SeNsiM-SEC: secure sensor networks integration to monitor rail freight transport

Valentina Casola*, Alessandra De Benedictis, Annarita Drago and Nicola Mazzocca

Department of Electrical Engineering and Information Technology,
University of Naples Federico II,
via Claudio 21, Napoli, Italy
E-mail: casolav@unina.it
E-mail: alessandra.debenedictis@unina.it
E-mail: annarita.drago@unina.it
E-mail: nicola.mazzocca@unina.it
*Corresponding author

Abstract: In recent years, the interest in monitoring infrastructures has spread in many application domains, even because of the number of natural disasters and terrorist attacks. This important activity can be seen in the general context of critical infrastructure protection, such as the freight trains meant for hazardous materials transportation. The design of these systems must answer to several issues: low-cost, easiness of installation, interoperability of information sources, security requirements. The use of wireless sensor networks emerged in this field as a compliant solution to these issues. In this paper, we will present a monitoring system that uses heterogeneous WSN to monitor a freight train transporting hazardous materials. The sensors interact through a security platform in order to share different information. We illustrate some details on the architecture and the software application to prove the feasibility of such system on a real scenario, by discussing the most significant results about measurement parameters and networks performance.

Keywords: sensor networks integration; secure sensor networks; monitoring critical infrastructures; rail protection; data integrity.

Reference to this paper should be made as follows: Casola, V., De Benedictis A., Drago, A. and Mazzocca, N. (2013) 'SeNsiM-SEC: secure sensor networks integration to monitor rail freight transport', *Int. J. System of Systems Engineering*, Vol. 4, Nos. 3/4, pp.291–316.

Biographical notes: Valentina Casola is currently an Assistant Professor at the Department of Computer and System Engineering of the University of Naples Federico II, Italy. She received her MS in Electronic Engineering from the University of Naples Federico II with honours, in 2001. She received her PhD in Computer Engineering from the Second University of Naples in 2004. Her research activities include both theoretical and experimental issues, in the areas of security of information systems and policy-based approaches for security evaluation and management. Her activities are documented by many publications, in national and international journals and conference proceedings.

Alessandra De Benedictis received her PhD in Computer Science and Engineering at the University of Naples Federico II in April 2013. She received her BS and MS in Computer Engineering, both Cum Laude, from the University of Naples Federico II in 2006 and 2009, respectively. Her research

interests include security in ad hoc and wireless sensor networks, embedded systems, and performance evaluation. She has been working on the design of secure architectures for constrained devices, security evaluation, moving target defence mechanisms and localisation of malicious nodes in mobile networks.

Annarita Drago is currently a PhD student in Computer Science and Engineering at the University of Naples Federico II, her fellowship being granted by Ansaldo STS. She received her BS and MS in Computer Engineering from the University of Naples Federico II. Her research activities concern critical infrastructure protection (CIP) with particular reference to railway transport sector. She has been working on the design of integration architectures for interoperability of security systems for monitoring critical infrastructures, security evaluation of wireless sensor networks and their support for CIP.

Nicola Mazzocca is a Full Professor of High-Performance and Reliable Computing at the Department of Computer and System Engineering of the University of Naples Federico II, Italy. He received his MS in Electronic Engineering and PhD in Computer Engineering, both from the University of Naples Federico II. His research activities include methodologies and tools for design/analysis of distributed systems, secure and real-time systems and dedicated parallel architectures.

This paper is a revised and expanded version of a paper entitled ‘Securing freight trains for hazardous material transportation: a WSN-based monitoring system’ presented at DHSS2012 Conference, Wien, 19–21 September, 2012.

1 Introduction

Wireless sensor networks (WSNs) are widely used in several application domains, as environmental monitoring, detection and classification of objects in military and civil settings, critical infrastructure monitoring and protection, automotive, health monitoring and so on. Their decentralised and self-organising nature makes the deployment very easy and this facilitates their adoption in any context without requiring the existence of a supporting infrastructure.

The diffusion of sensor systems and their applications has led to a large heterogeneity in the logic for interfacing and collecting data from these systems. A typical monitoring system is composed of different sensing infrastructures (sensor networks), that can be heterogeneous in the technology aspects, in the data formats, and in synchronisation and localisation standards.

Often, in complex systems, heterogeneity resides also in security requirements and deployed security solutions. WSNs are widely adopted in critical scenarios, thus making security issues a fundamental concern: the wireless nature of WSNs communications in fact, makes it possible to wage different types of attacks ranging from passive eavesdropping to active interfering, but often classical security solutions can not be directly applied due to sensor nodes resource constraints, thus requiring new protocols and mechanisms tailored to sensor HW/SW features. Depending on the sensitivity of sensed data and exchanged messages, and on the specific features of the devices

belonging to each sensing infrastructure, the adopted security solutions could be different among the composing subnetworks.

A complex monitoring system should be able to access data originated by different sensing infrastructures by means of a framework able to hide the heterogeneity in terms of sensor, networking, middleware technologies and security solutions, and to provide a standard way to manage, query and interact with them, in order to gain a deeper knowledge of observed phenomena.

Several monitoring systems are available in the literature, typically tailored for specific domains and specific technologies, and usually not cost-less customisable for new scenarios. They do not easily integrate new technologies or different data models and, furthermore, they usually do not provide any mechanisms to meet security requirements as data integrity and confidentiality, that are primary requirements for any critical application domains. Interesting research activities related to integration techniques for heterogeneous sensor networks have taken place, but nowadays only few architectures have been proposed. Most of them try to define a common exchange mechanism among different sensor systems in order to facilitate the integration, and provide a software integration layer which allows different sensor systems to collaborate for the same purpose. Very often they are strongly related to ad hoc technologies and, sometimes, they lack of a real implementation.

Monitoring functionalities are fundamental to several critical infrastructures, such as the railway and transportation infrastructures, that have gone through rapid developments in several technological aspects in the last two decades. In the past, wired communication systems were used for signalling and data communication in the railway industry, while recently wireless communication systems have emerged as alternatives to substitute wired systems (Lynch and Loh, 2006; Li and Wu, 2007; Casas and Cruz, 2003; Chebrolu et al., 2008). Wireless sensing infrastructures can be used to monitor and protect critical assets within a railway infrastructure, in order to ensure reliable, safe and secure operations, but also to protect citizens from any natural or anthropological hazards (Flammini et al., 2010).

In this paper, we address monitoring in rail freight transport, and propose an integration framework for WSNs able to cope with two different aspects, namely:

1 interoperability and security issues of different sensor networks (in terms of technologies and security mechanisms)

2 enforcement of different security mechanisms to provide confidentiality, authentication and integrity of exchanged messages.

The feasibility and effectiveness of the proposed framework have been verified within the pShield Project (2010) (Casola et al., 2010), that gave us the opportunity to verify the application of a WSN deployment in a real scenario to protect a freight train. Our experimental activities in fact, were conducted on a freight car made available by the Italian Railway Authority (RFI/Trenitalia) at Roma Smistamento.

The remainder of this paper is structured as follows. In Section 2, we discuss the motivations and open issues that are behind the choice of adopting WSN in monitoring transportation infrastructures. Section 3 presents the data model, while in Section 4 we illustrate the architectural model of the proposed monitoring system, able to integrate different sensor networks with different security requirements. Section 5 describes the security layer, and Section 6 illustrates the case study by discussing some experimental

results gathered in a real scenario. Finally, in Section 7, some conclusions and future work will be drawn.

2 Motivation

In recent years, the transport by rail of dangerous goods has increased substantially and consequently the problem of their control and monitoring has become of utmost importance, especially if we consider the negative effects and damages that can be caused to people and environment by any accident.

Monitoring systems play a fundamental role in the protection of a freight car against both natural and intentional threats. Monitoring is aimed at detecting abnormal operations/environmental conditions on board of vehicles as well as threats of burglary. If some abnormal activity is detected by sensors, (e.g., very high temperature or out of range vibrations) their transmission units are activated and information about the observed phenomenon is sent to the control centre, that will take proper countermeasures according to the specific scenario. The measurement of parameters as speed, acceleration, vibration and inclination of the wagon could be used to establish if a vehicle is properly moving: through these data is possible to detect collisions and derailments and analyse the behaviour of the driver (also noting any breaches that may compromise the security of cargo, such as exceeding speed limits on the way). Temperature and humidity measurements can help monitor and ensure optimal conditions for the transported goods and/or to prevent the risk of fire. Furthermore, with the adoption of localisation tools, as a GPS receiver, it is possible to associate a set of coordinates to an event, and send this information for alarm data quality improvement.

