simplicity of utilization. Many versions, shields, and extensions exist over the market for the Arduino platform. Among them, Arduino Yun is the framework able to provide Arduino capabilities along with Linux embedded features. Specifically, Yun is different from the other Arduino boards because the ATmega controller communicates with the Atheros AR9331 processor. The latter supports a Linux distribution based on OpenWRT named Linino, offering a powerful networked computer with the easiness of Arduino.

Figure 3 shows the Yun architecture. The left side of the picture depicts the Arduino part, whereas in the right side the Linino part is shown. Yun has built-in WiFi/Ethernet boards that provide communication capabilities. In our scenarios, Linino can be used to accomplish security features and to perform the interactions with the Cloud. In particular, Webclient (i.e., Curl), XMPP client, Python, and OpenSSL can be easily developed on this device for interacting with other devices and with the Cloud in secure way.

4.2. Security Keys, Cryptographic Algorithms, and Hidden IDs. In order to achieve the scenarios discussed in Section 3, an IoT device such as Arduino Yun should be equipped with

a new component mounted on the board by the manufacturer and offering several security capabilities. In particular, these security capabilities should include security keys (e.g., a couple of public/private keys X509v3 based (K_{pub}, K_{priv})), cryptographic algorithms, and a hidden ID (hID). The hID is a numeric serial-number used by manufacturer for recognizing each board. It is hidden because no one must read it. Here, we introduced the concept of Obfuscated ID (obH) derived from the MD5 hashing function. The major property of a hashing function is its incontrovertibly; in fact it is also defined as a one-way function (i.e., from the output of a hashing function it is not possible to deduct the input). Hence, obH is useful to track boards hiding information on public MAC addresses and board owners:

$$\text{obH} = \text{hash}(\text{hID}, \text{MAC}). \quad (1)$$

obH represents a board index does not provide sensitive information on the board itself and, hence, can be stored in whichever public database. In any communication between the device and the Cloud operator (e.g., Cloud A in Figure 2), a Message (M) should be included in the body of all communications concatenating obH, MAC, and a public key K_{pub} to implement secure communications. Consider

$$M = \text{concat}\left(\text{obH}, \text{MAC}, K_{pub}\right). \quad (2)$$

The signature mechanism based on the public key K_{pub} guarantees the trustiness of the sender. K_{pub} is assigned at the production stage by the Certification Authority (CA) of the manufacturer of the IoT device. On the contrary, the private key K_{priv} is not accessible externally from the chip embedded in the device, but it can be used by internal security algorithms:

$$\text{SM} = \text{signature}\left(K_{priv}, M\right). \quad (3)$$

4.3. Adding Secure Hardware Capabilities. Trusted Computing (TC), defined by the Trusted Computing Group (TCG) [19], combines hardware and software security mechanisms to enhance the security level of computing environments. The main goal of TC is to provide stronger security than the traditional software-based security systems and to enforce the integrity of a system when it interacts with other ones. The distinguishing feature of TC is the incorporation of Roots of Trust (RoT) that aims to perform specific functions in a secure way, such as measurement, storage, reporting, verification, and/or update. TC implies the adoption of a hardware chip called Trusted Platform Module (TPM) that is able to provide RoTs and to extend trust to the other parts of the device by building a chain of trust. It offers facilities for the secure generation of cryptographic keys and it is capable of performing platform authentication, since each TPM chip has a unique and secret RSA key burnt into it as it is produced (i.e., the Endorsement Key (EK)). The TPM includes capabilities such as machine authentication, hardware encryption, signing, secure key storage, and attestation. Born for securing traditional Personal Computers, the TCG is currently looking at both embedded and mobile devices whose reference architecture specification drafts were released, respectively, in April and June 2014. The specifications provide guidelines on how to onboard the TPM in a device even though there have not been so many implementations yet on real hardware devices. TC and embedded systems are at the early stage; however, in our opinion, TC is a valid solution to develop hardware security capabilities in IoT devices interacting with the Cloud.

