Project no: 269317

nSHIELD

new embedded Systems arcHItecturE for multi-Layer Dependable solutions

Instrument type: Collaborative Project, JTI-CP-ARTEMIS

Priority name: Embedded Systems

D7.1: Railways security demonstrator - integration and validation plan

Due date of deliverable: M22 –2013.06.30

Actual submission date: M22 – 2013.06.30

Start date of project: 01/09/2011 Duration: 36 months

Organisation name of lead contractor for this deliverable:

Ansaldo STS, ASTS

Revision [Final]

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2012)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	**X**

Document Authors and Approvals			
Authors		**Date**	**Signature**
Name	**Company**		
Mariana Esposito	ASTS		
Francesco Flammini	ASTS		
Carlo Pompili	TELC		
Lorena de Celis	AT		
Nikos Pappas	HAI		
Kiriakos Georgouleas	HAI		
Balázs Berkes	S-LAB		
Kostas Fysarakis	TUC		
Kostas Rantos	TUC		
Dimitris Geneiatakis	TUC		
Harry Manifavas	TUC		
Alexandros Papanikolaou	TUC		
George Hatzivasilis	TUC		
Andrea Morgagni	SES		
Andrea Fiaschetti	UNIROMA		
Inaki Eguia	TECNALIA		
Reviewed by			
Name	**Company**		
Approved by			
Name	**Company**		

Applicable Documents		
ID	**Document**	**Description**
[01]	TA	nSHIELD Technical Annex

Modification History		
Issue	**Date**	**Description**
Draft 01	30.05.2013	First Draft
Draft 02	19.06.2013	Second Draft, architecture for demonstration and revised use case
Draft 03	26.06.2013	Contribution from TELCRED and AT
Draft 04	29.06.2013	Revised version of TOC
Draft 05	30.06.2013	Added list of nSHIELD prototypes outcome of the Stockholm meeting; added Annex A for ICDs; updated descriptions; editorial work Added proposed template for ICDs; added SPD features subsections; updated descriptions. Add SPD functionalities by AT and TELCRED

Contents

Executive Summary 10

1 Introduction 11

2 Terms and Definitions 12

3 nSHIELD Railway security scenario 13

3.1 Needs and problems 15

3.2 Risk Analysis 16

4 Railway security demonstrator reference architecture 25

5 Railway security System demonstrator technology overview 27

5.1 IDS prototype 27

5.1.1 IDS prototype interfaces 27

5.1.2 IDS prototype SPD features 27

5.2 Network layer security 28

5.3 Offline Physical Access Control 29

5.4 Protection Profile 30

5.5 Multi Metrics Approach 31

5.6 Attack Surface Metrics 32

5.7 Semantic model 33

5.8 Control algorithms 33

5.9 OSGI Middleware 34

5.10 Secure Agent 36

5.11 Secure Discovery 36

5.12 Automatic Access Control 37

5.13 Policy Based Access Control 38

5.14 Reputation Based Secure Routing 40

5.15 Smart card based security services 42

6 nSHIELD Railway security use cases 43

6.1 Scenario n. 1 43

6.2 Scenario n. 2 43

7 Railway security System demonstrator integration 45

8 Railway security System demonstrator validation and verification 48

8.1 Validation and Verification methods 48
8.2 Example of Validation process for demonstrator scenarios 49
8.3 Justification based on prototype and platform Validation and Verification 50
8.3.1 Validation and verification results for prototypes 50
8.4 Verification of Demonstrator scenario execution 52
8.4.1 Tools and platforms for execution of Demonstrator scenarios 52
8.4.2 Other HW and SW resources for execution of Demonstrator scenarios 53
9 Conclusions 54
10 References 55
Appendix A Interface Control Documents 56
A.1 Interface Control Document Automatic Access Control 56

Figures

Figure 3-1: The monitoring architecture 14
Figure 3-2: SMS-Security Management System GUI 14
Figure 3-3: System Security Architecture 16
Figure 4-1: Reference Architecture 25
Figure 4-2: Scheme of reference architecture 26
Figure 5-1: Multi metrics approach - Analysis 32
Figure 5-2: Knopflerfish start-up environment 35
Figure 5-3: Bundle architecture 35
Figure 5-4: Service discovery architecture 36
Figure 5-5: The SHIELD secure policy-based access control 39
Figure 5-6: WSNs and Cameras connected to nSHIELD Middleware 41
Figure 7-1: Interfaces of reference architecture 45
Figure 8-1: Validation and verification activities 49

Tables

Table 3-1: Qualitative frequency evaluation using associative matrix 17
Table 3-2: Qualitative Risk evaluation using associative matrix 17
Table 3-3: Risk Analysis 18
Table 5-1: IPSEC Security Levels 28
Table 5-2: IPSEC SPD levels 29
Table 5-3: Offline Physical Access Control SPD features 30
Table 5-4: Multi metrics approach 31
Table 5-5: Multi metrics approach - Final Dashboard 32
Table 5-6: Reputation scheme SPD levels 38
Table 5-7: PBAC SPD levels 40
Table 5-8: Reputation scheme SPD levels 40
Table 5-9: Reputation scheme SPD levels 42
Table 8-1: Relevant requirements of scenario n.2 50

Glossary

Please refer to the Glossary document, which is common for all the deliverables in nSHIELD.

This Page is intentionally left blank

Executive Summary

The purpose of this document is to present the plan and methodologies driving the integration, the validation and verification activities for the nSHIELD Railway Security Scenario. The document is structured as follows:

- Chapter 1 – provides a brief introduction on Railway security demonstrator scenario and its planning regarding methodologies and activities for integration and V&V plan.
- Chapter 2 – presents the SHIELD taxonomy, and a table with all nSHIELD prototypes
- Chapter 3 – presents detailed description of railway security application, its needs and problems and finally the nSHIELD solution proposed
- Chapter 4 – presents the reference architecture for the railway scenario demonstration
- Chapter 5 – presents the prototypes involved in the demonstrator scenario and their SPD characteristics
- Chapter 6 – presents the integration plan
- Chapter 7 – presents the V&V plan
- Chapter 8 – presents the scenarios for the demonstration
- Chapter 9 – draws the conclusions
- Appendix A – presents a detailed description of some prototypes interfaces

1 Introduction

The objective of the nSHIELD project is to conceive and design an innovative, modular, composable, expandable and high-dependable architectural framework which allows to achieve desired SPD levels in the context of integrated and interoperating heterogeneous services, applications, systems and devices. nSHIELD also aims at developing concrete solutions capable of achieving the aforementioned objectives in specific application scenarios with minimum engineering effort.

One of the four scenarios considered in the project is represented by the dependable surveillance systems used for railway security. The aim of this deliverable is to adapt the nSHIELD framework to suit the SPD requirements of railway protection systems. In fact, physical security systems for infrastructure protection have already been designed by Ansaldo STS (task leader) for rail-based transit systems, including urban/metro railways (e.g. subways, i.e. featuring underground stations and tunnels); in these systems, heterogeneous intrusion detection, access control, intelligent video-surveillance and intelligent sound detection devices are integrated in a cohesive Security Management System (SMS), that is the Ansaldo STS solution for Physical Security Information Management (PSIM). In this deliverable, we will plan a sample application of nSHIELD to a reference railway scenario architecture which includes some nSHIELD prototypes developed and presented in the other project deliverables.

More specifically, in this document we will explain the Integration and the V&V plan for the Railway Security scenario. In other words, this deliverable will show the integration of components and prototypes developed in WP3 (node layer), WP4 (network layer) and WP5 (middleware/overlay layer), the interoperability of the various SPD modules and the addressing of all SPD metrics and requirements that the integrated Railways Scenario needs to fulfil. The final aim is to have a plan to customize the nSHIELD platform to railway security scenarios and to validate the final behaviour of the integrated components.

This document is structured as follows. First, it will provide the description of the railway security application considered for the nSHIELD demonstration as well as its current limitations in SPD. Then the nSHIELD solution proposed to enhance system SPD will be shown. For the integration part, the nSHIELD prototypes involved in the railway scenario will be presented together with their interactions and interface connections. For the V&V part, some use-case examples and the process for the validation and verification of requirements will be presented.