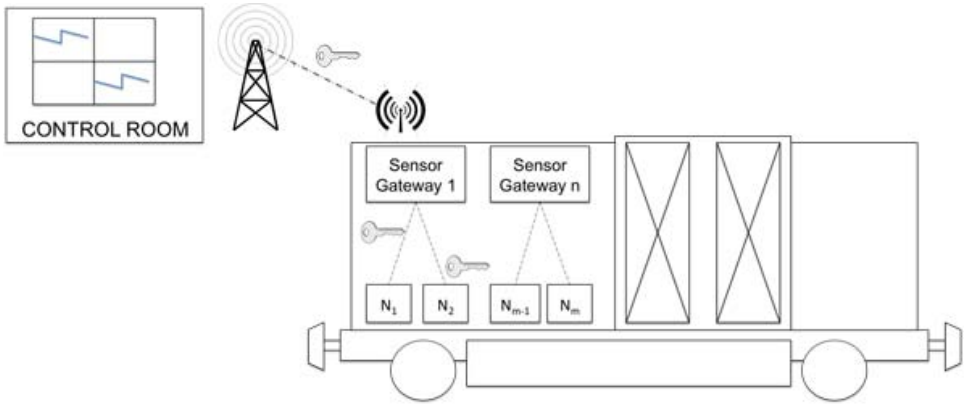
Critical parameters are mostly measured by pre-existing sensors already available and deployed, although sometimes new or just installed sensors are introduced to contribute to the observation of phenomena. In both cases, there is the need to collect and manage data coming from different and heterogeneous sensor technologies. Distributed applications in fact, require to collect information from different sources, so retrieved data are usually heterogeneous from many points of view (data structure, data format, semantic, protocols, sensing technologies) and need to be integrated to perform monitoring. Moreover, the heterogeneous sensing infrastructures composing a complex monitoring system are typically distributed in different points of the asset to protect (e.g., on board train and on the ground), and are interconnected by a communication network used to exchange sensing and control data.

Even if several standard solutions exist to handle heterogeneity in complex distributed systems, they can not be directly applied to the rail domain. The majority of freight cars in fact, are currently unpowered, so there is an increasing need for a power-aware and power-autonomous system architecture. Also, the railway structures are geographically distributed as mobile entities, thus requiring the capability to provide a connection to the central monitoring system through a wireless wide area network (WAN). To meet these requirements, monitoring functionalities should be provided by low-cost, easy to install and easy to maintain devices, and the system should be based on a small number of cheap components including wireless smart sensors that do not require connection cables, and are highly heterogeneous in terms not only of detection technologies but also of embedded computing power and communication facilities.

WSNs can be successfully used for monitoring purposes in rail freight transport. In WSNs, tiny sensors measure different parameters and send results to a gateway, periodically or on demand. The gateway forwards the results to a control centre for further processing and analysis according to a specific application.

In Figure 1, the main components that should be deployed to monitor a freight train are illustrated. In particular, we depicted different heterogeneous networks deployed inside a car to monitor several physical parameters. They send the retrieved data to a centralised control room that collects data and elaborates them according to a specific target application. With respect to this scenario, we focused our attention on heterogeneity and security issues to design a monitoring system based on WSNs.

Figure 1 The system view (see online version for colours)



As for interoperability issues, different middleware platforms (Hadim and Mohamed, 2006; Henricksen and Robinson, 2006; Römer, 2004; Amato et al., 2011) have been proposed in the literature to hide heterogeneity of sensor networks, in order to bridge the gap between the application and the underlying hardware and network platforms. As previously said, very often they are strongly related to ad hoc technologies and, sometimes, they lack of a real implementation. Also, they usually do not provide any mechanisms to meet security requirements as data integrity and confidentiality.

As for security issues, in the specific case of freight train monitoring, the main requirements to fulfil are the secure handling of the critical information on the transported material and the secure monitoring of the transport. Indeed, security plays a fundamental role in the development of monitoring applications, as data collected by sensors from the environment are often sensitive and should be accessed only by authorised entities, to prevent malicious users from intercepting them or sending corrupted data to compromise the monitoring activity. Moreover, real-time streams from smart-sensor alarms must be continuously available according to soft or hard real-time constraints. Securing a sensor network is critical, as communication among sensors is performed via a radio channel which is insecure by nature. Furthermore, in most cases, nodes are easily accessible, and could be reprogrammed, replaced or even destroyed.

Several attacks against WSNs exist and can be performed in many ways and at different levels (Padmavathi and Shanmugapriya, 2009). Attackers could aim at modifying transmitted data or at forging malicious data flows, or they could try to

eavesdrop on traffic in the network in order to extract important information and use it later in active attacks. In many cases, attackers are represented by laptop-level nodes, that have much better power supply, as well as larger computation and communication capabilities than a sensor node, and can cause big damages if not correctly handled. Unfortunately, due to the resource limitation (in terms of energy, memory, computation and communication capabilities) of the devices, security protocols and algorithms proposed for traditional ad hoc networks are not suited to small sensors (Ravi et al., 2004), and new approaches that try to balance security, performance and power consumption must be investigated.

In this paper, we address interoperability and security issues arising in a complex monitoring system composed of several heterogeneous sensing infrastructures. In particular, we propose a monitoring infrastructure that enables the management of heterogeneous networks with different security requirements, named SeNsiM-SEC (Casola et al., 2011a).

SeNsiM-SEC is an integration platform based on the sensor networks integration and management (SeNsiM) framework (Casola et al., 2009), a scalable software architecture for the integration of heterogeneous sensor systems. SeNsIM enables the deployment of applications based on multiple sensor systems by providing a standard way to manage, query, and interact with sensors. It is composed of two main layers: the sensor network layer for the sensing data and the distributed application layer for the management and elaboration of queries and data.

To face both interoperability and security problems, we developed:

- a *data model*, capable of representing with a unique logical view the heterogeneous 'sensor data'
- an *architectural model*, able to support in an efficient way the management of sensed data belonging to different networks
- a *suite of security protocols*, aimed at ensuring the security requirements as confidentiality, authentication and integrity of messages exchanged among the sensor nodes.

A detailed description of these components is reported in the following sections.

3 The data model

Data coming from different sources present different formats and semantics, making their management very complex. To solve such problems, it is necessary to define a data model that is capable of representing them in a unique logical view. In the literature, sensors have been modelled by using two kinds of approaches (The SensorML project, 2007; Park et al., 2000; Skov and Bro, 2005):

- *Structural* approach, which focuses the attention on the sensor structure in terms of hardware/software components (blocks model) or functional modules (layered model).
- *Data-oriented* approach, which represents a sensor according to a behavioural description in terms of its features and retrieved data [e.g., SensorML (The SensorML project, 2007)]. Hence, it is able to represent sensor global information

(type, producer, description, etc.…) as well as variables that a sensor can measure (temperature, light, humidity, etc.…), predicates that a sensor can calculate (e.g., temperature greater than a threshold), and sensor operating state/mode (on, off, sleep, etc.…).

Our approach aims at modelling sensor nodes and a sensor network by combining both structural and behavioural description of sensors. In the following, a formal description of our model is given along with basic concepts about sensors and sensor networks/systems.

3.1 Sensor node

A sensor node is an object characterised by a static or intensional part and a dynamic or extensional part.

The *static part* is represented by the following tuple:

$$\mathcal{S} = \langle M_s, I_s, S_s, V_s, P_s \rangle$$

where

- M_s is the set of time-invariant information of a sensor (e.g., ID, name, producer, type, model, description, latency time, accuracy, etc.)
- I_s is the set of time-variant information of a sensor (e.g., free ram, voltage, lost epochs, transmitted epochs, geographical position of a sensor, etc.)
- S_s is the set of the possible operating states of a sensor (e.g., on, off, sleep, normal, events working, standby, etc.)
- V_s is the set of physical variables that a sensor can measure (e.g., temperature, noise, pressure, etc.…)
- P_s is the set of predicates that a sensor is able to calculate on the measured variables (e.g., current value of noise, temperature is greater than a fixed threshold, pressure is in a fixed range, etc.…).