5. An IoT Cloud-Based Architecture

In this section we describe an IoT Cloud-based architecture that is able to support the self-identification on secure communications, as described in the previous sections. To this aim, we specifically refer to the Cloud-Enabled Virtual Environment (CLEVER), a secure Message-Oriented Middleware (MOM) able to support federated Cloud services developed in oru labs [20]. CLEVER sets up a Cloud system able to manage both sensing and virtualization services using the XMPP protocol. In addition, the middleware supports big data management by means of the NoSQL database MongoDB. Figure 4 shows our IoT Cloud-based architecture that supports both Single-Cloud and Multicloud scenarios. The infrastructure needs to rely on virtualization platform for guaranteeing a high degree of flexibility and elasticity through Virtual Machines (VMs). All the Physical Machines (PMs) are equipped with a piece of CLEVER middleware for the cluster managements. The circle shape with the label A represents the cluster where virtual instances of IoT services are lunched. The Cloud is aimed at the maintenance of the current solution but also for customers management. In the distributed instance of MongoDB, information on IoT obH, device status (e.g., firmware version, on-board add-ons), and the data devices collect are stored. The XMPP server sets up bidirectional bus channels. The ellipse shape with the B label highlights the execution of several VMs into a PM. Each VM contains different services and interacts with different customers and devices. Shapes with C and D labels show possible services, such as Apache, MySQ, PHP services, VMs with a minimal version of CLEVER, the XMPP server, and MongoDB. The D shape represents a container in which sensed data are stored and confined inside a VM. Many IoTs, having the XMPP client, are dynamically assigned (after the first contact with the secure webclient) to the XMPP server deployed into the VM. Many customers (since with their IoTs) can rely on the same VM. This service can scale up and down in relation to the number of customers and IoTs. Shape C represents the main web portal service used for customers and for making MAC-Users associations. Even it can scale up and down with respect to the number of customers.

5.1. Arranging Cloud Systems Using CLEVER. As depicted in Figure 5, each CLEVER cluster includes several PMs organized in a cluster. Each PM is controlled by a management module, called Host Manager (HM), and only one PM runs a cluster management module, called Cluster Manager (CM). CM acts as interface between Cloud clients (e.g. IoTs, applications, web services, and end-users exploiting Cloud