2 Terms and Definitions

This section contains terms definitions, information that aids in understanding the document.

nSHIELD prototypes

A number of individual prototypes have been identified in the nSHIELD framework. They are enumerated in the following table and associated to an Id code.

Id	Name	Author
00	Elliptic Curve Cryptography	UNIGE
01	Lightweight Cyphering	TUC
02	Key Exchange Protocol	TUC
03	Hypervisor	SICS
04	Secure Boot	T2D
05	Secure Power (&) Communication Cape	AT/TELC/TUC
06	Smart Card	TUC
07	Facial Recognition	ETH
08	GPU Hase	TUC
09	Smart Transmission	SES/UNIGE
10	Anonymity	TUC
11	Automatic Access Control	TUC
12	DDoS Attack Mitigation	ATHENA
13	Recognizing DoS	ATHENA
14	Cellular Automata	UNIUD
15	Intrusion Detection System	MGEP
16	Reputation-Based Secure Routing	TUC/HAI
17	Access Control Smart Grid	TECNALIA
18	Policy Definition	ASTS/SES/SESM
19	Policy Based Management Framework	TUC/HAI
20	Control Algorithms	UNIROMA
21	Gateway	SESM
22	Middleware Intrusion Detection System	S-LAB
23	Link Layer Security	INDRA
24	Network Layer Security	TUC
25	OSGI Middleware	UNIROMA1
26	Semantic Model	UNIROMA1
27	Multi-metrics	TECNALIA
28	Attack Surface Metrics	SES
29	Adaptation of Legacy System	ATHENA
30	Reliable Avionic	ALFATROLL
31	Protection Profile	SES
32	Secure Discovery	UNIROMA1
33	Secure Agent	UNIROMA1
34	Audio Surveillance System	ISD
35	BeagleBoard-Xm	SICS
36	OMNIA-IMA	SES

3 nSHIELD Railway security scenario

Rail-based mass transit systems are vulnerable to many criminal acts, ranging from vandalism to terrorism. Therefore, physical security systems for infrastructure protection comprises all railway assets as for tunnel, train on board, platform and public areas, external Areas, technical control room, depots, electrical substations and etc… The objectives are to forecast critical threats as: aggressions and abnormal behaviours, sabotage and terrorism, vandalism and graffiti, thefts and pickpocketing. A modern smart-surveillance [14] system suitable for the protection of urban or regional railways is made up by the following subsystems:

1. Intrusion detection and access control:
    a. volumetric sensors for motion detection;
    b. magnetic contacts to detect illicit doors opening;
    c. glass break detectors;
    d. microphone cables for fence/grill vibration detection;
    e. active infrared barriers for detecting intrusions inside the tunnels;
2. Intelligent video-surveillance and Intelligent sound detection:
    a. advanced cameras with special features;
    b. digital video processing and recording, using efficient data compression protocols;
    c. video-analytics of the scenes, using computer vision algorithms;
    d. Microphones.
3. Dedicated communication network
4. Integrated management system

Distributed smart-sensors are installed along the railway line both in fixed (e.g. bridges, tunnels, stations, etc.) and mobile (passenger trains, freight cars, etc.) locations [15], [16] (Figure 3-1).

They are integrated locally using local wireless infrastructures (e.g. Wi-Fi, ZigBee, etc.) and then data is collected by WSN gateway nodes and transmitted remotely by means of WAN (Wide Area Network). Low/average bandwidth networks are strictly required to transmit alarms to the control centre, which are often already available (like GSM-R for railways) or easy to deploy (like satellite) and provide an extensive coverage of the infrastructure. However, if high-quality video streams from cameras need to be shown to the operators in order to verify the alarm and/or supervise the situation; higher bandwidth is required which can be possible achieved by multiple low bandwidth connections. This system are already been designed by Ansaldo STS for metro railways, where heterogeneous intrusion detection, access control, intelligent video-surveillance and abnormal sound detection devices are integrated in a cohesive Security Management System (SMS), Figure 3-2

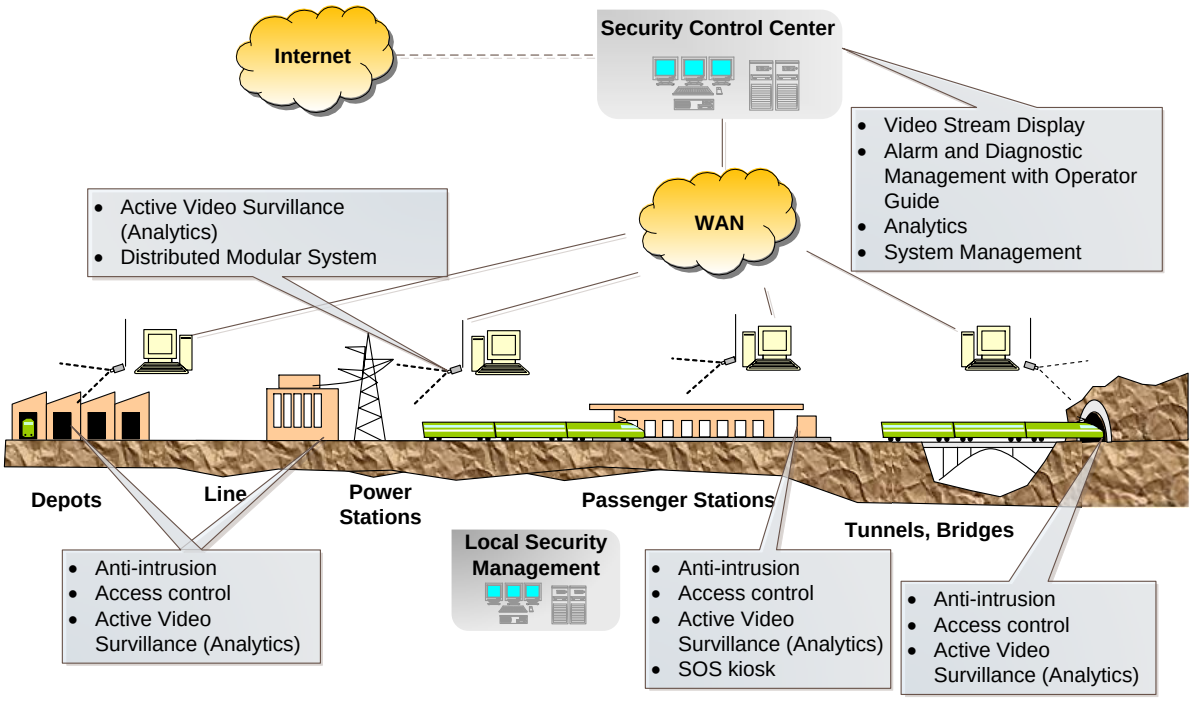

Figure 3-1: The monitoring architecture

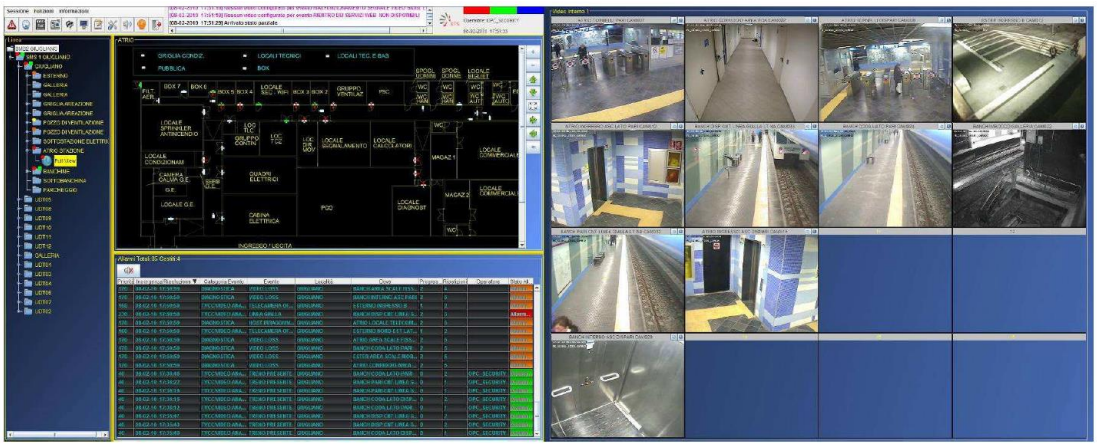

Figure 3-2: SMS-Security Management System GUI

The core of the SMS consists of a web-based software application featuring a graphical user interface. System architecture is distributed and hierarchical, with both local and central control rooms collecting alarms according to different scopes and responsibilities. In case of emergencies, the procedural actions required to the operators involved are orchestrated by the SMS. Redundancy both in sensor dislocation and hardware apparels (e.g. by local or geographical clustering) improve detection reliability, through alarm correlation, and overall system resiliency against both random and malicious threats. Video analytics is essential, since a small number of operators would be unable to visually control the large number of cameras which are needed to extensively cover all the areas needing to be protected. Therefore, the visualization of video streams is activated automatically when an alarm is generated by smart-cameras or other sensors, following an event-driven approach. Very high resolution cameras installed close to the turnstiles are used to automatically detect and store the faces of passengers, whose database can be accessed for post-event investigations. Real-time communication between the on-board and the ground is allowed by a wide-band wireless network.