The *dynamic part* is defined by the following (sensing) function:

$$\mathcal{F}_s : S_s \times Oi_s^n \times t \rightarrow Op_s^m$$

where

- Oi_s is the set of possible values of sensor time variant information (e.g., 1 V, 1 Mb, etc.…)
- Op_s is the set of possible output ranges of sensor predicates (e.g., [–100°C, 100°C], [–22°F, 22°F], {True, False}, etc.…)
- t is the time (sampling time, life time) for which the sensing function has to be applied.

3.2 Sensor network

A sensor network is a collection of sensor nodes disposed according to clustering/grouping policies and to a given topology.

It is defined as

$$\mathcal{N} = \langle M_n, Mat_n, Los_n, Loc_n, P_n \rangle \quad (1)$$

where

- M_n is the set of metadata describing the network (e.g., ID, name, type, middleware if present, textual description, etc.…)
- Mat_n is the topology matrix, where each generic element MAT(i, j) indicates the kind of link (e.g., no-link, wireless parent-child link, normal cabled link, wireless ring link) between the sensors i and j of the network
- Los_n is the list of sensors of a network
- Loc_n is the list of clusters with the related sensors which a network can be subdivided by
- P_n is the set of predicates which can be retrieved by a network (e.g., pressure in a given group is greater than a threshold, current value of average temperature in the network, etc.).

The proposed data model is able to represent a sensor node as well as the whole network. The state of a sensor can be modified by means of classical getting/setting functions, while the measured variables can be accessed using the sensing function. A collection of sensor nodes, deployed according to clustering policies and to a given topology, forms a sensor network. According to our model, a network object has to include global information such as type of sensor system, middleware (if present), supported sensor board as well as information related to sensor components (list of sensors, possible list of clusters, topology matrix). Network global predicates (e.g., average temperature of the network) have been modelled too.

XML (eXtensible Markup Language) has been used to represent the data model, since it provides platform independence, interoperability and can be easily parsed: we defined three types of XML descriptors, namely *netstructXML*, *queryXML* and *resultXML*, that will be described in the next section. XML-based descriptors provide a unifying grammar by which systems can describe their abilities and define a standard language protocol with which the different entities in the framework can communicate. In addition to providing a means to manage heterogeneity between different data sources, the XML language with its descriptors makes it possible to easily enhance the system by introducing new management functionalities (e.g., support for policies, administration and configuration of sensor systems).

4 The architectural model

SeNsIM-SEC architectural model has been designed by exploiting the *wrapper-mediator* paradigm, a well-known technique to integrate data from heterogeneous sources

(Wiederhold, 1992), according to which a *mediator* component accesses different data sources by means of ad hoc connectors (*wrapper* components, one for each network).

In a typical working scenario, each wrapper explores and monitors the local sensor network and sends to the mediator a description of the related information according to a common data model. The mediator, in turn, organises such information and keeps a unique view of all systems in order to satisfy user or application queries.

The SeNsIM-SEC architectural model can be considered as structured into four logical layers:

1 The *application or user layer* allows a generic user to submit queries and elaborate the retrieved data; a generic application should also be able to access sensor data through specific system APIs. The system provides support for monitoring queries which return the corresponding responses in real-time as well as for event queries. Many context-aware applications (i.e., those for critical infrastructure protection – CIP) need to trigger adequate actions/countermeasures after that some events have been generated from sensor systems.

2 The *mediator layer* aims at classifying networks' features as well as at formatting and forwarding queries to specific wrappers; a DBMS is used to store data related to networks with their sensors, user queries and related results.

3 The *wrapper layer* extracts and manages information about the underlying network and its sensors; at this layer queries issued by the mediator are received and executed on the local system by using its API and the local query language. A DBMS is kept also in each wrapper, to store network/sensors information according to the data model, as well as queries with their related results, information and data.

4 The *secure sensor system layer* interacts with the wrapper component in order to extract network features and carry out the retrieval process. In the current implementation of this layer, data exchanged among sensors are protected using a hybrid cryptosystem based on ECC (Kapoor et al., 2008), aimed to ensure confidentiality, integrity ad authentication requirements.

Figure 2 shows an example of deployment for the described architecture: it is composed of a mediator component, accessible by an end-user via a GUI interface, and of three different wrappers, each managing a different WSN. In particular, each network is devoted to monitoring a subset of physical parameters, and adopts a different security solution to protect exchanged data. In the following, we firstly describe the two main components of the SeNsIM system: the wrapper and the mediator. Then we illustrate interactions taking place between the user, the mediator, a generic wrapper and the underlying network during two main usage scenarios.

As already said, each wrapper component works as an adapter between the mediator layer and the specific sensing platforms. A wrapper gathers the features of the underlying network and of its sensors (e.g., discovering the network topology with its clusters/groups of sensors, the state of single sensors, etc.…), and accesses sensor data by querying single sensors, clusters or the entire network. The mediator classifies sensor information sent by wrappers and provides the user or the application with a simple way for querying the networks.

Figure 2 SeNsIM-SEC for train monitoring (see online version for colours)

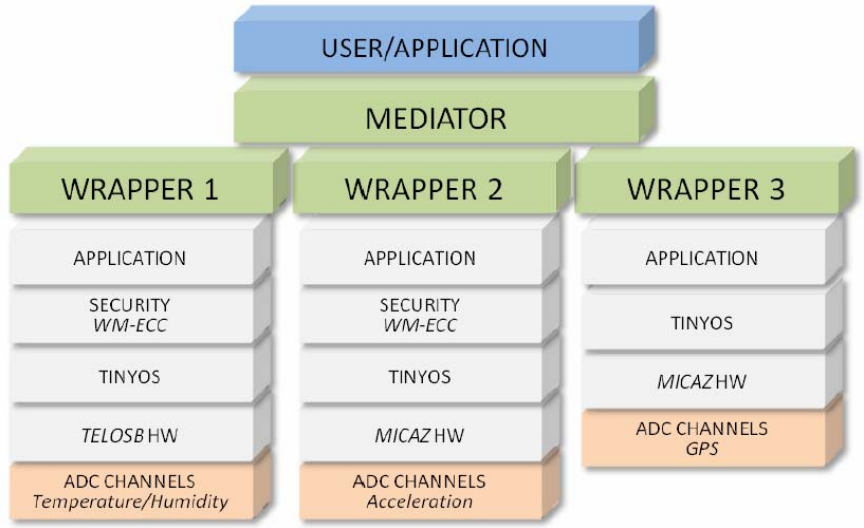
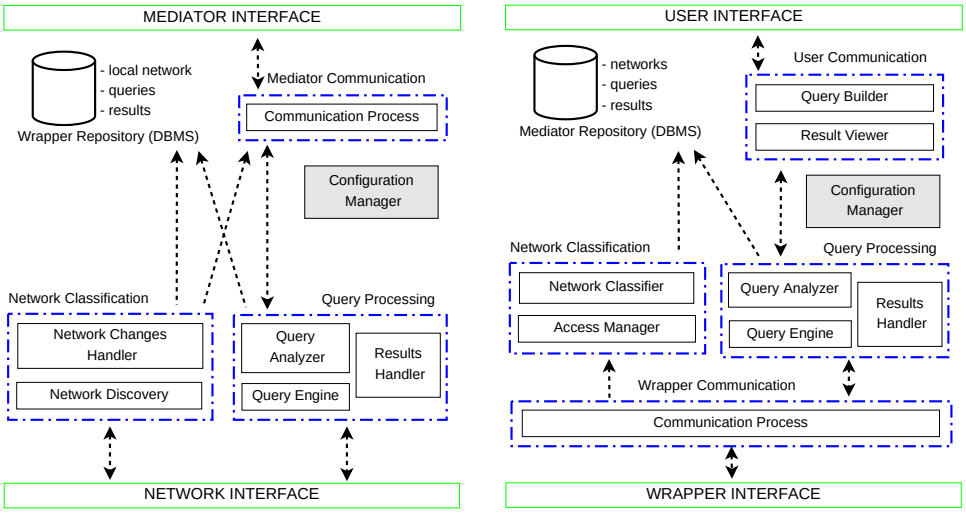


Figure 3 depicts the software architecture of wrapper and mediator components, illustrating their main modules. The macro-modules of both components, which are represented in dashed lines, carry out the main features of the related component.

