A COMPARATIVE STUDY OF CYBER-PHYSICAL CLOUD SYSTEMS

Gheorghe SEBESTYEN, Professor PhD,
Adela Bereş, PhD Candidate,
Technical University of Cluj-Napoca,
Béla GENGE, Associate Professor PhD,
"Petru Maior" University of Tîrgu Mureş

Abstract: *Cyber-physical systems are more and more integrated into Internet of Things (IoT) making it vulnerable to malicious attacks. These systems generate huge amount of information, making Cloud Computing a natural choice to store and process this data. Nevertheless, as shown in this paper, their integration raises security issues that need to be addressed. We present platforms for cyber-physical Cloud systems, as well as a set of security test scenarios we performed in order to discover their vulnerabilities.*

Index Terms: *Cloud computing, Security, Cyber-physical Systems*

1. Introduction

Nowadays there are huge volumes of digital information being generated, transmitted, processed or stored around the world. The amount of space and processing power of private companies and tenants is limited and usually not enough to handle all their data. More and more of these companies move their data to the Cloud. Cloud Computing provides the infrastructure and space to store large amounts of data and also process it in a timely manner, while providing availability and resilience.

Cyber-physical systems are becoming part of our daily life, being Smart Grids, sensors for humidity, air, pollution, wind or even body sensors to monitor our health. Smart cities are the next generation of cities in which we are going to live. Even today, cities like Barcelona have an infrastructure of sensors which are monitoring weather conditions or transportation in order to make the town smart, signaling alerts or helping to organize the municipality activities better. Cloud Computing comes as a natural choice to implement smart cyber-physical platforms due to its storage and computational capacity.

BDD-A23338 © 2015 Editura Universităţii „Petru Maior"
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

Security is an important aspect in Cloud integrated cyber-physical systems. Data needs to be safe throughout the whole communication chain, from the moment it is generated to the moment it is stored. Only authorized users are allowed to access the information stored in Cloud. Integrity of the data should also be guarded. Tampering with the data, bad data injection can cause false alarms or malfunctioning of the cyber-physical systems.

Briefly, in this paper we present the existing solutions and platforms which integrate Cloud Computing with cyber-physical systems focusing on the security aspects that they implement.

The paper is organized as follows. Chapter 2 depicts the existing researches and platforms which integrate Cloud Computing with cyber-physical systems. Chapter 3 is dedicated to the study and testing of two open source platforms: Sentilo for Smart Cities and Mirantis OpenStack. Chapter 4 discusses the findings of the studied literature and platforms. The paper concludes in Chapter 5.

2. Cyber-physical Cloud Systems

2.1 S. H. Shah et al. WSN integrated Cloud framework

The authors propose a framework to integrate wireless sensor networks with Cloud. The main modules of the architecture are:

- User Identity & Access Management Unit
- Monitoring & Metering
- Request Subscriber
- Data Processing Unit
- Pub/Sub Broker
- Data Repository

The framework is depicted in the following figure:

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

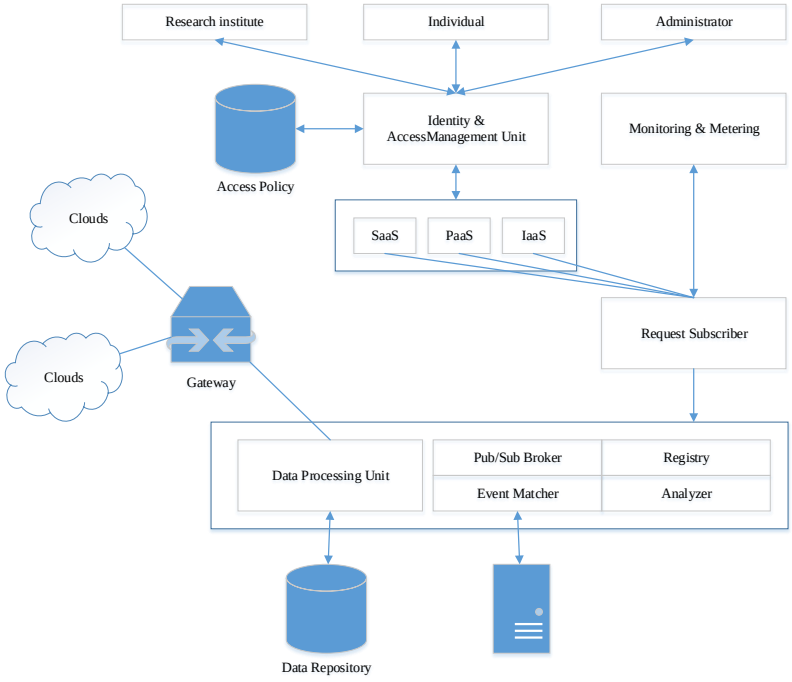


Figure 1 General overview of proposed architecture [1]

The users of the framework are not allowed to see or act on the data if they are not registered through the User Identity & Access Management Unit. Data coming from the WSN is passed through a gateway and sent to the Data Processing Unit which stores it in the Data Repository. Also it alerts the Pub/Sub Broker that new data was received. The Broker checks the subscriptions with the help of the Request Subscriber module and forwards the data to the respective users. Role-based Access Policy (RBAC) is used to authenticate and authorize users in the system. Also, in order to protect the data, the User Identity & Access Management Unit implements Diffie-Hellman keys and Kerberos.

2.2 Wen-Yaw Chung et al. Cloud Computing system for agricultural WSN

Wen-Yaw Chung et al. present an integrated framework for agriculture monitoring systems composed of a WSN which collects data from temperature, humidity etc. sensors and a Cloud platform which stores and presents the data and useful graphs to the users [2].