3.1 Needs and problems

Currently, the security system described above is highly heterogeneous in terms not only of detection technologies (which will remain such) but also of embedded computing power and communication facilities. In other words, sensors differ in their inner hardware-software architecture and thus in the capacity of providing information security and dependability. This causes several problems:

- Information security must be provided according to different mechanisms and on some links - which are not “open” but still vulnerable to attacks - information is not protected by cryptographic nor vitality-checking protocols;
- Whenever any new sensor needs to be integrated into the system, a new protocol and/or driver must be developed and there is no possibility of directly evaluating the impact of such integration on the overall system dependability;
- New dedicated and completely segregated network links often need to be employed in order not to make the sensor network exposed to information related threats;
- The holistic assurance and evaluation of dependability parameters (e.g. for assessment/certification purposes) would be a very difficult task.

In particular both natural and malicious faults can impact on system availability and indirectly on safety, since the SMS is adopted in critical infrastructure surveillance applications. The problems mentioned above can be solved by adopting the nSHIELD architecture. Cohesion will be assured by wrapping sensors of any nature with homogeneous embedded hardware and software providing information security, by e.g.:

- Cryptographic protocols
- Vitality checking (heartbeat/watchdog timers based on sequence numbers and time-stamping).

The mechanisms provided by nSHIELD would mitigate the effects on the system of the following logical threats:

- Repetition (a message is received more than once)
- Deletion (a message is removed from a message stream)
- Insertion (a new message is implanted in the message stream)
- Re-sequencing (messages are received in an unexpected sequence)
- Corruption (the information contained in a message is changed, casually or not)
- Delay (messages are received at a time later than intended)
- Masquerade (a non-authentic message is designed thus to appear to be authentic)

Some sensing devices will be converted into smart-sensors by integrating the sensor unit with the nSHIELD processing units (both hardware and software) at the node level. The sensor networks will be integrated by the nSHIELD middleware before data is collected by the SMS and used at the presentation level (integration and reasoning). Typically, the monitoring system is composed by different sensors (IP-cameras, microphones, anti-intrusion device, etc…). They are connected through different communication networks and several topologies to a data centre. The data centre is connected to command and control centre. In Figure 3-3 is showed a typical architecture.

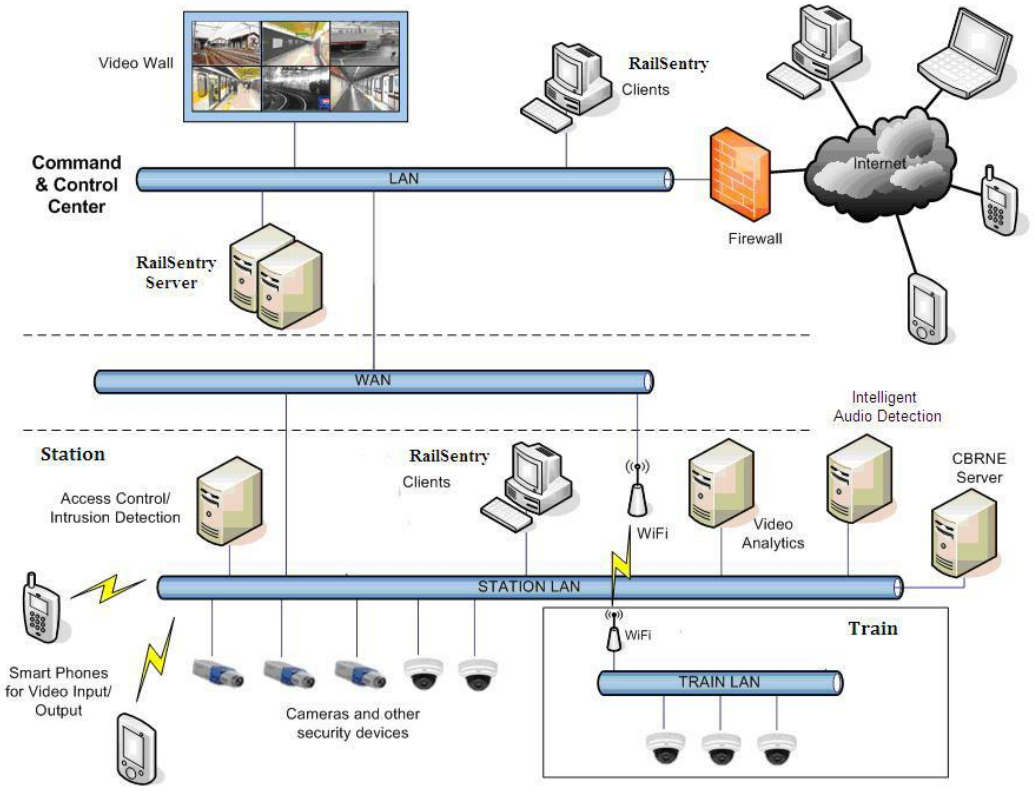

Figure 3-3: System Security Architecture

The sensors collect information about asset and send them to the Security Management Systems (SMS). The security system is composed by different sub-systems such as: video-surveillance, anti-intrusion detection, smart-audio surveillance. Signals coming from different sub-systems are elaborate in order to detect the corresponding events. The video-surveillance sub-system is able to guarantee both traditional functionalities (video stream management from different cameras, digital recording) and automatic video-analytics (motion detection, motion tracking) in order to manage critical events in the station. The anti-intrusion sub system and access control detect non-authorized access to protected sites (depots, Technical control rooms, etc...). The elaboration servers are able to recognize false alarms with the use of different type of technologies.

The smart-audio-surveillance Sub-systems are able to:

- Detect abnormal sound corresponding to vandalism, aggressions, etc…
- Identify the place in which happens this acts.

All sub-systems are connected through dedicated and completely segregated network links. Instead, real-time communication between the on-board and the ground is allowed by a wide-band wireless network.

3.2 Risk Analysis

Risk analysis will be used for evaluation of SPD risk in the nSHIELD railway security scenario [17].

Is possible to approach with the following steps:

- Define asset/component;
- For each asset/component will be identified the threat (T);

- For each T identified will be defined:
    - Likelihood (P): expected probability of occurrence of T (i.e. how probable is the threat);
    - Vulnerability (V): expected vulnerability with respect to T (i.e. how probable is it that T will cause the expected consequences);
    - Consequences (D): expected damage caused by T (i.e. an estimation of consequences caused by the threat);
    - Calculate risk (R): **R= P · V · D**

For likelihood, vulnerability and consequences evaluation is adopted a qualitative technique.

Qualitative evaluation use reduced scales of values of intuitive meaning, for instance: Low, Medium, and High. The advantage is that estimations can be more straightforward and computations easier. The disadvantage is that results are usually less rigorous and the combination of qualitative indices questionable.

The P · V factor is compacted into a single factor, which – to avoid confusion – we will define here as the frequency F of "successful" threats. Hence:

$$F = P \cdot V$$

The F evaluation is conducted through an associative matrix:

Table 3-1: Qualitative frequency evaluation using associative matrix

P \ V	Low	Medium	High
Low	Low	Low	Medium
Medium	Low	Medium	High
High	Medium	High	High

Qualitative risk evaluation uses associative matrix reported below using the estimated values of F and D:

Table 3-2: Qualitative Risk evaluation using associative matrix

PV \ D	Low	Medium	High
Low	Low	Low	Medium
Medium	Low	Medium	High
High	Medium	High	High

Based on this information is possible to identify the mean element of architecture to protect and possible threats for Railroad Security scenario.