Figure 3 (a) Wrapper architecture (b) Mediator architecture (see online version for colours)



Each wrapper component performs the following tasks:

- discovering, extracting and managing information of the underlying network with its sensors (*network classification* module)
- receiving user queries from the mediator and executing them on the system by using its APIs and the local query language (*query processing* module)
- managing the communication process with the mediator (*mediator communication* module).

Network classification and *query processing* modules interact with a local DBMS for storing and accessing information related to network/sensors (according to the data model), queries and related results. A wrapper is also provided with a *configuration manager* module, which the can be used by an administrator to set the state of sensors or define clustering/grouping policies.

The mediator, on the other side has to:

- classify and manage network/sensor information sent by wrapper (*network classification* module)
- manage user queries (*query processing* module)
- manage the communication process with wrappers (*wrapper communication* module)
- interact with the user, by taking his queries and showing him the related results (*user communication* module).

Even in this case, *network classification* and *query processing* modules interact with a local DBMS in order to store data related to networks and their sensors (static part of the data model), user queries and related results (dynamic part of the data model). Finally, the mediator is provided with a *configuration manager* module, which is used during the initialisation phase of the system to define the admissible information for a network and its sensors according to the data model.

4.1 Communication and query management

The communication between the mediator and the wrappers is carried out by means of XML files, written according to a standard format and containing information about the structure of the underlying networks, the user-defined query parameters and the retrieved results. As already said, we introduced three different XML descriptors, defined as follows.

The *netstructXML* has been directly derived from the data model and represents features of both networks and sensors. Each wrapper builds a netstructXML descriptor after having injected a discovery query on the underlying system (to get information about the sensor network). If some parameters could not been extract in an automatic manner (i.e., sensor producer, middleware for WSN), a wrapper administrator can manually fill the descriptor with the missing information.

The *queryXML* and *resultXML* descriptors represent user requests over remote data sources and the related responses. At mediator side, the queryXML descriptor is built after the user has sent the query through the mediator interface; at wrapper side, the resultXML descriptor is built after the wrapper has received sensor values.

We also introduced a *retrieval interval*, representing the time interval during which a wrapper collects query results before sending them back to the mediator in a single resultXML file. This results in a reduction of the communication overhead between wrapper and mediator components, and is specified in the queryXML file when generating a query.

The *netstructXML* contains structural information about the network and its composing nodes (e.g., sensor producer, middleware, physical channels, etc.), while the *queryXML* and *resultXML* descriptors represent user requests over remote data sources and the related responses.

Figures 4 and 5 illustrate the pseudo-sequence diagrams of interactions taking place among main system components throughout the two main usage scenarios, related to the registration and querying processes.

- *Registration*. Once the application has been deployed, the mediator keeps listening for incoming connections on a UDP socket bound to a specific port (this information, along with the IP address of the mediator machine is specified in a configuration file which is read by the wrapper component at its startup). When a new network is deployed, its wrapper must register with the mediator by sending a connection request. Before doing it, the wrapper performs a discovery process on the underlying sensor system (by injecting a specific discovery query), and builds the related XML descriptor (netstructXML); after that, it sends a registration request message to the mediator. When receiving a connection request, the mediator verifies the possibility of including a new system in the framework, chooses a free port and communicates it to the wrapper in a datagram packet. The wrapper uses such port as the remote TCP port to send, via a TCP communication, the netstructXML file containing the specification of the connected network. The mediator, upon reception of such file, stores the information about the new network in a local database.
- *Querying*. The querying process starts when a user sends a query request through the mediator user interface after having selected the destination of the query (a network, a cluster if possible, or a specific sensor). The mediator takes the query parameters, builds the related XML descriptor (queryXML) and sends it to the appropriate wrapper. The latter extracts the query parameters by parsing the XML descriptor and executes the query on the local system. The query results are grouped by the wrapper, which builds the result XML descriptor and periodically sends it to the mediator. Finally, the mediator extracts the results and shows them to the user.

Figure 4 Communication protocol in the registration scenarios

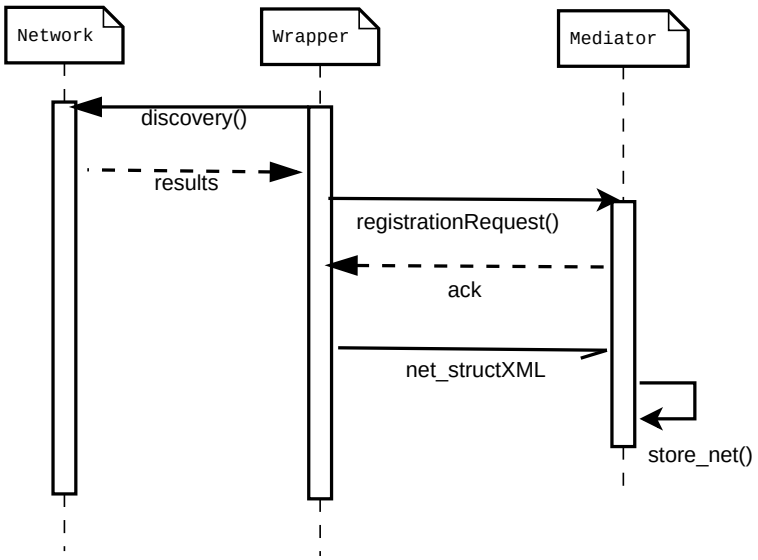
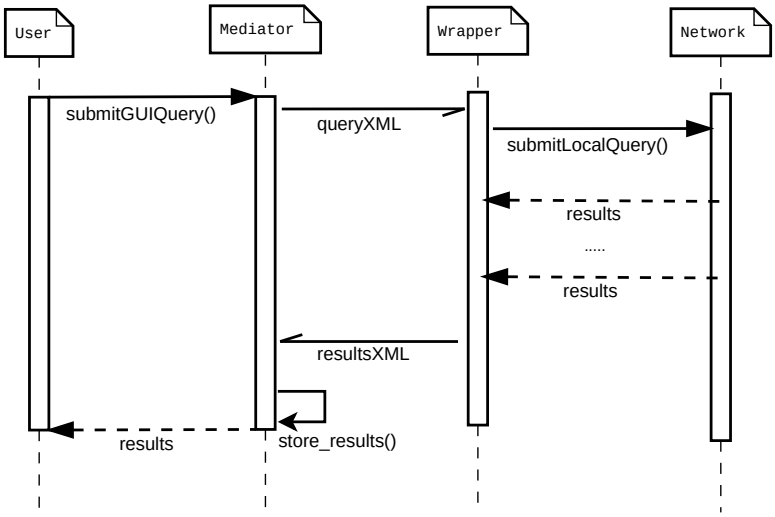


Figure 5 Communication protocol in the querying scenarios



The system provides support for monitoring queries that retrieve the requested data from the sensor systems and return the corresponding responses in real-time as well as for event-based queries. Each query message is characterised by the following information:

- *query identifier*
- *destination*, indicated as the requested *data source identifier* (sensor system/cluster/sensor)
- *type*, monitoring query or, when possible, event-based query

- *temporal parameters*, such as sample time, duration and results retrieval interval (see Section 4.1 for details)
- *destination*, a specific sensor, a cluster of sensors or the whole sensor system
- *sensing functions*, such light, temperature, acceleration, etc., among those allowed by the specified data source.

A result message is characterised by:

- *timestamp*, a discrete value which counts samples of each sensor
- *query identifier*
- *data source identifier*
- *output values* of the requested sensing functions.

5 Security protocols

As previously discussed, the wireless nature of WSNs communications makes it possible to wage different types of attacks ranging from passive eavesdropping to active interfering. Due to the specific HW/SW features of sensor nodes, the application of classical security approaches is often unfeasible: most security protocols are based on cryptographic operations, which massively involve the adoption of keys and complex mathematical functions that require dedicated computational resources and turn out to be critical from a performance and power consumption point of view. Sensor nodes are characterised by constrained processing and storage capabilities and limited energy resources, thus making these solutions not practical in real applications.

The security of a cryptographic system relies mainly on the secrecy of the key it uses. So, the main problem to face with when setting up a secure communication between nodes is the way cryptographic keys are established at each node. There are two main well-known mechanisms to handle this problem: in symmetric key cryptography (SKC) a unique secret shared key is used for both encrypting and decrypting messages, while in public key cryptography (PKC) each node manages a couple of keys.