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

The proposed Cloud system has 1 master server and 4 slave computers which are doing the work of collecting, sampling and analyzing sensors data. Also the client communicates with the system through a web service. The data is stored in relational databases. The system was implemented using SQL database, stored procedures, Linq-to-Sql to query the data, web service which uses XML to send/receive the messages and the user interface was written in C#.
The data can be visualized in two ways: either a data curve or a panorama map. On the panorama map the user can click on the specific sensor to see its information and data that it generated.

2.3 Multi-Level Authentication for Sensor-Cloud Integration Systems
In this paper, the authors describe an authentication system based on multi-level authentication technique which they modeled using Petri nets. This system is used to secure the data generated by the sensors and stored in the Cloud [3].
The levels on which the password gets generated then concatenated and checked are:

- Organization level
- Team level
- User level

The architecture of the system consists of multiple sensor networks which are connected to the Cloud platform. Data is being routed from the nodes to the base stations and then to the Cloud using the ant colony optimization technique. Afterwards it is stored in the Cloud. To gain access to the data a user must pass all levels of authentication.

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

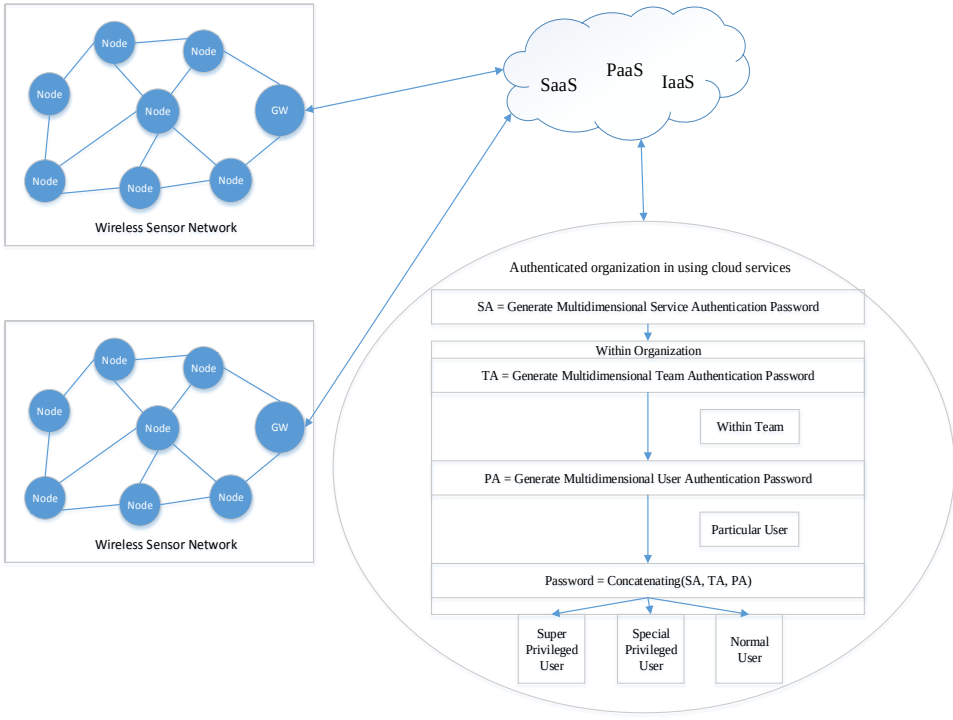


Figure 2 System Architecture of the Proposed Cloud computing based sensor data analysis environment [3]

2.4 Open.Sen.se architecture for integrating WSN with Cloud

The authors present a flexible and extensible architecture for a WSN Cloud integrated system. It has 3 layers:

- Sensor layer
- Coordinator layer
- Supervision layer [4]

The sensors are organized in the Sensor Layer. They are called End Devices and form a Mesh network which sends data through the base station to the Coordinator Layer. All sensor nodes use XBee ZB platform. The Coordinator Layer acts like a buffer, storing the data and sending it to the Supervision Layer at predefined intervals. The Supervision Layer upon receiving data connects to a web service to publish the data in the Cloud. An Open.Sen.se server is used to store and retrieve the data generated by the sensors. From a security point of view, a Sen.se key is generated for each user of the API which is supposed to be unique. The Open.Sen.se server provides a graphical display of the data through a Senseboard and also sends event notifications using predefined If-

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

conditions in the form of text messages or tweets if the value measured for a sensor is pass the limit set by the user.

2.5 Secure Cloud-based Architecture for e-Health WSNs

The architecture proposed in this paper has the following main components:

- WSN which collects the data
- User applications to access the stored data
- HealthCare Authority (HA) which control the security protocols
- The cloud servers where the data is stored

Security-wise the authors propose to use ABE (Attribute Based Encryption) and symmetric cryptography to encrypt the data. More specifically, they propose to encrypt each file with a randomly generated symmetric key (RSK) and encrypt the RSK with ABE. Both the encrypted file and the encrypted RSK are sent to the cloud for storage to allow fine grained data sharing with authorized users. HA generates and sends to each user his ABE security parameters which are a pair of access structure and secret key. The secret key is tagged with the user attributes set which represent the user privileges. This information is required to decrypt data that the user is allowed to access. The access structure represents the access policy that protects the user data. When a user encrypts the random symmetric key (RSK) that protects his data using this structure, he can be sure that only authorized users (who have the correct attributes) can decrypt and access his data [5].