Table 3-3 shows some components and relative risk analysis:

Table 3-3: Risk Analysis

Assets to protect	Threats	Vulnerability (V)	Likelihood (P)	Consequences (D)	Risk R=P xV x D
Ethernet Camera **Analog Microphone**	Physical tamper/manumission such as: • Cable disconnection; • Theft • Significant movement or replacement • Other relevant damage meant to put the unit out of order	**HIGH** If they are located in a public c area.	**HIGH**	**LOW** Operation of the single sensor is compromised, as the related monitoring functionality. The easy diagnosability of the attack reduces its impact	**MEDIUM**
Ethernet Camera **Wi-Fi Camera** **Mote WSN**	HW fault: • Loss of component functionality • Loss of sensor functionality SW fault: • Bug • Aging • Transient fault	**MEDIUM** In general HW and SW are vulnerable, especially after some operation time, to this fault.	**HIGH** It depends on HW and SW robustness and environmental condition.	**MEDIUM** Effects range from loss of specific functions to loss of related monitoring functionality. It is difficult to diagnose	**HIGH**
Ethernet Camera **Wi-Fi Camera** **Mote WSN**	Alteration of connection due to: • Network overload • Involuntary disconnection	**MEDIUM** It depends on environmental condition, bandwidth, capacity of connection, number of sensors	**MEDIUM** It depends on network architecture.	**LOW** Communication of the single sensor is compromised, as the related monitoring functionality. The easy diagnosability of the fault reduces its impact	**LOW**
Ethernet Camera	Unauthorized network access Data destruction	**LOW** Connection are wired and encrypted It depends on attacker ability.	**LOW**	**HIGH** The attacker takes the control of communication and he can alterate the data of sensors and relative alarms. It is difficult to diagnose.	**MEDIUM**

Wi-Fi Camera **Mote WSN**	Physical tamper/manumission, such as: • Theft • Significant movement or replacement • Any other relevant damage meant to put the unit completely out of order	**HIGH** If they are located in a public c area.	**MEDIUM**	**MEDIUM** Operation of the single sensor is compromised, as the related monitoring functionality. The easy diagnosability of the attack reduces its impact	**HIGH**
Wi-Fi Camera **Mote WSN**	Alteration of connection due to: • Interferences with electromagnetic device	**HIGH** The connection are wireless	**LOW** It depends of network architecture.	**LOW** Operation of the single sensor is compromised, as the related monitoring functionality. The easy diagnosability of the fault reduces its impact	**LOW**
Wi-Fi Camera **Mote WSN**	Unauthorized network access Data destruction	**MEDIUM** Connection are wireless but can be encrypted	**MEDIUM**	**HIGH** The attacker takes the control of communication and he can alterate the data of sensors and related alarms. It is difficult to diagnose.	**HIGH**
Ethernet Camera **Wi-Fi Camera** **Mote WSN**	Data alteration	**MEDIUM** Connections can be encrypted	**MEDIUM**	**HIGH** The attacker takes control of communication and he/she can modify the data of sensors and related alarms. It is difficult to diagnose.	**HIGH**
Ethernet Camera	Data Sniffing	**LOW** Connection is wired	**MEDIUM** It requires physical access to cable connection.	**HIGH**	**MEDIUM**
Wi-Fi Camera **Mote WSN**	Data Sniffing	**HIGH** Connection is wireless	**HIGH** It requires equipment available commercially	**HIGH**	**HIGH**

Analog Microphone	HW fault: • Loss of component functionality • Involuntary disconnection	**MEDIUM** In general HW and SW are vulnerable, especially after some operation time, to this fault.	**MEDIUM** It depends on HW and SW robustness and environmental condition.	**MEDIUM** Effects range from loss of specific functions to loss of related monitoring functionality. It is difficult to diagnose	**MEDIUM**
Wi-Fi Camera **Mote WSN**	Transmitted data scrambling (e.g. high-power microwave generators)	**HIGH**	**LOW** It requires equipment not available commercially	**MEDIUM** Since it can affect a large number of sensors located in the same area.	**MEDIUM**
Anti-intrusion sensor (via serial loop through proprietary protocol)	Physical tamper/manumission such as: • Cable disconnection; • Theft • Significant movement or replacement • Other relevant damage meant to put the unit out of order	**HIGH** If they are located in a public c area.	**MEDIUM**	**HIGH** Operation of the single sensor is compromised, as the related monitoring functionality. The easy diagnosability of the attack reduces its impact	**HIGH**
Anti-intrusion sensor (via serial loop through proprietary protocol)	HW fault: • Loss of component functionality • Involuntary disconnection SW fault: • Bug • Aging • Transient fault	**MEDIUM** In general HW and SW are vulnerable, especially after some operation time, to this fault.	**MEDIUM** It depends on HW and SW robustness and environmental condition.	**MEDIUM** Effects range from loss of specific functions to loss of related monitoring functionality. It is difficult to diagnose	**MEDIUM**
Anti-intrusion sensor (via serial loop through proprietary protocol)	HW fault: • Loss of component functionality • Involuntary disconnection	**MEDIUM** In general HW and SW are vulnerable, especially after some times, to this fault.	**MEDIUM** It depends on HW and SW robustness and environmental condition.	**MEDIUM** Effects range from loss of specific functions to loss of related monitoring functionality. It is difficult to diagnose	**MEDIUM**
Application server	Random corruption of data Loss of data integrity	**MEDIUM** It depends on redundant/fault-tolerant components.	**LOW** It depends on how much the HW is reliable / ruggedized and on environmental conditions (e.g. air conditioning).	**HIGH** Effects range from loss of specific functions to loss of a whole (sub) system.	**MEDIUM**

Application server	Physical tamper/manumission such as: • Cable disconnection; • Theft • Significant movement or replacement • Other relevant damage meant to put the unit out of order	**LOW** The servers are in technical control room	**LOW** The servers are in technical control room	**HIGH** The monitoring application is compromised	**MEDIUM**
Application server	Unauthorized network access Sniffing	**MEDIUM** The network is connecting to the Internet. Using firewalls reduces vulnerability	**MEDIUM** Nowadays attempts to attack public utility servers are not rare	**HIGH** Once accessed by the attackers, the servers are completely under their control, and furthermore the attack can be difficult to detect.	**HIGH**
Application server	Transmitted data scrambling (e.g. high-temperature generators, fault of air conditioning)	**LOW** The servers are in technical control room	**MEDIUM** It requires equipment available commercially.	**HIGH** Since it can affect a large number of servers located in the same area. The monitoring application is compromised	**MEDIUM**
Application server	HW fault: • Loss of component functionality • Loss of server functionality SW fault: • Bug • Aging • Transient fault	**MEDIUM** In general HW and SW are vulnerable, especially after some operation time, to this fault.	**MEDIUM** It depends on HW and SW robustness and environmental condition.	**MEDIUM** Effects range from loss of specific functions to loss of related monitoring functionality. It is difficult to diagnose	**MEDIUM**
Application server	Alteration of connection due to: • Network overload • Involuntary disconnection	**MEDIUM** It depends on environmental condition, bandwidth, capacity of connection, number of servers	**LOW** It depends of network architecture.	**HIGH** Operation of the server is compromised, as the whole monitoring system.	**LOW**
Emergency button	HW fault	**MEDIUM** In general HW and SW are vulnerable, especially after some times, to this fault.	**MEDIUM** It depends on HW and SW robustness and environmental condition.	**MEDIUM** Loss of alert functionality	**MEDIUM**