Until a few years ago, the less resource-consuming symmetric schemes were adopted. This choice was dictated by the impossibility to use asymmetric ones (i.e., RSA) (Rivest et al., 1978) as they are power consuming and require a large amount of computational and storage resources. Recent studies have shown that it is possible to implement PKC on sensor networks by exploiting the primitives offered by the elliptic curve cryptography (ECC) (Kapoor et al., 2008). The strength of this scheme is in offering a security level equivalent to that provided by classical asymmetric schemes, by adopting smaller keys and simpler computations, thereby reducing processing and communication overhead – for example, it has been shown that ECC with 160 bits key provides the same security level compared to RSA with 1,024 bits.

In previous work (Casola et al., 2011b; De Benedictis et al., 2010), we investigated the adoption of new security mechanisms within a WSN, and proposed a hybrid approach to combine symmetric and asymmetric cryptographic schemes to benefit of both the security provided by asymmetric protocols and the better performance of symmetric ones. In Casola et al. (2011a), in particular, we analysed and compared two different security

libraries both based on ECC, namely TinyPairing and WM-ECC, from the security and performance points of view, analysing the overhead introduced by cryptographic operations and discussing constraints that should be taken in consideration before designing and deploying any WSN. Our analysis highlighted that, by adopting the WM-ECC library (Mary et al., 2007), it is possible to achieve an acceptable level of security without sacrificing performance that is a fundamental requirement in the critical rail transport scenario. For this reason, we chose this library to build a cryptosystem able to ensure confidentiality, authentication and integrity of messages exchanged among nodes belonging to the monitoring infrastructure deployed on the freight car.

WM-ECC is a public available open source implementation of a 160-bit ECC cryptosystem targeted to MICAz, TelosB and Tmote Sky platforms, based on recommended 160-bit standards for efficient cryptography group (SECG) elliptic curve parameters. The WM-ECC library provides all the ECC operations and some of them are optimised to give the best possible performance; it also provides an implementation of elliptic curve digital signature algorithm (ECDSA) protocol but it does not support any key exchange protocol. We aided the application running on nodes with an implementation of the elliptic curve Diffie-Hellman (ECDH) protocol that allows to establish a unique shared secret that is used as a symmetric key between the master and the motes for encrypting and decrypting the messages. The encryption and decryption operations are performed by means of the Skipjack cipher, with 80 bit keys and 64 bit blocks.

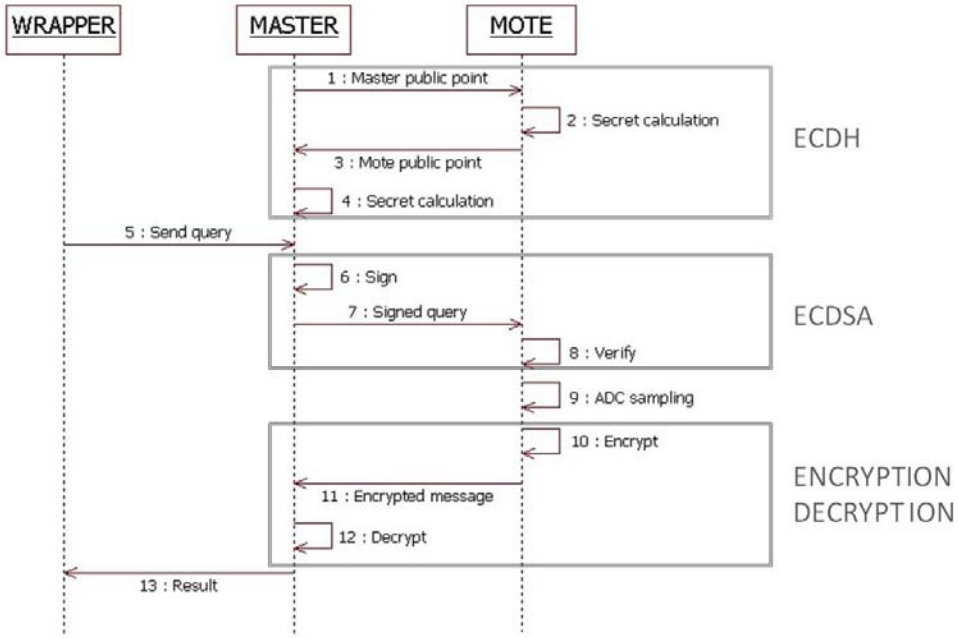
Figure 6 illustrates the sequence diagram describing the secure monitoring application that we developed based on WM-ECC. As shown, it is possible to identify three main phases, namely the initialisation through a key agreement protocol, the querying of sensor motes, and the transmission of encrypted samples:

1 *ECDH phase*. In the first phase, the master and mote nodes exchange their public points to calculate the shared secret key through the primitives provided by ECDH protocol.

2 *ECDSA phase*. At the arrival of a query, the master node constructs a query message with the received parameters, digitally signs it and then broadcasts it to the mote via radio channel; when receiving a query message, the mote verifies the digital signature and starts the sampling of the required physical values, according to the query parameters, only if the verify procedure is successful, otherwise it discards the message.

3 *Encrypt/decrypt phase*. When the results are ready, the mote inserts them into the payload of the response message, which is encrypted with the shared key obtained in the ECDH phase and finally it sends the message to the master; at the arrival of the message, the last extracts the payload, decrypt it with shared key obtained at the first phase and then returns the query results.

Figure 6 Secure communication protocol (see online version for colours)



6 The experimental case study

Figure 2 shows the developed architecture for train monitoring: it is composed of a mediator component, accessible by an end-user via a GUI interface, and of three different wrappers, each managing a different WSN. In particular, each network is devoted to monitoring a subset of physical parameters, and adopts a different security solution to protect exchanged data.

The mediator and the monitoring application were installed in the control room, located 30 metres far from the stationary train position. The wrappers ran on a different laptop located on the car, and connected to the mediator via WiFi over an SSL connection. The monitoring architecture was installed on a freight car made available by the Italian Railway Authority (RFI/Trenitalia) at Roma Smistamento, shown in Figure 7.

Figure 7 The car: outside and inside (see online version for colours)

Figure 8 Deployment

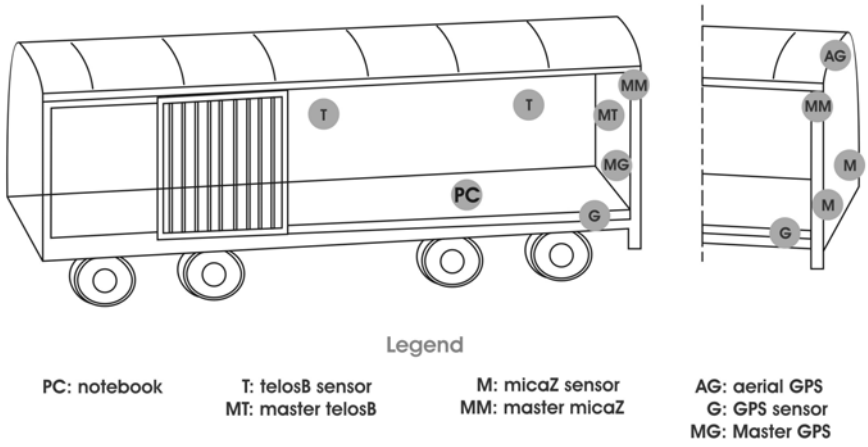


Figure 8 shows the deployment of the three sensor networks on the freight car. It is composed of:

1 A TelosB network (TelosB Datasheet, 2013), inside the car, with humidity and temperature sensors (Figure 9).

2 A MicaZ network (MicaZ Datasheet, 2013), outside the car (Figure 10), with acceleration sensors. The outside motes were equipped with a box in order to protect them from bad weather conditions.

3 A MicaZ network with a GPS receiver, installed outside too.

Figure 9 TelosB network (see online version for colours)

Figure 10 MicaZ network master outside the car (see online version for colours)

We developed two different applications, respectively for the master and the mote side of each network. The master node is represented by the gateway or base station, while a mote is a simple sensor device able to sample physical parameters and communicate with the gateway via radio. The applications installed on network nodes were designed to implement a WM-ECC-based cryptosystem: the master application was configured in order to digitally sign outgoing query packets addressed to the motes and decrypt the incoming response packets before sending the results to the wrapper, while the mote application was configured to participate to the ECDH protocol initiated by the master, verify the digital signature of the incoming query packets, and encrypt all outgoing response packets.