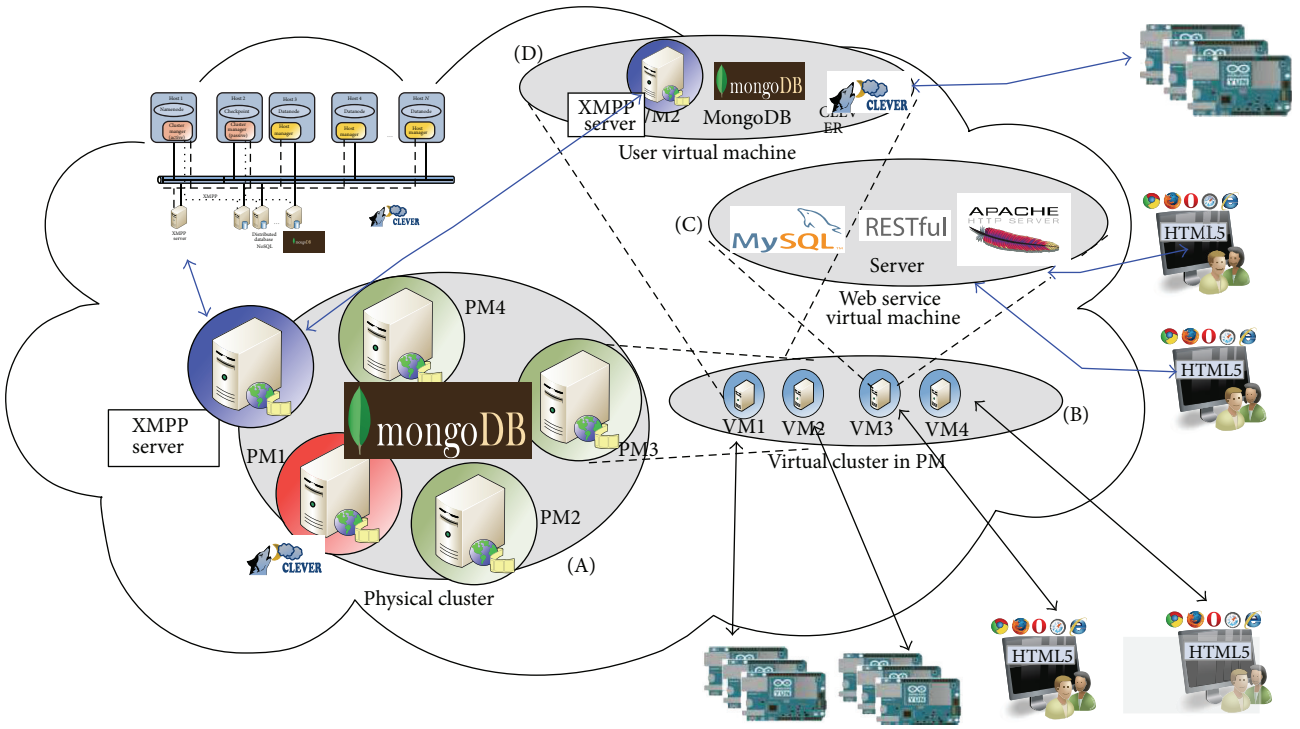

FIGURE 4: Example of IoT device manufacturer managing its own datacenter using our architecture. The figure shows several web clients and IoT devices extended with security capabilities interacting with a Cloud system arranged using CLEVER.

resources) and software agents running on PMs. CM receives commands from clients, gives instructions to HMs, elaborates information, and finally sends back results to the clients themselves. It also performs tasks for the management of Cloud resources and the monitoring of the working state of the cluster. A CM is elected among all the HMs using a distributed *self-election protocol* and it works in the ACTIVE mode. A second CM is also elected as backup, but it is configured in the MONITORING mode. The MONITORING CM is not involved in the cluster management, but it keeps a synchronized copy of the internal state of the ACTIVE CM; hence, it is able to set up all the active services if the ACTIVE CM fails. To make our dissertation easier, from now on, we refer to the ACTIVE CM as just CM.

Communications among distributed components into a CLEVER Cloud are based on XMPP, due to its flexibility and high level of reactiveness. A Jabber/XMPP server provides basic messaging, presence, and XML routing features within the Cloud. All the PMs in the Cloud are connected via the Multiuser Chat (MUC) labeled *Main Room* and cooperate according to the CM orchestration directives. A MUC, identified as *Shell Room*, allows clients to submit their requests and receive the service. Due to the cluster-based architecture of CLEVER, only CM and clients access the Shell Room. These two communication channels are shown in Figure 5 as thick black lines.

To set up a federation, CMs belonging to different Clouds exchange messages through the MUC identified as *Federation Room*, and only CMs of federated Clouds access it. MONITORING CMs cannot enter the Federation Room, since they are not directly involved in resource management.

The XMPP server necessary to establish the federation and, hence, to manage the Federation Room can be entrusted by a third-party entity, which sets up a Jabber Server out from the domains of Cloud providers, only to fulfill federation requirements. Since the MUC can be accessed only by components into federated domains, it is necessary to check for their credentials. To avoid a priory static configuration of accounts into the XMPP server, the third-party entity has to authenticate component credentials. To this aim, XMPP federation capabilities can be exploited. The third-party entity does not need to maintain the credentials of all CMs involved in the federation, but a trustiness agreement among all the Jabber Servers allows setting up an XMPP federation where credentials of components are verified by at least one Jabber Server (tokens).

5.2. Interaction of IoT Devices with the Cloud: A Hybrid Approach. In this section, we discuss a hybrid approach for allowing the IoT device to interact with a Cloud system, for example, arranged by means of CLEVER. To describe the interaction approaches, we refer to two possible cases.

Case 1. We consider the "simplified" scenario, where the device manufacturer coincide with the Cloud operator (i.e., Cloud B); hence, it has a full control over the board with bidirectional XMPP communications.