Emergency button	Physical tamper/manumission such as: • Cable disconnection; • Destruction	**LOW**	**MEDIUM**	**MEDIUM** Loss of alert functionality	**MEDIUM**
Client operator/video wall	HW fault: • Loss of component functionality • Loss of video functionality SW fault: • Bug • Aging • Transient fault	**MEDIUM** In general HW and SW are vulnerable, especially after some times, to this fault.	**MEDIUM** It depends on HW and SW robustness and environmental condition.	**MEDIUM** Loss of specific functions functionality. It is easy to diagnose	**MEDIUM**
Client operator/video wall	Alteration of connection due to: • Network overload • Involuntary disconnection	**MEDIUM** It depends on environmental condition, bandwidth, and capacity of connection.	**LOW** It depends on network architecture.	**MEDIUM** Loss of specific functions functionality. It is easy to diagnose	**MEDIUM**
Client operator/video wall	Unauthorized network access • Data disruption/alteration • Loop video	**LOW** Connection are wired	**LOW** It depends on attacker ability.	**HIGH** Difficult to diagnose	**MEDIUM**
Mobile client (PDA, Smartphone, etc.) or remotely connected client (using Internet)	HW fault: • Loss of component functionality • Loss of client functionality SW fault: • Bug • Aging • Transient fault	**MEDIUM** In general HW and SW are vulnerable, especially after some times, to this fault.	**MEDIUM** It depends on HW and SW robustness and environmental condition.	**MEDIUM** Effects range from loss of specific functions to loss of alert function.	**MEDIUM**
Mobile client	Alteration of connection due to: • Network overload • Involuntary disconnection	**MEDIUM** It depends on environmental condition, bandwidth, and capacity of connection.	**LOW** It depends on network architecture.	**LOW** Loss of alert functionality	**LOW**

Mobile client	Alteration of connection due to: • Interferences with electromagnetic	**MEDIUM** The network is connecting to the Internet. Using firewalls reduces vulnerability	**LOW** It depends on network architecture.	**LOW** Loss of alert functionality	**LOW**
Mobile client	Unauthorized network access due to: • Data destruction/alteration	**MEDIUM** The network is connecting to the Internet. Using firewalls reduces vulnerability	**MEDIUM** It depends on hacker ability.	**HIGH** Difficult to diagnose	**HIGH**
Network Switch	Physical tamper/manumission such as: • Cable disconnection; • Theft • Other relevant damage meant to put the unit out of order	**LOW** The switch are in technical control room	**LOW**	**HIGH** Loss of communication	**MEDIUM**
Network Switch	HW fault: • Loss of component functionality • Loss of switch functionality SW fault: • Bug • Aging • Transient fault	**MEDIUM** In general HW and SW are vulnerable, especially after some times, to this fault.	**MEDIUM** It depends on HW and SW robustness and environmental condition.	**HIGH** Loss of communication	**MEDIUM**
Network Switch	MAC flooding Control of switch	**HIGH**	**MEDIUM** Some security means can limit this threat (e.g. Port security.)	**HIGH** Loss of communication	**HIGH**
Logical control unit for Anti-intrusion/Access Control via Ethernet	Physical tamper/manumission such as: • Cable disconnection; • Theft • Significant movement or replacement • Other relevant damage meant to put the unit out of order	**LOW** The Control Units are in technical control room	**MEDIUM**	**HIGH** Loss of functionality	**MEDIUM**

Logical control unit for Anti-intrusion/Access Control via Ethernet	HW fault: • Loss of component functionality • Loss of camera functionality SW fault: • Bug • Aging • Transient fault	**MEDIUM** In general HW and SW are vulnerable, especially after some times, to this fault.	**MEDIUM** It depends on HW and SW robustness and environmental condition.	**HIGH** Effects range from loss of specific functions to loss of related monitoring functionality.	**HIGH**
Logical control unit for Anti-intrusion/Access Control via Ethernet	Alteration of connection due to: • Network overload • Involuntary disconnection	**MEDIUM** It depends on environmental condition, bandwidth, and capacity of connection.	**LOW** It depends on network architecture.	**HIGH** Loss of functionality	**MEDIUM**

4 Railway security demonstrator reference architecture

Figure 4-1 shows the reference architecture for the nSHIELD demonstrator. While the Figure 4-2 shows the scheme of architecture and prototype involved.

Let us suppose a video surveillance from a control room of a vehicle (e.g. light metro, tram) moving in an urban area, or of shelter (building for housing electrical and electronic high-tech telecommunications, and control systems for railways).

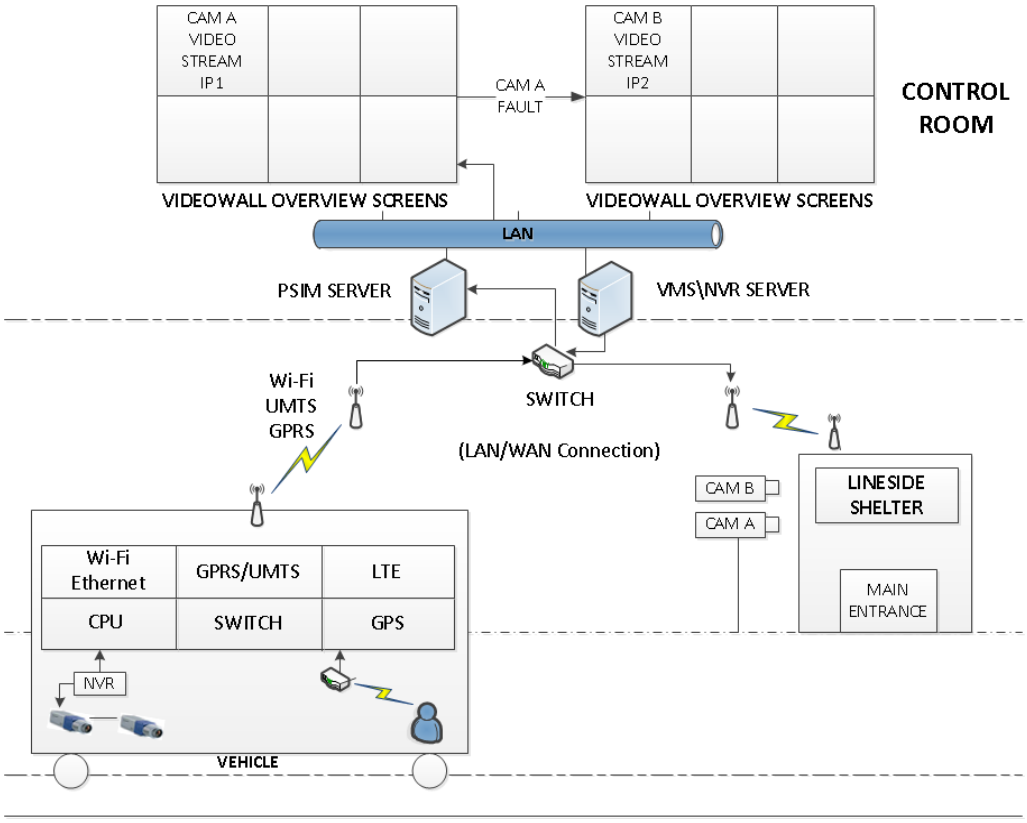

Figure 4-1: Reference Architecture

The link between vehicle/shelter and control room is wireless and the exchanged information are:

- Vehicle to control room:
  - On-board diagnostics and alarms generated by emergency buttons, environmental sensors and video analytics running on the on-board NVR (Network Video Recorder)
  - GPS information for localization (periodic, on request or on event)
  - Event activated real-time video streaming (at adaptable quality, resolution and frame-rate depending on available bandwidth) from on-board cameras
  - User-requested video recordings from on-board NVR
- Shelter to control room:
  - Video stream of the shelter main entrance;

- Event activated real-time video streaming (at adaptable quality, resolution and frame-rate depending on available bandwidth) from cameras

- Control room to vehicle/shelter
  - Configuration change commands
  - Requests for information (position, video streams, downloads of recording, device status report, etc.)

In next sub-section we will describe the nSHIELD technologies and prototypes can be integrated.