6.1 Some experimental results

In order to test the architecture and demonstrate the functional and security features, different test cases were conduced; we evaluated the parameters sensed by the networks (temperature, humidity, acceleration and GPS coordinates) and measured the packet loss rate on different nodes in two different working conditions:

1 Test 1 – train standing in the station

2 Test 2 – train running.

In the following, we will illustrate some results of these evaluations, focusing in particular on the queries performed on the TelosB network. We want to underline that the goal of this experimental phase was to evaluate the feasibility of the proposed system (WSN hardware and software for the monitoring) and not properly the parameters and values sensed by the different sensors; nevertheless, we will report some of these results, too.

6.1.1 Test 1: train standing in the station

The first test was conduced when the car was standing in the station in order to verify and evaluate the reliability of the connection among nodes; we also evaluated some parameters like temperature and humidity. Assume that the TelosB network has two motes with ID 4 and 5, respectively. For the first test, we decided to send a query of 5 minutes long (lifetime) with a sample period 0.5 seconds and a retrieval interval of 10 seconds: the wrapper was asked to collect samples coming from the underlying network for 10 seconds, and to send them back to the mediator in a resultXML file at the end of this time period.

Figure 11 shows the sampled values during the query for temperature and humidity respectively. Each value of the X axis represents a result file received by the mediator, containing samples collected over 10 seconds of monitoring (retrieval); the corresponding value on the Y axis is the mean value calculated over such samples for each result file.

In Table 1 instead, we report the mean values and their standard deviation for the whole query lifetime and for each sensor.

Table 1 Sensors mean values

Sensor node	*Mean*	*Standard deviation*
Temperature:Node4	19.5 C	0.6
Temperature:Node5	18.06 C	0.8
Humidity:Node4	63.4%	10.9
Humidity:Node5	61.4%	4.4

By analysing the received result files, we can count the number of received samples and easily evaluate the samples loss rate. According to the query lifetime, the sample period and the retrieval, the mediator should receive 30 result files from network, where the expected number of samples in each of them was 40. In Figure 12, the number of received packets against the expected ones for each node is presented; from this information, we can evaluate the loss rate of different nodes in the network, shown in Figure 13.

In Table 2, the mean number of received samples for each node and for the whole network is reported, evaluated with respect to the expected number of samples.

Figure 11 TelosB Network results, (a) temperature and (b) humidity mean values per result file

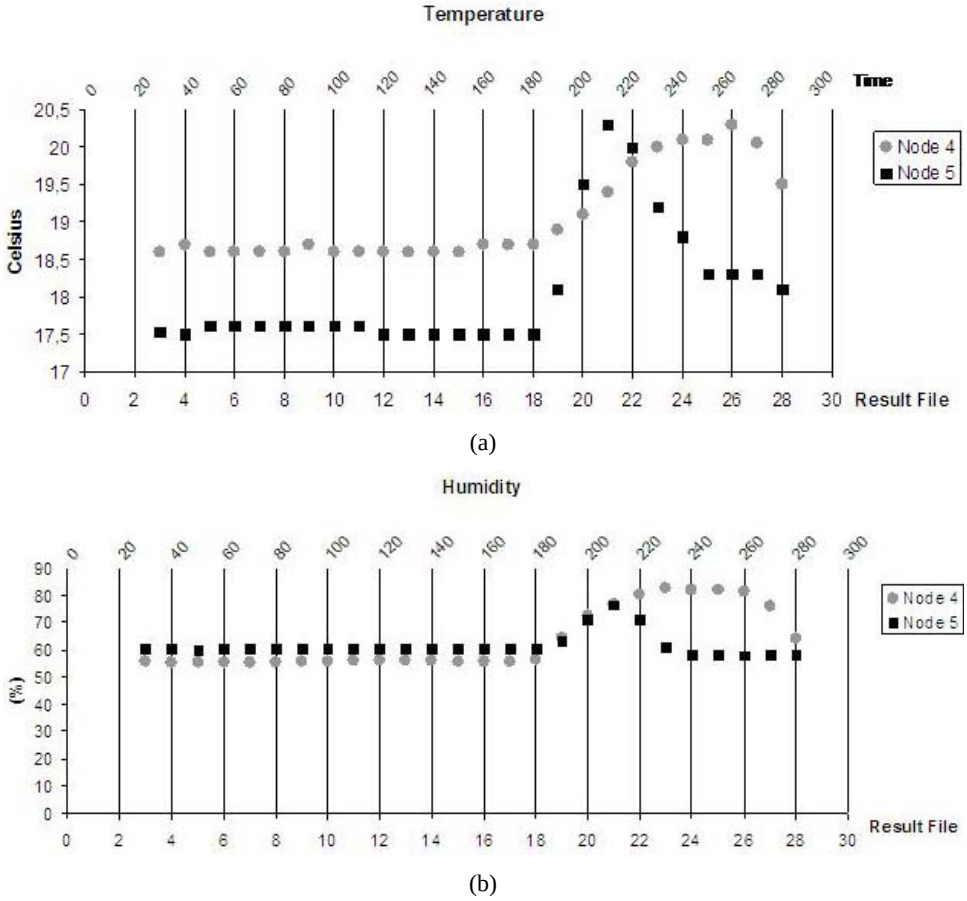


Table 2 Samples loss

TelosB	*Mean*
Node4	18
Node4LossRate	9%
Node5	19.4
Node5LossRate	3%
NetLossRate	6%