Furthermore the communication between entities is performed via SSL and data is encrypted before being sent to the cloud server. This architecture guaranties the following security services:

- Fine-grained access control
- Integrity and authenticity
- Availability
- Collusion resistance

Using AES to encrypt the data and CP-ABE (Ciphertext-Policy ABE) to only encrypt the AES key, this system proved to be 27-47% faster than plain ABE encryption/decryption of the data.

2.6 Multi-agent system based architecture for secure Cloud

The authors propose an architecture which enforces the integrity of the data in Cloud. The architecture is based on Multi Agent Systems, a term used in artificial intelligence. The architecture has two layers and five

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

agents from which two are mainly used, Cloud Service Provider Agent (CSPA) and Cloud Data Integrity Backup Agent (CDIBA):

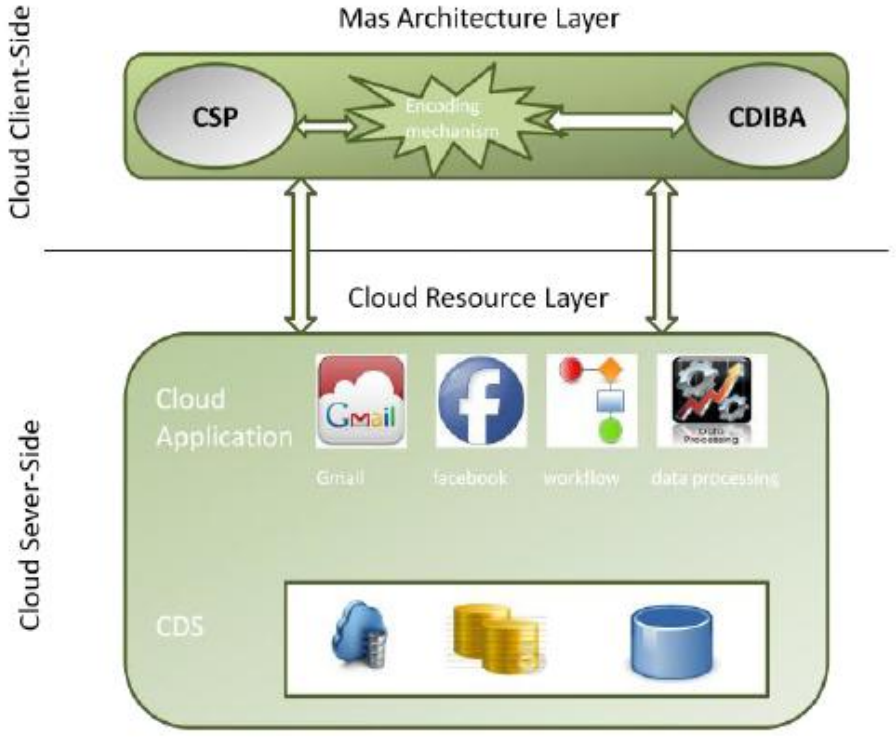


Figure 3 Multi-agent architecture for Cloud [6]

The main responsibility of CSPA is to back up the data in Cloud and to send alerts in case the data is altered or any kind of error happened, be it human, software or hardware. CDIBA is responsible with the encoding of data by using hash functions.

3. Security tests on open source platforms

3.1 Sentilo

Sentilo is an open source sensor and actuator platform designed to fit in the Smart City architecture of any city that looks for openness and easy interoperability. It has been sponsored by the Barcelona City Council, through the Municipal Institute of Informatics (IMI), as part of a project started in November 2011 conceived for defining the strategy and the necessary actions in order to achieve global positioning Barcelona as a reference in the field of Smart Cities [7].

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

It is developed using open standards and free software so everybody can benefit from it, experiment and use it in their own community and it's also supported by a variety of companies and cities. The main reason for using open source software was to come in the aid of public administration and private providers of sensors to make their interaction and implementation of smart cities transparent and homogenous.

Until now, the sensors networks implemented in cities or other environments are mainly proprietary solutions. This makes them dependent of a specific technology and also usually impossible to share data easily and without conversion between these networks and environments. The incompatibility also increases the amount of data that is duplicated in different systems and the costs of implementation and maintenance of such interconnected structures.

So the main idea that inspired the design of Sentilo is first and foremost the desire to create a cross-platform oriented infrastructure and data management service, escaping from vertical ITC "silo" solutions, for sharing information between heterogeneous systems and to easily integrate legacy applications [7].

The regular users of Sentilo are:

- municipalities or organizations who need to process lots of information received from the terrain generated by heterogeneous hardware and software devices (sensors, etc.), and who want a centralized and homogeneous way of managing and distributing these data across their information systems
- anyone from the IT world interested in contributing to the expansion of the "Internet of Things" and smart cities with the goal of improving citizens' quality of life [7]

Sentilo is already implemented in Barcelona from the beginning of 2013. The platform collects data from smart sensors all over the city including water, lightning and energy sensors and plans to expand its networks in the future years. Sentilo is also in testing phase in other Catalonian regions and cities.