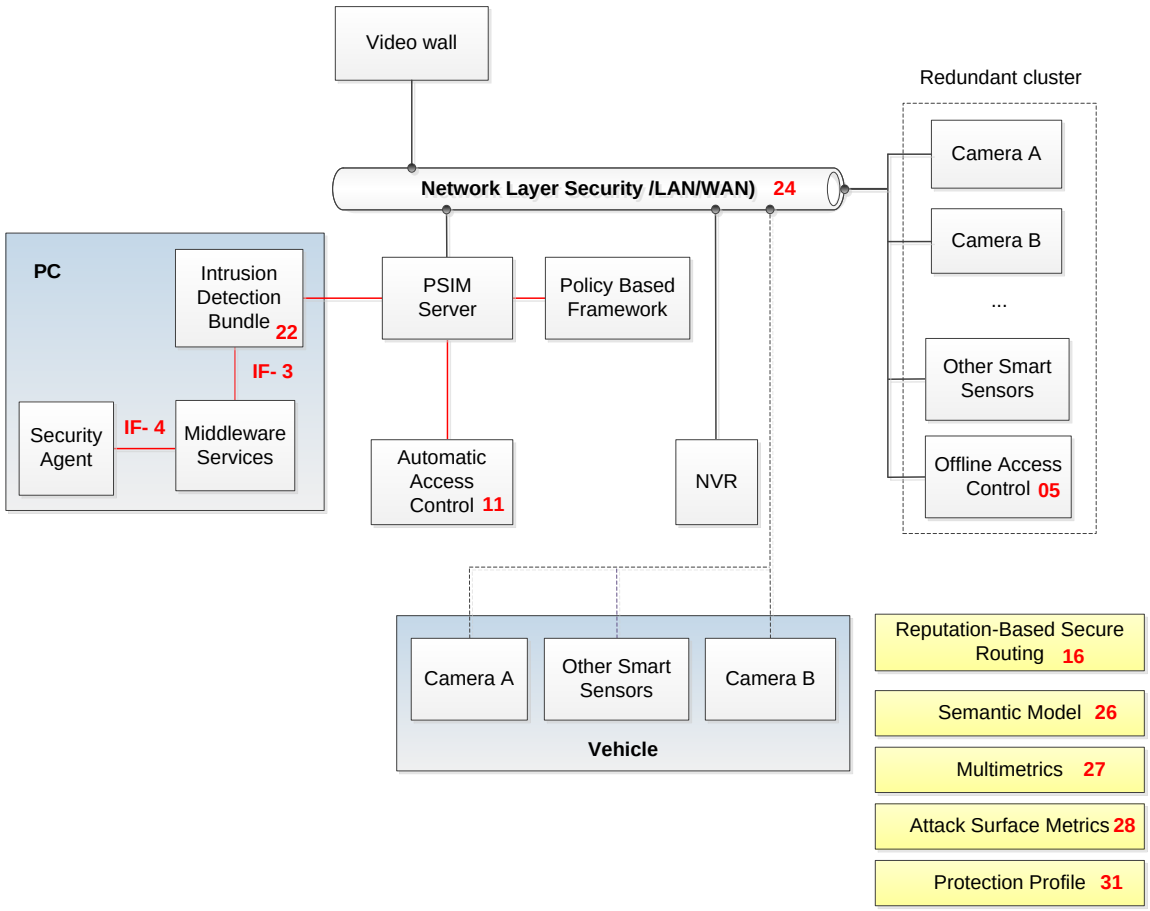

Figure 4-2: Scheme of reference architecture

5 Railway security System demonstrator technology overview

This section explains benefits of nSHIELD platform in the railway case study. Furthermore, it will specify nSHIELD prototypes and technologies can be applied. In each section will report the description of prototype and its integration

5.1 IDS prototype

The Intrusion Detection and Filtering Module was delivered as a preliminary prototype in D5.2 [5], and is described in detail in D5.3 [6].

The IDS prototype operates as a TCP/UDP gateway, having network interfaces connected towards the Middleware services present, and similar network interfaces towards other parts of the system accessing the Middleware services. The main role of the preliminary IDS prototype is to filter requests towards middleware functionality, received from network interfaces that utilize common and/or public network infrastructure, also protecting against requests from compromised nSHIELD nodes. The IDS prototype is created to operate autonomously after initial setup, but it also provides function interfaces for real-time monitoring and control according to SPD requirements, implemented natively in the OSGI Framework environment.

5.1.1 IDS prototype interfaces

In its current status, the preliminary IDS prototype has generic network interfaces for receiving and forwarding requests that are to be filtered. It is however anticipated that TAP / TUN virtual network interfaces could be used to physically separate and protect internal (Middleware services) and external (other components and networks besides Middleware) network domains. These changes could mostly be implemented in a transparent manner for the system components using middleware services, but may impact how connection methods towards middleware services should be implemented.

The network interfaces operate in a transparent manner, but require setting up network infrastructure so that requests are received by the gateway instead of the middleware services natively. For this purpose, the Intrusion Detection and Filtering Module provides additional function call interfaces towards the middleware services that implement the use of the IDS – see next chapter about SPD features.

5.1.2 IDS prototype SPD features

The preliminary version of the Intrusion Detection and Filtering Module provides the following features for the Middleware services utilizing the IDS. These features are controllable via function interfaces in the Middleware environment in Java:

- Intrusion detection configurable per service
- Provides blacklisting and white-listing for clients – operation mode and lists can be controlled from the Overlay based on higher level semantic SPD information (e.g. based on trust level associated with clients obtained from Secure Discovery)
- Critical Load Detection of Server
- Can be switched to white-listed or blacklisted mode, or can switch automatically under critical load (can be controlled according to required SPD level changes as well)
- Provides function interface to query Service Metrics that can be used to assess SPD level of the prototype:
    - *totalIncomingRequestCount*

- *totalOutgoingResponseCount*
- *totalDroppedFromQueueCount*
- *currentQueueSize*
- *totalBlacklistRejection*
- *totalWhitelistRejection*

5.2 Network layer security

The railway monitoring system comprises of a wide number of distributed nodes that remotely report sensitive information to the control room. This communication is subject to unauthorized disclosure and/or alteration and therefore proper protection of these messages is a strong requirement and can be satisfied by securing them at various layers of the TCP/IP stack.

Security at the network layer is provided by the standardised IPSEC protocol adapted to the very restricted environment that some nSHIELD nodes provide. One of the main features that needed to be considered for this adaptation is the limited messages length that the underlying protocols such as the IEEE802.15.4 provide. IPv6 enabled nodes can utilise header compression, a technique that significantly reduces the long IPv6 header to an acceptable for the underlying IEEE802.15.4 frame maximum length, so that the number of exchanged frames is reduced, and in most cases can it into a single message. The resulting protocol provides message confidentiality, integrity and authentication to IP and upper layers' messages while the participating nodes have to ability to choose among a variety of options regarding the level of protection provided to messages, including, the encryption or not of the messages and the size of the authentication block.

The proposed IPSEC protocol can provide end-to-end security and save significant resources of the intermediate routing nodes which are relieved from the need to decrypt and re-encrypt communication messages. As a result the command and control centre can utilise IPSEC to communicate securely with remote nodes and transfer sensitive information without endangering unauthorized disclosure or modification of this sensitive information. Within this context, the IPSEC protocol has been implemented on Contiki platform. The corresponding functionality comprises part of the TCP/IP stack where the IPSEC functionality is provided as an additional feature to the existing IP protocols.

Network layer security SPD features

IPSEC protocol can provide confidentiality, integrity and authentication on exchanged messages using pre-established keys shared among the participating entities. The system owner has the capability to define the security features provided by the IPSEC, such

The security levels available by the aforementioned functionality are shown in Table 5-1.

Table 5-1: IPSEC Security Levels

Security Level	Security Attributes	Data Confidentiality	Data Authenticity	Encrypted authentication tag length, M octets
0	None	OFF	NO	0
1	MIC-32	OFF	YES	4
2	MIC-64	OFF	YES	8
3	MIC-128	OFF	YES	16
4	ENC	ON	NO	0
5	ENC-MIC-32	ON	YES	4
6	ENC-MIC-64	ON	YES	8
7	ENC-MIC-128	ON	YES	16

These levels can be mapped to the following SPD levels shown in Table 5-2

Table 5-2: IPSEC SPD levels

SPD Level	Functionality
1 (lowest)	No encryption – No Authentication-
2 (low)	Encryption only, Authentication only
3 (medium)	Encryption + MIC-32, MIC-64
4 (high)	Encryption + MIC-128

5.3 Offline Physical Access Control

Railway operators are responsible for a large number of locked assets spread over a large area. Due to the high cost of traditional access control systems, many railway operators still rely on traditional mechanical keys, which are both inefficient and insecure. For example, the operator of the subway system in Stockholm, Sweden, is responsible for more than 20 000 doors, of which only 10% are currently equipped with an access control system.