Figure 12 Number of received samples for each node in TelosB network

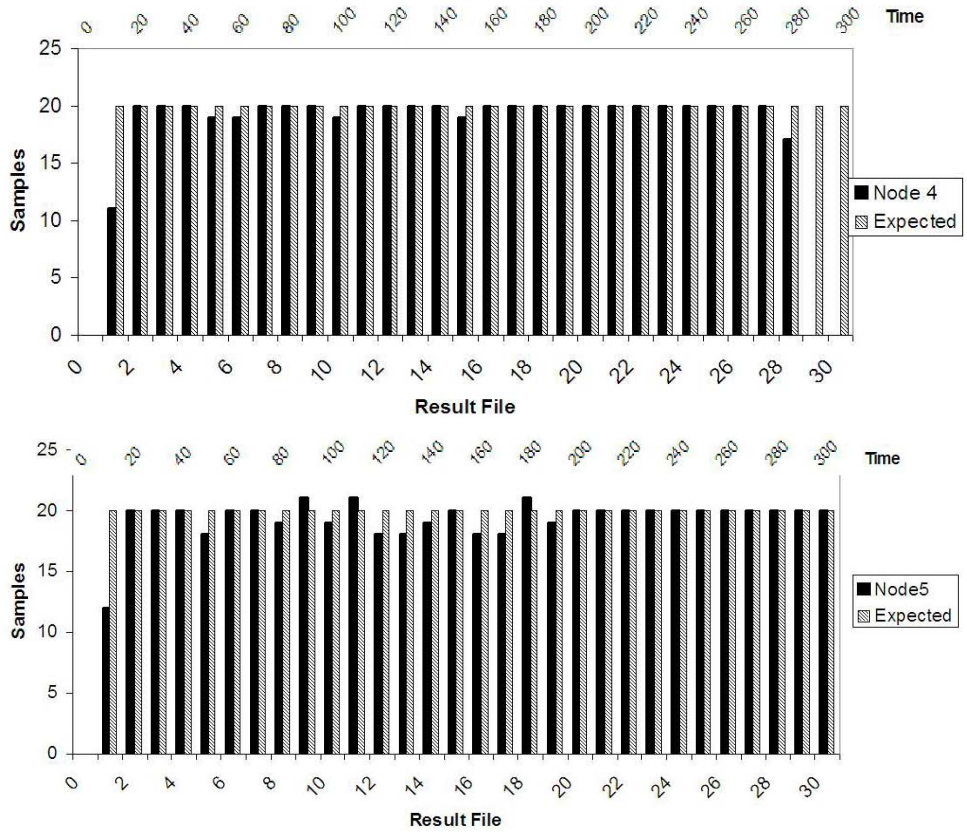


Figure 13 Samples loss rate – TelosB

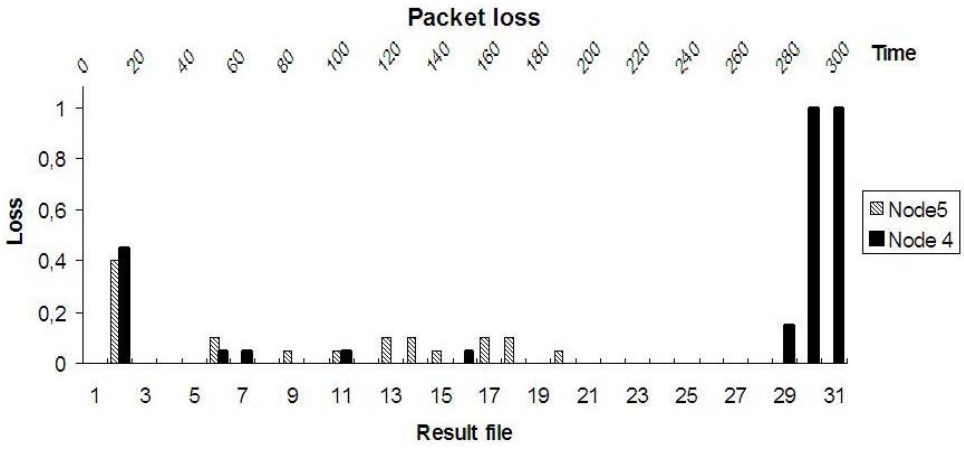
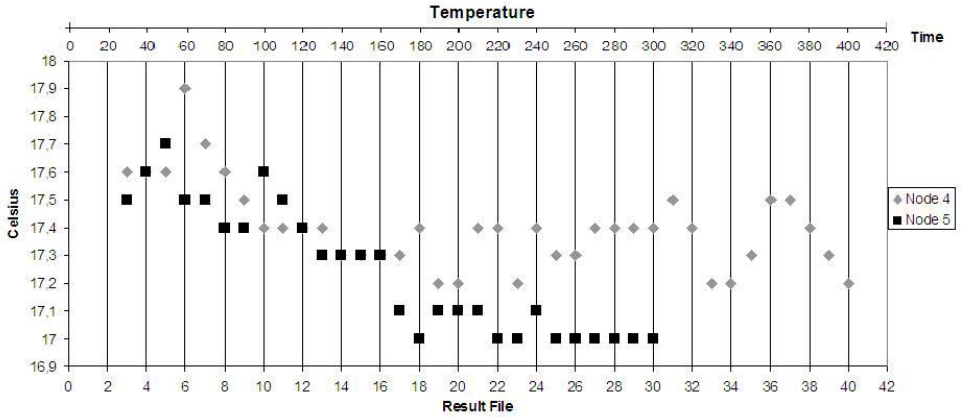
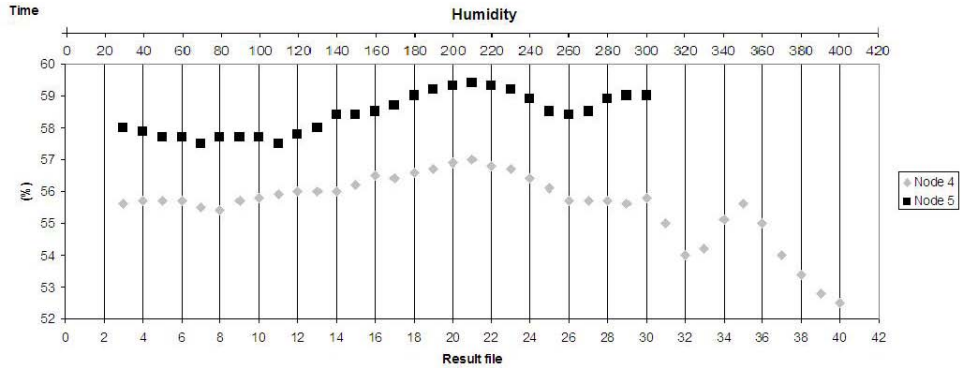


Figure 14 Car in movement – TelosB, (a) temperature (b) humidity



(a)



(b)

During the experiment, we decided to stop node 4 operation: as illustrated in Figure 12, the node looses all samples in the last two files. Moreover, node 5 shows an oversampling in some intervals, due to the way SeNSiM aggregates results (e.g., at result file 9, 12 and 18). By the way, both nodes present a similar samples loss at the beginning, due to the verification of signature in the ECDSA protocol (result file 1).

6.1.2 Test 2: train running

The second test was conducted while the car was moving, in order to test the connection between nodes and evaluate the measured parameters in a real-time condition. For this test, we sent a query of 7 minutes long (lifetime) with a sample period of 1 second for both networks and 10 seconds of retrieval time. Figure 14 shows the sampled values during the query for each network.

Similarly to the previous discussed case, during its operation, node 5 stopped working at a certain point. This situation, clearly visible in Figure 14, was caused by a not-well closed door that abruptly opened and cut off the node.

In Table 3, we reported the mean value and standard deviation of parameters for each network.

As previously illustrated, we can evaluate the samples loss rate by counting the number of received samples in the received result file. According to the query lifetime, the sample period and the retrieval, the mediator should receive 42 result files, each supposed to contain 20 samples. The actual number of received samples against the expected one is shown in Figure 15. The corresponding packet loss rate for the different nodes in the network is then shown in Figure 16.

Table 3 Sensor mean values for the car in movement

Sensor node	*Mean*	*Standard deviation*
Temperature:Node4	17.4 C	0.1
Temperature:Node5	17.2 C	0.1
Humidity:Node4	55.6%	1.1
Humidity:Node5	58.3%	0.5

Figure 15 Number of received samples for each node – TelosB

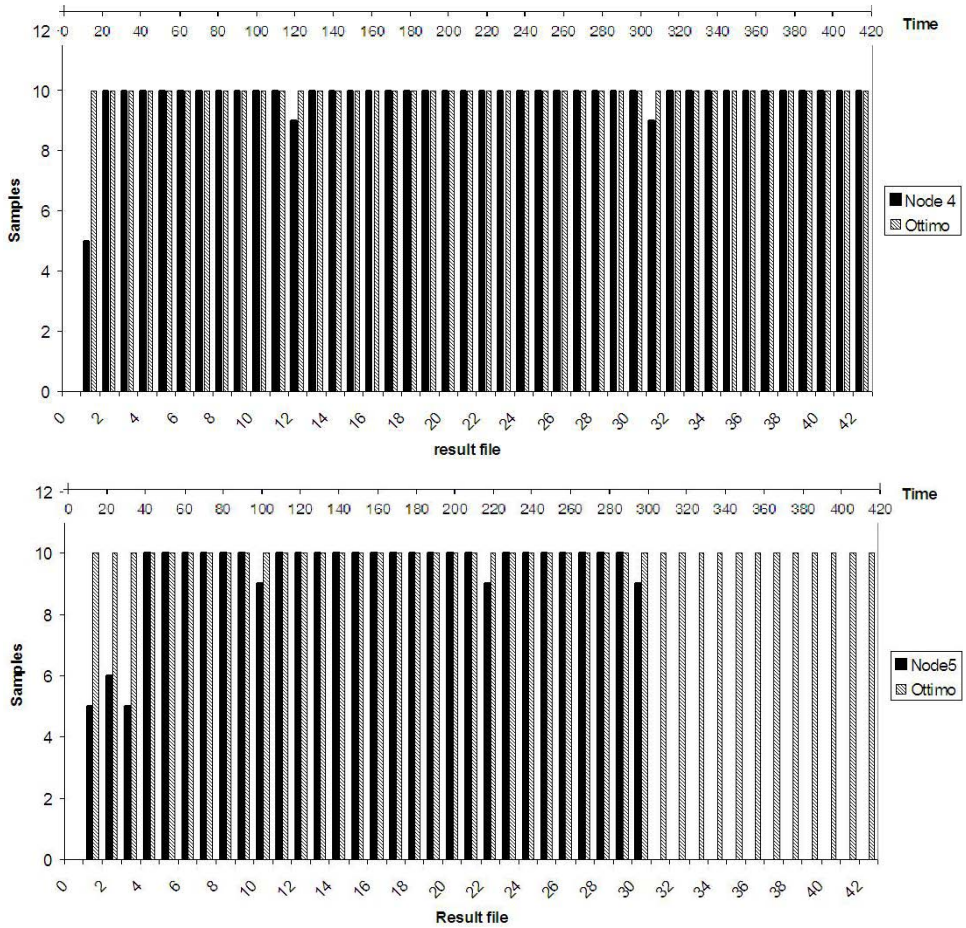
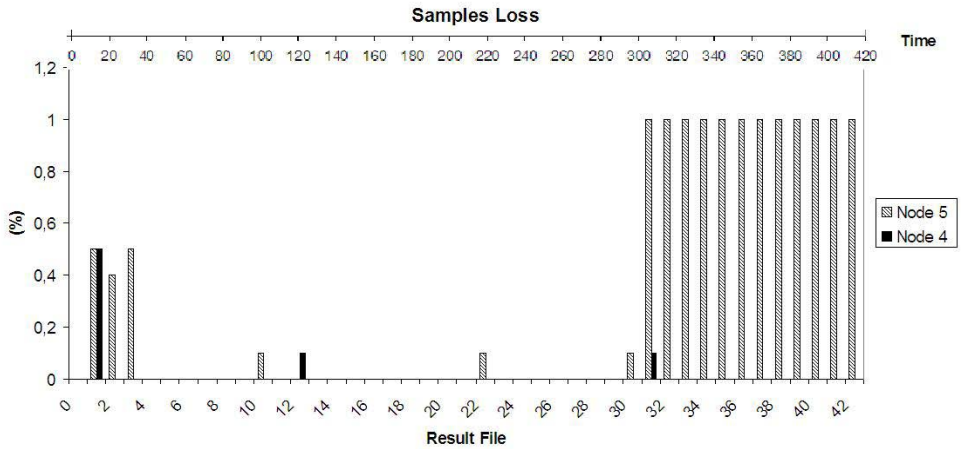