Sentilo makes it easy to integrate sensors and actuators from different manufactures with applications to analyze and visualize the data. It acts like a middle layer between the sensor networks layer and the applications layer:

BDD-A23338 © 2015 Editura Universităţii „Petru Maior"
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

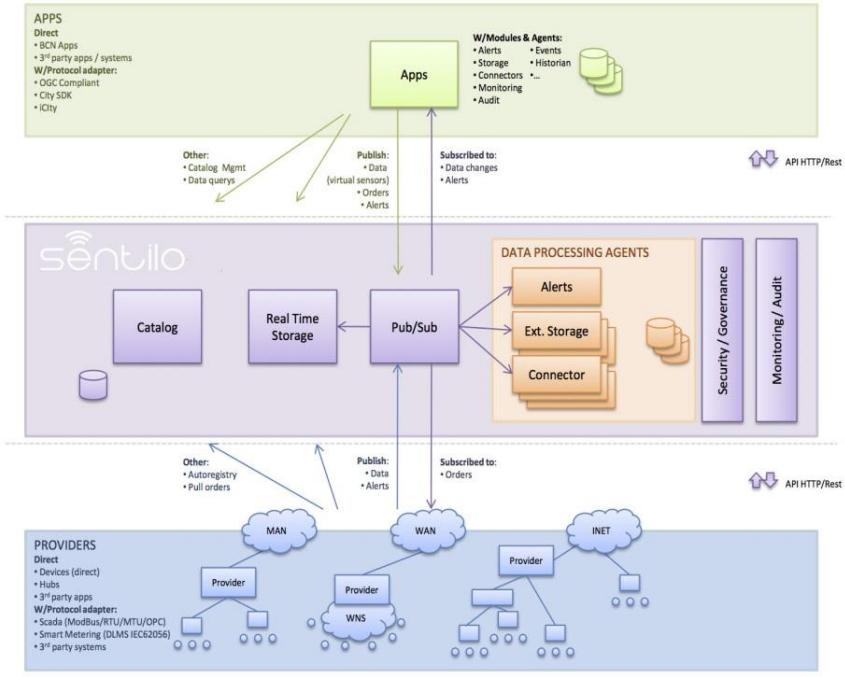


Figure 4 Sentilo platform architecture [7]

Sentilo includes:

- A front-end for message processing, with a simple REST API
- An administration console for configuring the system and managing the catalog
- A memory database, aimed to accomplish high performance rates
- A non-SQL database, in order to get a more flexible and scalable system
- A universal viewer provided as a public demo what can be used as a start point for specific business visualizers
- A basic statistics module that records and display basic platform performance indicators
- An extensible component architecture, to enlarge the platform functionality without modifying the core system. Sentilo starts with an initial set of agents: one for exporting data to relational databases and another to process internal alarms based on basic rules [7]

The open source technologies used to develop the platform are: Java, Redis, Mongo DB, JQuery, JSON.

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

The Catalog component is used by the web application. As stated before, Sentilo offers an administration console from where the users, applications, providers, sensors, sensors types, components etc. can be managed. Also it provides a graphical way to see the sensors, sensors locations and their data through maps and charts. Alerts and alarms can also be viewed from the web application.

Data received by the sensors is stored using the Real Time Storage component. Besides this task, it also has the role of making periodical backups of the data in the system.

The Pub/Sub component is written in Java and has two layers:

- Transport layer
- Service layer

In order to respond to client's requests the transport layer uses workers on separate threads. The requests are added to a queue and the workers pick them as soon as they are available. After a task is assigned, the worker sends it for processing to the Service layer. When it receives an answer it will forward the message back to the client:

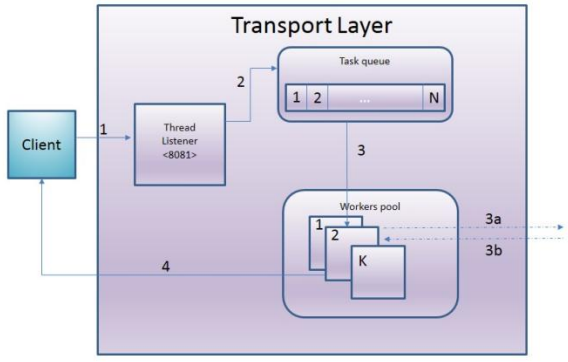


Figure 5 Transport layer request flow [8]

The Service layer is in charge of processing, validating and authorizing the requests. All tasks are done in memory using Redis, so the client doesn't have to wait long for the response.

BDD-A23338 © 2015 Editura Universităţii „Petru Maior"
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

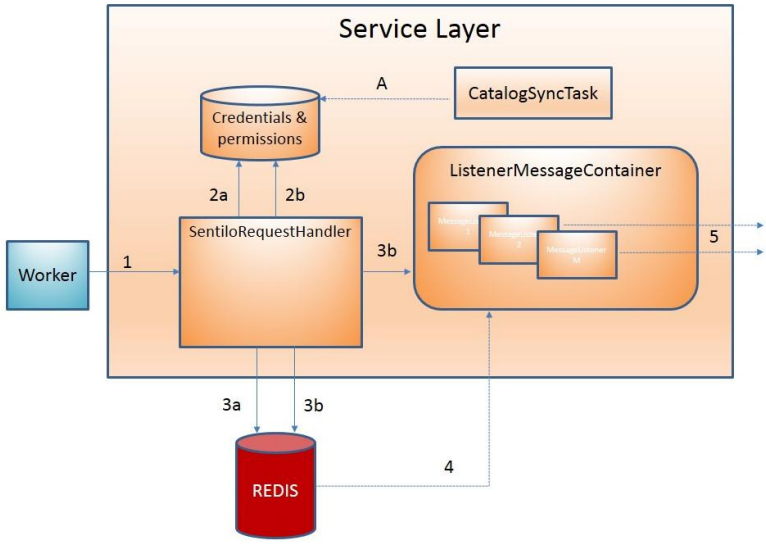


Figure 6 Service layer processing flow [8]