The nSHIELD partner Telcred (TELC) develops an offline access control system. An offline system is much less costly to install compared to an online system, and therefore an interesting alternative to online systems and traditional keys. At the door, the solution is comprised of a reader, a lock controller and an electric lock.

The lock controller is placed on the inside of the door and is a critical component since it is responsible for making the actual access control decisions based on the credentials presented by the user to the reader. In other words, it is highly important that this component is both reliable and resistant to attacks.

Within the scope of the project, Telcred (TELC), Acorde (AT), and SICS are collaborating on developing a secure micro node, which can be used as a lock controller. A custom “cape” for a standard BeagleBone low end Linux computer will be developed. This cape will provide features such as tamper detection, backup power, secure storage of cryptographic keys, and a real time clock.

For the use case demonstration, the physical access control system can be integrated with other nSHIELD components through the back end. In other words, the overarching SMS (security Management System) can be integrated with the oPACS (Physical Access Control System) on the back end, while the secure lock controller will operate offline/standalone.

Offline Access Control SPD features

This nSHIELD node prototype is composed of different subsystems that are directly related to different partners’ expertise. This prototype has been designed as a BeagleBone cape. The main functionalities of this prototype

a. **Custom encapsulation + Supervisor and anti-tampering**
b. **Power unit** for the BeagleBone board and third-party boards
c. **TPM module** to support the storage of the security keys that are involved in the partners cryptographic developments. This feature is provided through a holder/slot for a smart card with form factor ID-000 (same as a typical SIM-card). This way, different hardware can be used depending on the application (using a smart card with Java Card for secure storage and to serve as a crypto co-processor).
d. **RF Module** that supports the 802.15.4, based on the MRF24J40 that provides a wireless communication link.

e. **Other features**:
   i. Additional RS-485/RS-232 external interfaces (driver + connector) will be available in the cape.
   ii. RTC signal will be provided.
   iii. Two relays
   iv. Several digital inputs

With this prototype some SPD functionalities that could be covered are listed in Table 5-3:

Table 5-3: Offline Physical Access Control SPD features

Digital Signatures	Different signature verifications can be done using Java Card applets (like implementation of ECDSA java card applet on the smart card)
Physical/tamper resilience	This feature is a requirement that has been considered during design stage. It has been included a supervisor chip to cover this feature connected to a switch.
ECC Authentication	This feature can be covered by means a software solution like using Java Card/JCOP smart cards (built-in functionality in Java Card)
Accommodations for future energy sources	The design of AT custom power module will include a power input interface for alternative power sources and on-board battery to allow the future implementation of power harvesting technologies
Power management	This requirement manages any system power supply risk, which might affect to the node behaviour. In case of failure of any of the countermeasures, being able to protect all the electronics and devices, in order to avoid further damages into the system and increase the node availability

5.4 Protection Profile

The nSHIELD project has the ambitious to be a commercial standard for Security, Privacy and Dependability regarding embedded systems. At this purpose the idea of a Protection Profile (at the moment only for middleware layer) is a first step to define a security problem definition and security objectives for embedded systems.

As defined in D5.3, a protection profile (PP) is a Common Criteria[1] (CC) term for defining an implementation-independent set of security requirements and objectives for a category of products, which meet similar consumer needs for IT security. Examples are PP for application-level firewall and intrusion detection system. PP answers the question of "what I want or need" from the point of view of various parties. It could be written by a user group to specify their IT security needs. It could also be used as a guideline to assist them in procuring the right product or systems that suits best in their environment. Vendors who wish to address their customers' requirements formally could also write PP. In this case, the vendors would work closely with their key customers to understand their IT security requirements to be translated into a PP. A government can translate specific security requirements through a PP. This usually is to address the requirements for a class of security products like firewalls and to set a standard for the particular product type.

Protection Profile defines the rules or rather the SPD requirements that must be met by prototypes Integration that make up an embedded system aiming to be SHIELD compliant (as indicated above, at this time are shown only the SPD requirements that the middleware of the system must meet).

[1] Common Criteria is a standard for security specifications and evaluation, ISO15408.

Protection Profile SPD features

As indicated in the nSHIELD middleware PP [10] Protection Profile (PP) applies to middleware layer of a generic embedded system which aims to be compliant to nSHIELD project, which we'll consider as Protection Profile Target of Evaluation (TOE).

The TOE is part of a system. It is software and its purpose is to act as glue for the different SPD services offered by nSHIELD compliant embedded system.

The TOE features security functions for:

- Identification & Authentication;
- Auditing;
- Data Integrity;
- Availability.

This generic identification of security functions can be mapped on TOE through the following statement:

- Orchestrator that improves services discovery/composition is able to identify and authenticate services/devices (discovered/composed);
- TOE is able to record security relevant events;
- TOE is able to verify the integrity of composition command definition;

TOE is able to grant services availability

5.5 Multi Metrics Approach

Multi metric approach will address the procedure maintained by document 2.8 called "An Evolutionary-Fuzzy Approach towards Multi-Metric Security Risk Assessment in Heterogeneous System of Systems". For this specific scenario the scenario owner should reflect the following in Table 5-4 .

1. Select correct metrics for Railway scenario from document 2.5. This selection takes into account the risks identified in the present document. The following table could be an example of selected metrics:

Table 5-4: Multi metrics approach

Threat	Metric
Repetition	nº messages repeated/total messages
Deletion	nº messages deleted/total messages
Insertion	nº messages inserted/total messages
Re-sequencing	nº messages re-sequenced/total messages
Corruption	nº messages altered/total messages
	nº message digest corrupted (MD5/SHA1)
Delay	Time of message from A to B
Masquerade	Cryptographic ley length for AUTH
	nº message header altered/total headers

2. An analysis of selected metric should be featured as in Figure 5-1:

Threat	Full domain	Comments	Funtion types in correct region	Function type in incorrect regior	Layers	SPD
Repetition	[0, max$_{time}$]*	unit is relation of nº of messages	linear	logarithmic	Network	S
Deletion	[0, max$_{time}$]	unit is relation of nº of messages	linear	logarithmic	Network	D
Insertion	[0, max$_{time}$]	unit is relation of nº of messages	linear	logarithmic	Network	S,D
Re-sequencing	[0, max$_{time}$]	unit is relation of nº of messages	linear	logarithmic	Network	S
Corruption	[0, max$_{time}$]	unit is relation of nº of messages	linear	logarithmic	Network	S
	[0, max$_{time}$]	unit is relation of nº of messages	logarithmic	logarithmic	Net, midd	S
Delay	[0, max$_{time}$]	Depending of assests. Unit is in nanoseconds	exponential	esponential-->logarithmic	Net, midd	D
Masquerade	[0,2048]	Key length (129, 192, 256, 512)	linear	logarithmic	Net, midd	S
	[0, max$_{time}$]	unit is relation of nº of messages	linear	logarithmic	Net, midd	S

Figure 5-1: Multi metrics approach - Analysis

3. Expert system development according to scenario owner expertise. In this case, a specific FUZZY-LOGIC IF- THEN algorithm will be issued according to Deliverable 2.8 document.

4. Final Dashboard should be as in Table 5-5 :

Table 5-5: Multi metrics approach - Final Dashboard

Layer	S	P	D
Node	(G,Y,R)	(G,Y,R)	(G,Y,R)
Network	(G,Y,R)	(G,Y,R)	(G,Y,R)
Middleware	(G,Y,R)	(G,Y,R)	(G,Y,R)
Overlay	(G,Y,R)	(G,Y,R)	(G,Y,R)

Summarising, Scenario owners identify key metrics, normalise and analyse them, and finally with the help of a fuzzy IF-THEN algorithm can obtain an aggregated heterogeneous multi metric measurement via this dashboard.

5.6 Attack Surface Metrics

Attach Surface Metric approach starts from the following considerations:

1. A threat is the origin of the fault chain (fault -> errors -> failures) for the dependability concerns and as the potential for abuse of protected assets by the system for security concerns.
2. The attacker is the threat agent; it is a malicious human activity or non-malicious event.
3. An attacker uses nSHIELD's entry and exit points to attack the system.