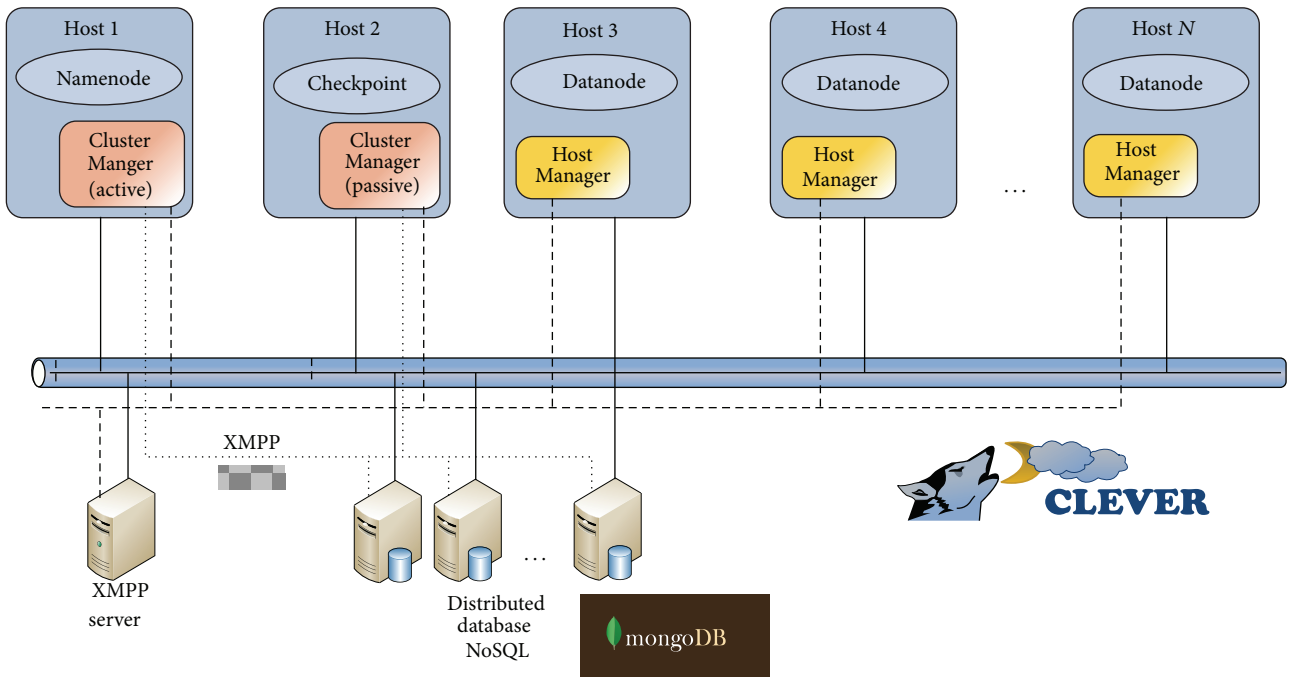

Figure 5: CLEVER architecture.

Case 2. We consider the "challenging" scenario, where the device manufacturer (i.e., Cloud B) makes an explicit full agreement with the Iot service provider (i.e., Cloud A) and both belong to a Cloud federation. In this scenario, Cloud B offers this Plaftorm as a Service (PaaS) to other Clouds (XMPP server, authentication system, NoSQL database, etc.). Two chat-rooms are preconfigured between the XMPP client of the IoT device and the XMPP client of the Cloud provider: the first one is used for firmware deployment and bug management. The device manufacturer (i.e., Cloud B) uses such a communication system to communicate with the IoT device. Cloud A can join this communication. The second chat-room is used for sending sensed data: the IoT service provider (i.e., Cloud A) is such a communication system for its purposes, but Cloud B cannot hear the communication, if an encrypted channel is used.

This hybrid interaction approach includes two different types of client running on IoT devices that operate at different stages: a web client for secure RestFul communication and the XMPP client for bidirectional communication.

(1) RESTFul Web Client. The main idea is to use the web client after the first association of an IoT device with its home WiFi network. Thus, the web client accesses Internet and contacts the default web server of its Cloud provider using the https protocol. The IoT device, such us Arduino Yun, using its private key performs a challenge-response process to perform a SSL mutual authentication with the web server, and if it succeed, the IoT device sends a message M (see (2) in Section 4.2) in a JSON document's body to the web server. Thus, the IoT device is recognized and binds to the service. The secure web client is used only at the beginning for the simplified scenario. After that the message M is sent; the web server at the Cloud of the manufacturer takes the control of the IoT device and can perform the subsequent actions using the XMPP protocol.