The processing flow of the Service layer is described below:

- The Worker delegates the request to the associated handler depending on the type of request (data, order, alarm etc.)
- The following validations are performed on each request:
  - (2a) Integrity of credential: checks the received token sent in the header using the internal database in memory containing all active credentials in the system
  - (2b) Authorization to carry out the request: validate that the requested action can be done according to the permission database
- the validity of the request parameters: mainly structure and typology
- After that stores the data in Redis (in memory) and depending on the type of data:
  - (3a) publish the data through publish mechanism
  - (3b) or register of the subscription in the ListenerMessageContainer
- Redis is responsible for sending the published information to ListenerMessageContainer event, who is responsible for managing the subscription in Redis as a client for any type of event
- The container notifies the event to each subscription associated with it [8]

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

The Credentials & Permissions database is updated periodically and refreshed also in memory where is used for processing.
Sentilo provides two agents:

- Relational database agent – used to export historical data to a relational database
- Alert agent – used for processing each data received by the platform and validate it with the business rules configured in the catalog [8]

Security-wise, each request must have an authentication token. If this authentication token is not present or is incorrect, the request is dumped and not processed. The permissions, credentials and authentication tokens are managed through the web application by the Catalog module. The authentication tokens are unique per application and provider, so even though all data is stored in one place, a client can have access only to its own data. To secure the push messages that the platform sends, Sentilo uses HMAC, specifically the SHA-512 hash algorithm. The messages are sent and received using REST API and in JSON format.
We conducted a set of tests on Sentilo platform. The test environment was set up on Windows 7 and Windows 8 servers and virtual machines. We configured and started:

- MongoDB
- Redis
- Sentilo Pub-Sub server
- Sentilo Web App Catalog
- Sentilo Agent Alert Server

We added the sensors through direct calls to the RESTFul API:
curl --request POST --header "IDENTITY_KEY: identity_key" --data '{"sensors":[{"sensor":"RE0025","description":"sensor 25 of moisture","type":"humidity","dataType":"number","unit":"%", "component":"METEO-1", "componentType":"meteo","componentDesc":"Test componente", "location":"41.39479 2.148768","timeZone":"CET" }]}' --header "Content-type: application/json" http://sentilo:8081/catalog/testApp_provider

In order to simulate a sensors network we wrote a Python script which publishes data to the platform on behalf of the registered sensor above:

import requests

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

```
import random
import time

url = 'http://localhost:8081/data/testApp_provider/RE0025'
headers = {'IDENTITY_KEY': 'identity_key'}

while True:
        data = '{"observations":[{"value":"' + str(random.randint(1,200)) +
'"}]}'
        r = requests.put(url, headers=headers, data=data)
        time.sleep(3)
```

We also added a subscriber which was listening for the nodes observations. The subscriber was written using Node.js:

```
my_http = require("http");
url = require("url");
var querystring = require('querystring');

function processPost(request, response, callback) {
   var queryData = "";
   if(typeof callback !== 'function') return null;

   if(request.method == 'POST') {
      request.on('data', function(data) {
         queryData += data;
         if(queryData.length > 1e6) {
            queryData = "";
            response.writeHead(413, {'Content-Type': 'text/plain'}).end();
            request.connection.destroy();
         }
      });

      request.on('end', function() {
         request.post = querystring.parse(queryData);
         callback();
      });
```


BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

```
    } else {
        response.writeHead(405, {'Content-Type': 'text/plain'});
        response.end();
    }
}
my_http.createServer(function(request,response){
                    if(request.method == 'POST') {
        processPost(request, response, function() {
            console.log(request.post);
            response.writeHead(200, "OK", {'Content-Type': 'text/plain'});
            response.end();
        });
    } else {
        response.writeHead(200, "OK", {'Content-Type': 'text/plain'});
        response.end();
    }
}).listen(88);
console.log("Server Running on 88");
```

The alarm was set to be triggered if the observations received from the sensor were bigger than 45. We subscribed both to observations and alarms:

```
curl --request PUT --header "IDENTITY_KEY: identity_key" --data
'{"endpoint":"http://subscriber:88"}'
http://senstilo:8081/subscribe/data/testApp_provider
curl --request PUT --header "IDENTITY_KEY: identity_key" --data
'{"endpoint":"http://subscriber:88"}'
http://sentilo:8081/subscribe/alarm/alarm_re0025
```

Data was transmitted over a normal public network. No VPNs were set up or any kind of tunneling between the sensors and the platform.

We focused mainly on the integrity of the messages sent from the sensors to Sentilo platform. In order to test if the data can be tampered with or modified we tried the following types of attack:

- Sniffing
- Man in the Middle attack
- Bad Data Injection attack
- Replay attack

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

Sensors send their observations in clear text. Any other value than integer is not recognized as valid by the platform. So encrypting the data is not possible. Also being a REST call, we had access to the headers of the message and we were able to find out the authentication token of the application/provider. This is also sent in clear text.
We successfully captured, replicated and modified the observations from the sensors. In this manner we were able to raise alarms that were not real. Neither the Alert agent nor the Pub/Sub Server noticed that the data was altered.