Figure 16 Samples loss rate – TelosB



Finally, in Table 4, we reported the mean value and standard deviation of parameters for the network under examination. As the table shows, in this test the network has a good behaviour with a low rate samples loss. Samples are lost only at the beginning, due to the execution of the ECDSA protocol.

Table 4 Samples loss

TelosB	*Mean*
Node4	9.7
Node4LossRate	2%
Node5	6.6
Node5LossRate	33%
NetLossRate	17%

7 Conclusions and future work

In this paper, we proposed an integration framework for WSNs adopted for rail freight transport monitoring. The framework, named SeNsiM-SEC, was designed to hide the heterogeneity of the different sensing infrastructures composing a typical complex monitoring systems, by providing a standard way to manage, query, and interact with their sensor devices. To meet the imposed security requirements for the critical rail infrastructure, the framework includes the adoption of proper cryptosystems to secure the communication among nodes and allows the management of different security solutions for different networks.

The feasibility and effectiveness of the proposed framework have been verified by conducting an experimental activity on a freight car made available by the Italian Railway Authority (RFI/Trenitalia) at Roma Smistamento. We verified that the introduction of security mechanisms does not affect the accuracy of measurements, even if it introduces a small delay in the monitoring activity due to the execution of cryptographic operations.

As a part of the experimental activity, we conducted an analysis on network performance, by measuring the packet loss rate on different nodes in two different working conditions, that is:

1 train standing in the station

2 train running.

The analysis showed that, even in running condition, the adoption of WSNs is feasible on trains.

These results motivated our activity and, in next future, we intend to propose more sophisticated monitoring applications based not only on threshold definitions but also on the implementation of decision support systems integrated with available train safety systems.

References

Amato, F., Casola, V., Gaglione, A. and Mazzeo, A. (2011) 'A semantic enriched data model for sensor network interoperability', *Simulation Modelling Practice and Theory*, Vol. 19, No. 8, pp.1745–1757.

Casas, J.R. and Cruz, P.J.S. (2003) 'Fiber optic sensors for bridge monitoring', *Journal of Bridge Engineering*, Vol. 8, No. 6, pp.362–373.

Casola, V., De Benedictis, A., Drago, A. and Mazzocca, N. (2011a) 'Analysis and comparison of security protocols in wireless sensor networks', in *2011 30th IEEE Symposium on Reliable Distributed Systems Workshops (SRDSW)*, IEEE, pp.52–56.

Casola, V., De Benedictis, A., Mazzeo, A. and Mazzocca, N. (2011b) 'SeNsiM-SEC: security in heterogeneous sensor networks', in *2011 Conference on Network and Information Systems Security (SAR-SSI)*, IEEE, pp.1–8.

Casola, V., Esposito, M., Mazzocca, N. and Flammini, F. (2010) 'Freight train monitoring: a case-study for the pshield project', in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pp.597–602.

Casola, V., Gaglione, A. and Mazzeo, A. (2009) 'A reference architecture for sensor networks integration and management', in *Proceedings of the 3rd International Conference on GeoSensor Networks, GSN'09*, Springer-Verlag, Berlin, Heidelberg, pp.158–168.

Chebrolu, K., Raman, B., Mishra, N., Valiveti, P.K. and Kumar, R. (2008) 'Brimon: a sensor network system for railway bridge monitoring', in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, ACM, New York, NY, USA, pp.2–14.

De Benedictis, A., Gaglione, A. and Mazzocca, N. (2010) 'Securing a tiered re-taskable sensing system', *6th International Conference on Information Assurance and Security*.

Flammini, F., Gaglione, A., Ottello, F., Pappalardo, A., Pragliola, C. and Tedesco, A. (2010) 'Towards wireless sensor networks for railway infrastructure monitoring', in *Electrical Systems for Aircraft, Railway and Ship Propulsion (ESARS)*, pp.1–6.

Hadim, S. and Mohamed, N. (2006) 'Middleware for wireless sensor networks: a survey', in *First International Conference on Communication System Software and Middleware, 2006, Comsware 2006*, pp.1–7.

Henricksen, K. and Robinson, R. (2006) 'A survey of middleware for sensor networks: state-of-the-art and future directions', in *Proceedings of the International Workshop on Middleware for Sensor Networks, MidSens '06*, ACM, New York, NY, USA, pp.60–65.

Kapoor, V., Sonny, V. and Abraham Singh, R. (2008) 'Elliptic curve cryptography', *ACM Ubiquity*, Vol. 9, No. 20, pp.20–26.

Li, S. and Wu, Z. (2007) 'Development of distributed long-gage fiber optic sensing system for structural health monitoring', *Structural Health Monitoring*, Vol. 6, No. 2, pp.133–143.

Lynch, J.P. and Loh, K.J. (2006) 'A summary review of wireless sensors and sensor networks for structural health monitoring', *The Shock and Vibration Digest*, Vol. 38, No. 2, p.91.

Mary, W., Wang, H., Tan, C.C. and Li, Q. (2007) 'WM-ECC: an elliptic curve cryptography suite on sensor motes', Technical report, College of William & Mary.

MicaZ Datasheet (2013) [online] http://www.datasheetarchive.com.

OpenGIS® Sensor Model Language (SensorML) (2007) *Implementation Specification* [online] http://www.opengeospatial.org/standards/sensorml.

Padmavathi, G. and Shanmugapriya, D. (2009) 'A survey of attacks, security mechanisms and challenges in wireless sensor networks', *CoRR*, abs/0909.0576.

Park, S., Savvides, A. and Srivastava, M.B. (2000) 'Sensorsim: a simulation framework for sensor networks', in *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, *MSWIM '00*, ACM, New York, NY, USA, pp.104–111.

Ravi, S., Raghunathan, A., Kocher, P. and Hattangady, S. (2004) 'Security in embedded systems: design challenges', *ACM Trans. Embed. Comput. Syst*., August, Vol. 3, No. 3, pp.461–491.

Rivest, R.L., Shamir, A. and Adleman, L. (1978) 'A method for obtaining digital signatures and public-key cryptosystems', *Communications of the ACM*, Vol. 21, No. 1, pp.120–126.

Römer, K. (2004) 'Programming paradigms and middleware for sensor networks', in *GI/ITG Workshop on Sensor Networks*, pp.49–54.

Skov, T. and Bro, R. (2005) 'A new approach for modelling sensor based data', *Sensors and Actuators B: Chemical*, Vol. 106, No. 2, pp.719–729.

TelosB Datasheet (2013) [online] http://www.datasheetarchive.com.

The pShield Project (2010) [online] http://pshield.unik.no/wiki/Main_Page.

Wiederhold, G. (1992) 'Mediators in the architecture of future information systems', *IEEE Computer*, Vol. 25, No. 3, pp.38–49.