(2) The XMPP Client. It allows simplifying the interaction among the IoT device and several Cloud providers, especially in the "challenging scenario." Considering such scenario, let us assume that Cloud A (the IoT service provider) needs to set up two elements, that is, a SD card to plug into the IoT device and a web service. The SD card contains data for accessing Cloud A (e.g., the URI and the public certificate of the CA). The web service is able to perform a SSO authentication, redirecting a request to Cloud B for getting authentication tokens. Hence, Cloud B (the device manufacturer) and Cloud A (IoT service provider) are both in charge of finalizing the step of early device identification. The XMPP protocol was originally designed for efficiently managing messages among peers over the Internet. The nature of XMPP allows to transparently gain the access over the Internet, using mechanisms of firewall pass-through, without any user intervention, establishing bidirectional communications. It is based on the XML standard; hence, it is rather simple to be extend for enforcing security and signing mechanisms. In addition, it is possible to accomplish signature and encryption of XML messages leveraging public/private keys. Any device with the OpenSSL stack can use such functionalities. The XMPP client starts its execution; after that, the interaction of web client successfully ends. This XMPP client can be used both in the simplified and in challenging scenarios.

6. Registration Strategies of IoT Devices Joining the Cloud

The IoT device, for example, the Arduino Yun extended with security capabilities, can follow two different registration methods:

(i) Unsupervised: autoregistration of MAC address and obH.

(ii) Supervised: end-user web registration of MAC address and obH.

In both cases, the end-user needs to enable the IoT device (e.g., the Arduino Yun board) to maintain the WiFi network association using the wps button on his wireless AP. Hence, the IoT device can access the Internet performing the authentication as described in Section 4. In the supervised case, the IoT device board flashes an orange LED, and after its partial registration it shows an orange fixed-on LED. The full registration is achieved when the end-user associates the IoT device board with his/her web profile. The users adopt a website to register the board, in particular, typing the MAC address shown in the external part of the box provided by the manufacturer. If the MAC in M matches the MAC typed in the website, the board flashes a green LED, and the user can confirm the operation; otherwise (obtaining no flashing LED), he/she should repeat the procedure. After that, the full registration has been accomplished, and the board shows a green fixed-on LED. Now the Cloud has the full control of the board; hence it can deploy firmware, managing configuration, install software, and so on. The user only pushed a button (wps) and typed a code in the website of the Cloud operator.

The solution proposed can follow two main version branches. The early version is strongly bound with the device manufacturer (i.e., Cloud B) that is in charge for the following tasks: (i) preprogramming the IoT device, (ii) updating the firmware, and (iii) releasing new applications. The second version looks at situations in which the device manufacturer is in charge for task (i), whereas the other IoT service providers (i.e., Clouds A and C) are able to deal with tasks 2 and 3. These tasks are possible if Cloud IoT service providers establish an agreement with the Cloud acting as device manufacturer. Cloud acting as IoT device manufacturer have to equip IoT device with SD card to achieve such scenario. Thus, our idea is to leverage the Single-Sign-On concept for allowing a device to make a registration on a Cloud IoT service provider, interacting with the Cloud acting as IoT device manufacturer. When the IoT device is able to establish a connection, it reaches Cloud IoT service provider; hence, its service operates a redirection to the Cloud IoT device manufacturer that acts as identity provider, for verifying the obH message with its signature. If the Cloud manufacturer recognizes the device, it releases a token; hence, the Cloud IoT service provider can complete the unsupervised registration. The supervised registration is performed by end-users that can initially follow the same approach. To accomplish this scenario, the Cloud IoT service provider needs to make an explicit agreement with the Cloud IoT device manufacturer for understanding how to interact with each other. The obH message cannot allow the device manufacturer Cloud to know detailed info about the board. However, it can track its general status, firmware version, bugs, and so on.

7. Conclusion

In this paper, we discussed an approach to integrate the IoT with Cloud computing. In particular, a system is presented analyzing the different elements involved and how they interact with each other. Using the Arduino Yun for example, we discussed how IoT devices can be extended to support the interaction with the Cloud. In particular, we focused on a system that allows a Cloud provider to deploy the firmware and configure the device and on the one hand to perform sensed data transfer from the device to the Cloud provider. In the end, we discussed how the overall system works, also presenting two possible Cloud scenarios. Currently, IoT devices are at the early stage and, as discussed in this paper, they are not ready yet to support complex Cloud scenarios, even though the roadmap toward innovative Cloud IoT services begins to be tracked.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research was supported by the European Unions Horizon 2020 Research and Innovation Programme project BEACON under Grant Agreement no. 644048.

References

[1] M. Fazio and A. Puliafito, "Cloud4sens: a cloud-based architecture for sensor controlling and monitoring," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 41–47, 2015.