3.2 Mirantis OpenStack

OpenStack is a cloud computing platform which was started by NASA and Rackspace. Now it is managed by the OpenStack Foundation, a non-profit organization. It is an open source project for providing cloud computing mainly for public and private clouds. Being open source means that a community of developers worldwide is contributing to the development and enhancement of the platform every day.
OpenStack consist of a series of modules making it flexible and scalable. Any user can add its own particular module if needed or modify the existing ones. These modules and their functionality are described below:

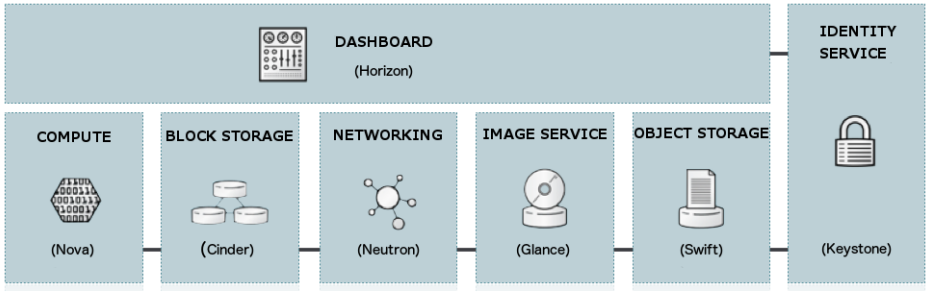


Figure 7 OpenStack architecture [9]

Nova – Compute module
This module provides services to support the management of virtual machine instances at scale, instances that host multi-tiered applications, dev/test environments, "Big Data" crunching Hadoop clusters, and/or high performance computing [9].
Security-wise the instances should be isolated and protected as well as the public endpoints and the communication between the components.

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

Cinder – Block Storage module
The Block Storage module is responsible of providing storage for the blocks of the compute instances. It is mainly important for tasks which depend on the speed of accessing data.
Encrypting the data both on storage and during communication, as well as providing strong authentication and authorization techniques should protect the confidentiality and integrity of the blocks and data.
Neutron – Networking module
Neutron provides various networking services such as IP address management, DNS, DHCP, load balancing, and security groups (network access rules, like firewall policies). It provides a framework for software defined networking (SDN) that allows for pluggable integration with various networking solutions [9].
From a security point of view this module includes techniques for protecting the confidentiality, integrity and availability.
Glance – Image Service module
This module's task is to manage (discover, register, deliver) the virtual machines images needed by the Compute module. The security issues are the same as for Nova.
Swift – Object Storage module
Swift is the module which stores the objects and files in the cloud. It has a native API but also an Amazon Web Services S3 specific API. Data is not stored only in one copy, but it's replicated in order to provide resilience.
Security-wise the Object Storage module has the same issues and tasks as the Block Storage module.
Keystone – Identity Service module
Keystone is a shared service that provides authentication and authorization services throughout the entire cloud infrastructure. The Identity service has pluggable support for multiple forms of authentication [9].
Security concerns here pertain to trust in authentication, management of authorization tokens, and secure communication [9].
Horizon – Dashboard
OpenStack comes with a web application for cloud administrators and tenants. From this application the cloud resources (computing instances, storage, security rules, images etc.) can be managed according to the role and authorization of the logged in user.

BDD-A23338 © 2015 Editura Universităţii „Petru Maior"
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

Security-wise OpenStack suggests three architectures to make the infrastructure safe and proof against attacks and intrusions:

- SSL/TLS proxy in front – in this configuration the SSL/TLS proxy is placed in front of the OpenStack environment. Communication is encrypted only until the OpenStack API endpoints, afterwards clear communication is used.
- SSL/TLS on same physical hosts as API endpoints – this architecture is similar with the previous model, the difference being that the SSL/TLS proxy is hosted on the same machine as the OpenStack API endpoints. The endpoints will be configured to listen only to local network interface, while remote calls will go through the SSL/TLS proxy.
- SSL/TLS over load balancer – this architecture is mainly useful for high availability or load balanced environments which need to inspect traffic. This can be achieved using the HAProxy which is able to pass the HTTPS traffic straight to the API endpoints.
- Cryptographic separation of external and internal environments – this architecture suits the cases where on the public network certificates are issued by a certain CA, but internally one might want to use their PKI to issue certificates for SSL/TLS. Subsequently, cryptographic separation can be accomplished by terminating SSL at the network boundary, then re-encrypting using the internally issued certificates. The traffic will be unencrypted for a brief period on the public facing SSL/TLS proxy, but it will never be transmitted over the network in the clear. [9]

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

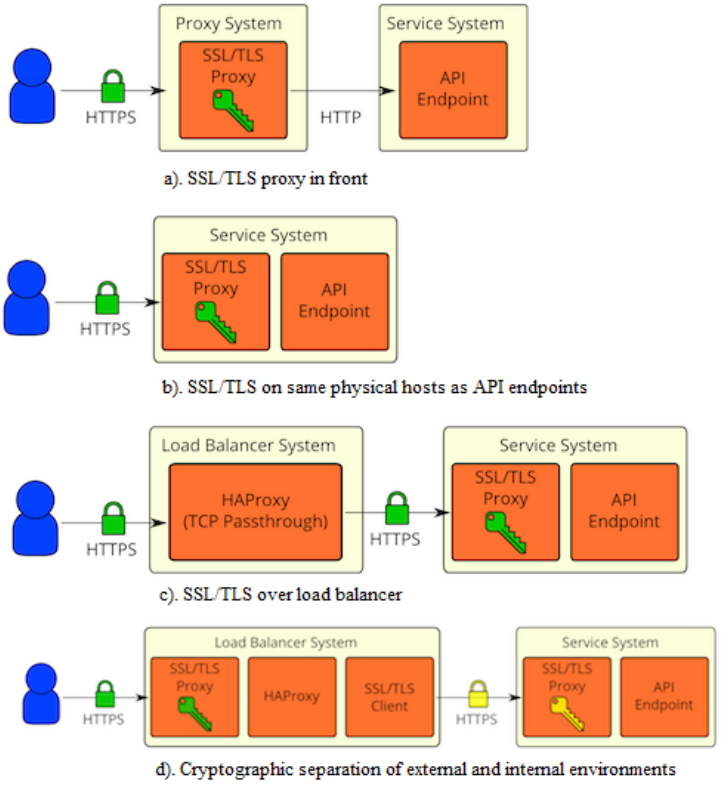


Figure 8 OpenStack secure architectures models [9]

Additional security measures that can be implemented on the network service, Neutron, include:

- VLANs with IEEE 802.1Q headers
- L2 tunneling
- Access Control Lists
- L3 routing and Network Address Translation (NAT)
- Quality of Service (QoS)
- Load balancing
- Firewalls [9]

We have chosen to install Mirantis, the most flexible and easy to use deployment of OpenStack. We have it running using VirtualBox on a Windows 7 host with this architecture:

- 1 master node
- 3 slave nodes

Afterwards, using the Fuel web interface we've configured Mirantis OpenStack environment and deployed it on the four VMs:

BDD-A23338 © 2015 Editura Universităţii „Petru Maior"
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

- Running Juno on Ubuntu 14.04.1
- QEMU hypervisor
- Neutron with VLAN segmentation
- Default security group

Using the Horizon Dashboard application we configured the images and instances for Nova service, as well as specific keypairs for security purposes. This keypairs are saved in .pem files and registered in the Nova database. All subsequent calls to the Nova instances will need to contain this keypair in order to be declared secure and processed. The key is encrypted using RSA:

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAy4xhtburmDEzJrJykg2bVaMI8FHN+Mtf9rym7N5/rN5OMkCh
V+kXx3h4LgUsMJNDtvNgCyvd9GltGGp0me4vsFhUvryjBHGy/RsRBDbaktk2fxU2
uYI05Y7SzTrE9Np2yrZW35/LWVtVHsm7Mrjwvv3UUN2IWU+GhQd6rIAiUeQ1yr/f
SEBzUK1b2OmLsBy2Zf5gdAOUg91qC6SBMoAQDMT8HUu3AEoLtnoPQZOcsDfEDG2a
yqvUGKpO8HWHgRCpdW5JYcugAZwHvzxphKzoOaMfh4Yh60sFn0986RHIKLwTzu1M
qLvahc4COeILteiSmQEEbg02wi0aMVMWKScZ2QIDAQABAoIBADyp3gUXpZB6OVBz
Joo42+6t5GAbQVPA6RzUJLu7pDmVF6EKeguFQL0GPmsYx47ClQ/VgO5TaJcLRKRe
NdoajsPzz235MmyEpl+gzXWAE6xoaEF/xQuMrs8rvc/EkCaZZYYMyd9j9GrxOXEv
lkcLr1R8ueZP3+8eMG09SWZX2eKlpMdDSWgSAF/Rgr0ikryBmAIffBhqujxt+jSy
DS3hd2MFRnNNjG5FsH97v1mgbON9nufIJVhebx8Lq9PP1QfAbVNDzDnOvyWBP3JP
Wyv4Rg7WFBK3WWFsxGDzmNua7LWFasvS0KegczZx7F6ttuyf3MWQPVX0Qu/RnNTN
F4ZQHXECgYEA8d12raKVt7fSO1nDyPD/skwP+swb2Hc31TA3d8pfcwkVryubUi4X
OSP/MHiUrgpSn9m+HMkC0tmVRFb/UnjjQvsJR7qVnAkJ9EtuzE2F6qr2JSgJZBz7

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

EX+fdSCRWs8+4lXbzf1ZMju71V1yvFuBtSxTHhx8PqGdLVUomdGozgcCgY
EA13Gp
poPOlj7QgNKKlEGL/y1IisZ8Rv3Opke8fb4gBSCrCOL1BiQCSp05Z4xyIiHZ/t
oL
VEScsPYsH7v2iXHMWB4A+5phQU6ZCUmQGRvYYEnj1Krv3pAASy6hK3
8N+bCzCAoc
tTa4pwurETlpoHqpAl3htjbpnHvvVV4N6KnO4R8CgYBU2Emfk59NuXBIXa7
uuIwa
MdCRPDSwdPHjGWz72sQtCIzQzE1KQNzosJX+nO4bN4fQh4PHeURCTw2
r0ZDzj3C5
uHKC9RMyj4Esb6HIjZFixuJeGnNg6UGx28FGR0x2PKlkoJXg5vT5SDcWHB
f5t2gC
9C+cKoXzOqJ2mp8JhqDe1QKBgHKpy+ETxZ+xTsdBRwAg4qGtOC6j9QDTl
GhrRaam
yePwvxa7tCzQfWe4xhSWayg/XAaHhgAThFGqs1EweMYuCtPbJCrEv35ClrC
atlam
u0KEEP3e/Es32PAqoRzFQmrh4Gcm+qB3v08opqNEKzN+FPVtgfO4xhzC2V5
V8JEj
zzeBAoGBAJRAXxUpTzQLV83kVG1X8neChyeWnGwIbnqCjs1AP9msE7tTfp
9yUvow
KTNrrN5QhVIFVo1yqnKxsg6ZfxUsdrHeMEJLM7hnlrG7cntlz/eZa4iAglYY5t
RW
fxB+oBuDH7f1YrQS+QsJWPSwmMwHeN4rqB3LLrHeKKk7VI+rU6Mf
-----END RSA PRIVATE KEY-----

Not storing the keypairs in clear text and making them required for each call ensures that the Nova service is receiving and processing only trustworthy requests. Also these keypairs can be configured to be edited only by the user that created them.