[2] Unleashing potential of Future Internet and Cloud computing, November 2013, https://ec.europa.eu/digital-agenda/en/news/unleashing-potential-future-internet-and-cloud-computing.

[3] M. Villari, A. Celesti, M. Fazio, and A. Puliafito, "AllJoyn Lambda: an architecture for the management of smart environments in IoT," in *Proceedings of the International Conference on Smart Computing Workshops (SMARTCOMP '14)*, pp. 9–14, Hong Kong, November 2014.

[4] M. Villari, A. Celesti, M. Fazio, and A. Puliafito, "A secure selfidentification mechanism for enabling IoT devices to join cloud computing," in *Internet of Things. IoT Infrastructures*, R. Giaffreda, D. Cagov, Y. Li, R. Riggio, and A. Voisard, Eds., vol. 151 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 306–311, Springer, Berlin, Germany, 2015.

[5] Y. H. Hwang, "Iot security & privacy: threats and challenges," in *Proceedings of the 1st ACM Workshop on IoT Privacy, Trust, and Security (IoTPTS '15)*, p. 1, ACM, Singapore, April 2015.

[6] Z.-K. Zhang, M. C. Y. Cho, and S. Shieh, "Emerging security threats and countermeasures in IoT," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (ASIA CCS '15)*, pp. 1–6, ACM, Singapore, April 2015.

[7] C. Bekara, "Security issues and challenges for the iot-based smart grid," *Procedia Computer Science*, vol. 34, pp. 532–537, 2014.

[8] J. M. de Fuentes, P. Peris-Lopez, J. E. Tapiador, and S. Pastrana, "Probabilistic yoking proofs for large scale IoT systems," *Ad Hoc Networks*, vol. 32, pp. 43–52, 2015.

[9] M. R. Abdmeziem and D. Tandjaoui, "An end-to-end secure key management protocol for e-health applications," *Computers & Electrical Engineering*, vol. 44, pp. 184–197, 2015.

[10] S. R. Moosavi, E. Nigussie, S. Virtanen, and J. Isoaho, "An elliptic curve-based mutual authentication scheme for RFID implant systems," *Procedia Computer Science*, vol. 32, pp. 198–206, 2014.

[11] H. Jiang, F. Shen, S. Chen, K.-C. Li, and Y.-S. Jeong, "A secure and scalable storage system for aggregate data in IoT," *Future Generation Computer Systems*, vol. 49, pp. 133–141, 2015.

[12] N. Kang, J. Park, H. Kwon, and S. Jung, "ESSE: efficient secure session establishment for internet-integrated wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 393754, 11 pages, 2015.

[13] L. A. Grieco, A. Rizzo, S. Colucci et al., "IoT-aided robotics applications: technological implications, target domains and open issues," *Computer Communications*, vol. 54, pp. 32–47, 2014.

[14] G. Suciu, A. Vulpe, S. Halunga, O. Fratu, G. Todoran, and V. Suciu, "Smart cities built on resilient cloud computing and secure internet of things," in *Proceedings of the 19th International Conference on Control Systems and Computer Science (CSCS '13)*, pp. 513–518, Bucharest, Romania, May 2013.

[15] A. A. Chandra, Y. Lee, B. M. Kim, S. Y. Maeng, S. H. Park, and S. R. Lee, "Review on sensor cloud and its integration with arduino based sensor network," in *Proceedings of the 3rd International Conference on IT Convergence and Security (ICITCS '13)*, Macao, China, December 2013.

[16] M. S. Aslam, S. Rea, and D. Pesch, "Service provisioning for the WSN cloud," in *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD '12)*, pp. 962–969, June 2012.

[17] A. Panarello, A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "A requirements analysis for iaas cloud federation," in *Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER '14)*, Setúbal, Portugal, April 2014.

[18] M. Fazio, A. Celesti, M. Villari, and A. Puliafito, "The need of a hybrid storage approach for IoT in PaaS cloud federation," in *Proceedings of the 28th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA '14)*, pp. 779–784, Victoria, Canada, May 2014.

[19] Trusted Computing Group (TCG), http://www.trustedcomputinggroup.org/.

[20] A. Celesti, M. Fazio, and M. Villari, "SE CLEVER: a secure message oriented Middleware for Cloud federation," in *Proceedings of the 18th IEEE Symposium on Computers and Communications (ISCC '13)*, pp. 35–40, Split, Europe, July 2013.