4. Discussions

From the studied articles and researches we can see there is an increasing interest in creating and developing platforms that integrate cyber-physical systems like sensor networks or Smart Grids with Cloud Computing. The main areas in which these systems can be used are:

- Agriculture
- Military
- Transportation

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

- Health Care
- Smart cities

The advantages that Cloud is bringing in such systems are the storage capacity and computational power. Data can be stored for real-time processing or historical purposes. Most of the studied platforms have the purpose of helping users to see their data in a more organized manner but most importantly help them make informed decisions in a short period of time.

Usually, the data that is being generated, sent through the communication channels and stored must have the following properties:

- Freshness
- Integrity
- Availability
- Non-repudiation
- Privacy

Security-wise, only a few of the studied frameworks implemented solutions to keep the data and system secured. From these solutions the most used ones are:

- Encryption
- Digital signatures
- Authentication and authorization systems for the users
- Third Party Auditors

From the two studied open source platforms, Sentilo is specifically targeted for sensor networks integration. Mirantis OpenStack is a more generic platform, which can be used for other types of applications as well. The characteristics of the two platforms are presented in the following table:

Characteristics	Sentilo	Mirantis OpenStack
Type	Open source	Open source
Target	Integrated cloud system for sensor networks	Cloud system with general purpose (cyber-physical systems, running time consuming tasks, cloud applications)
Architecture	Modular	Modular

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

Encryption	HMAC, SHA-512 (only for callback messages)	RSA (encrypted keypairs), data can also be encrypted
Certificates	Not specifically mentioned	Yes
SSL/TLS	Not specifically mentioned	Yes, also proposes different architectures to implement it

Table 1 Characteristics of Sentilo and Mirantis OpenStack platforms

Mirantis OpenStack is more security oriented than Sentilo, mostly because it has a wider range of applications. Both platforms have a modular architecture, each module having precise tasks to process. Having this kind of architecture both Sentilo and MirantisOpenStack support additional modules so a user/developer which is not satisfied of the functionality can add its own custom module or change the implementation of the existing ones.
Sentilo is used in real deployments in cities in Spain, most important being Barcelona. Mirantis OpenStack deployments are also used world-wide. Both platforms have proved to be reliable in real environments.

5. Conclusions

Cyber-physical systems have already been integrated with Cloud platforms. There have been researches and implementations of such frameworks in laboratories, test beds and in real life environments. Cloud is the most appropriate solution for storing the huge amount of data that the sensor networks, grids or other cyber-physical systems generate. Also, because of their computational power Clouds can process this information quickly and they offer high availability and resilience.
We have presented the current state of the research in cyber-physical Cloud integrated systems. From a security point of view most of the frameworks have chosen simple solutions or didn't implement any solution at all.
We also studied two open source platforms which can be used for developing Cloud integrated cyber-physical systems: Sentilo and Mirantis OpenStack. These have a stable and mature architecture which is suitable for supporting such systems. Security-wise OpenStack has more features than Sentilo.

BDD-A23338 © 2015 Editura Universităţii „Petru Maior”
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)

In the future we intend to add our customs security modules to these platforms to enhance the integrity of the data flow through the system and also test in depth their performance.

References

[1] Sajjad Hussain Shah, Asad Iqbal, Fazle Kabeer Khan, Wajid Ali, "A New Framework to Integrate Wireless Sensor Networks with Cloud Computing", Aerospace Conference, p. 1-6, 2013.

[2] Wen-Yaw Chung, Pei-Shan Yu, Chao-Jen Huang, "Cloud Computing System Based on Wireless Sensor Network", Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, pp. 877–880, 2013.

[3] Dinesha H. A., R. Monica, V.K. Agrawal, "Formal Modeling for Multi-Level Authentication in Sensor-Cloud Integration System", International Journal of Applied Information Systems, Volume 2– No.3, New York, USA, May 2012.

[4] Rajeev Piyare, Sun Park, Se Yeong Maeng, Sang Hyeok Park, Seung Chan Oh, Sang Gil Choi, Ho Su Choi, Seong Ro Lee, "Integrating Wireless Sensor Network into Cloud Services for Real-time Data Collection", International Conference on ICT Convergence (ICTC), pp. 752 – 756, 2013.

[5] Ahmed Lounis, Abdelkrim Hadjidj, Abdelmadjid Bouabdallah, Yacine Challal, "Secure and Scalable Cloud-based Architecture for e-Health Wireless Sensor Networks", International Conference on Computer Communication Networks (ICCCN), Munich, Germany, Jul 2012.

[6] Satyakshma Rawat, Richa Chowdhary, Dr. Abhay Bansal, "Data Integrity of Cloud Data Storages (CDSs) in Cloud", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 3, March 2013.

[7] Malcom Bain, "Sentilo - Sensor and Actuator Platform for smart Cities", JoinUp European Commission, May 2014.
(https://joinup.ec.europa.eu/community/eupl/document/sentilo-sensor-and-actuator-platform-smart-cities)

[8] Sentilo project page - http://www.sentilo.io

[9] OpenStack documentation - http://docs.openstack.org

BDD-A23338 © 2015 Editura Universităţii „Petru Maior"
Provided by Diacronia.ro for IP 216.73.217.42 (2026-11-16 04:53:11 UTC)