So it was introduced an entry and exit point framework to identify three relevant factors: Porosity, Controls, and Limitations.

An entry and exit point contribution to the attack surface reflects factors' likelihood of being used in attacks. For example an entry point running a method with root privilege is more likely to be used in attacks than a method running with non-root privilege. We introduce the notion of a damage potential-effort ratio (DER) to estimate porosity contribution.

A system's attack surface measurement (Actual SPD Level) is the total contribution of the system's factors along the porosity, controls, and limitation.

Each supplier of a product or system that will be part of this demonstrator must provide the data needed for the calculation of SPD level defined by the adopted metric approach.

These data will be provided by filling in an excel sheet which is being finalized and will contain all the information necessary to Actual SPD level calculation.

The Attack surface metric approach definition and the details of data to be provided are contained in deliverable D2.5 [11].

5.7 Semantic model

The Semantic Model was delivered as a preliminary prototype in D5.2 [5], and is described in detail in D5.3 [6].

The composability of SPD functionalities works as a closed loop control scheme, driven by the comparison between the desired SPD level and the current one and the implementation of the proper control actions to reduce their gap. The key of this mechanism is the possibility of abstracting the system into a technology independent layer on which the intelligent control algorithms and metrics can be applied: the abstraction is necessary due to the fact that the underlying system is heterogeneous both in technologies and in purposes.

The methodology derived to produce the SHIELD semantic models is based on the definition of two information repository: one for the subtract information and one for the domain dependent information. The first one contains the ontology provided by the components and the second one contains the parameters, relations or rules tailored by the domain experts once the system is deployed.

For the purposes of the demonstrator, two small but significant example of these repositories will be provided.

- For the ontology, an XML template will be distributed to the prototypes providers to be filled with relevant information: this files will be collected by the OSGI and stored in a specific repository (that is not necessarily a Data Base but could be also a portion of RAM).
- For the domain dependent library, a set of entries will be prepared and manually inserted in the OSGI, to be used for control purposes

Semantic model SPD features

The ontology file includes simply a list of SPD functionalities offered by a specific system component. For each SPD functionality, some attributes are reported, in line with the procedures for metrics computation, and these attributes are filled with the metric values elaborated following metrics guidelines (attack surface and/or multi metrics).

The domain dependent library contains mainly: i) a set of inclusion/non-inclusion relations for the different SPD components (if any), ii) the overridden values for the ontology attributes (if any) and iii) policies to force the behaviour of the SHIELD system (if any).

5.8 Control algorithms

The Control Algorithms was delivered as a preliminary prototype in D5.2 [5], and is described in detail in D5.3 [6]. However the version of the control algorithms that will be included in the demonstrator will most likely be the one delivered with the final prototypes, since it will be fully compliant with the mechanism for metrics computations. The control algorithms work as follows: at first a set of candidate technologies is identified as well as the SPD value desired by the user; then an algorithm is applied (a simple optimization, an extensive graph search, a model driven control, ECC) whose result is a list of components that should be activated to reach the desired objectives. Finally these solutions are filtered by including the domain constraints/tailoring and, if there are no changes in the metric value, the solution is confirmed; however it is reiterated taking into account the new inputs.

Control algorithms SPD features

The control algorithms are implemented by the Security Agent in the Overlay layer. On a practical point of view, two solutions are envisaged to include them in the demonstrator, depending on the maturity level reached by the implementation task: direct inclusion in the OSGI source code, or interfacing with an external computation platform (i.e. Matlab) that provides the problem solutions. In both these cases the interfaces between the control algorithms and the system are internal and can be easily managed

5.9 OSGI Middleware

The OSGI framework was delivered as a preliminary prototype in D5.2 [5], and is described in detail in D5.3 [6]. This framework is the platform adopted to emulate the functionalities of a generic Middleware. The considerations that lead to this choice are reported in the following and taken directly from the pSHIELD documents, in which this choice was preliminarily investigated. In fact, considering the possible available SOA open solutions, the decision was to select OSGi as the reference service platform to implement the Middleware services. The main reasons leading to this decision are:

- OSGi is an open standard;
- OSGi has a number of open source implementation (Equinox, Oscar, Knopflerfish);
- OSGi can be executed even over lightweight nodes (Embedded Systems Devices);
- OSGi has been implemented using different programming languages (e.g. Java, C, C#);
- The Java implementations of OSGi is fast to deploy and it is much easier to learn, facilitating even an active and collaborative prototype deployment among partners;
- OSGi plugins are available for a number of IDE tools (i.e. Eclipse, Visual Studio, etc.);
- OSGi can be easily deployed in Windows (XP, 7, Mobile), Linux, MAC and Google (Android) OSes.

More in particular it has been decides to use the open source Knopflerfish OSGi service platform. Knopflerfish (hereafter referred as to KF) is a component-based framework for Java in which units of resources called bundles can be installed. Bundles can export services or run processes, and have their dependencies managed, such that a bundle can be expected to have its requirements managed by the container. Each bundle can also have its own internal classpath, so that it can serve as an independent unit, should that be desirable. All of this is standardized such that any valid Knopflerfish bundle can be installed in any valid OSGi container (Oscar, Equinox or any other).

Basically, running OSGi is very simple: one grabs one of the OSGi container implementations (Equinox, Felix, Knopflerfish, ProSyst, Oscar, etc.) and executes the container's boot process; much like one runs a Java EE server. Like Java EE, each container has a different startup environment and slightly different capabilities. The KF environment can be downloaded here: http://www.knopflerfish.org/

The KF start-up environment is shown in Figure 5-2:

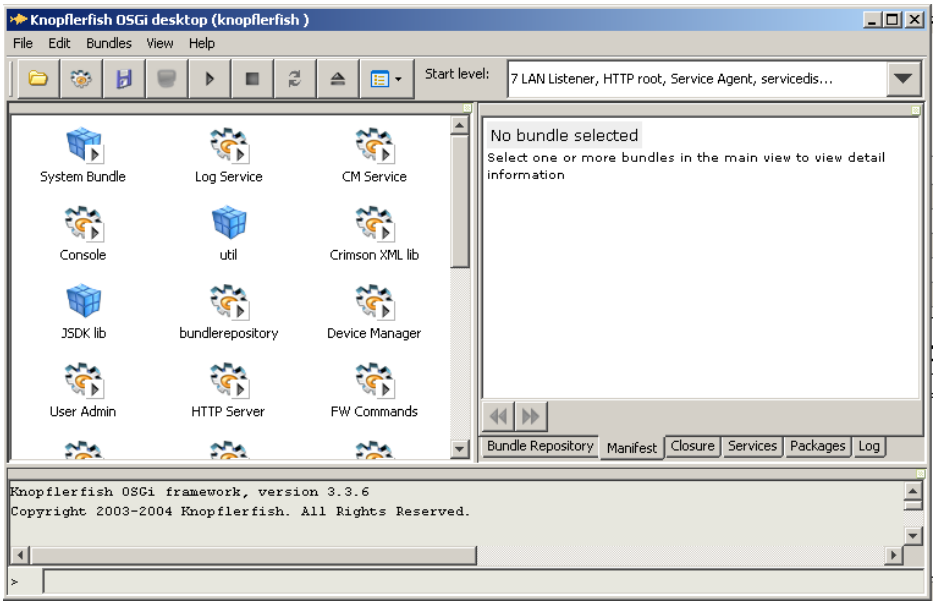

Figure 5-2: Knopflerfish start-up environment

OSGI Middleware SPD features

One of the most important peculiarities of the KF OSGi is that it already offers a standard orchestration environment that, once correctly setup, can act as the SHIELD Orchestration Core SPD Service. Thus the Orchestration functionalities come for free when using an OSGi framework, instead of using other SOA implementations.

The prototype architecture derives directly from the architecture described in the previous section. Each SHIELD Middleware component is mapped into an OSGi bundle and, when needed, decoupled into a composition of interoperating bundles each providing a specific functionality. This modular approach simplify the design, development and debugging of the whole system. Even the Innovative SPD Functionalities have been implemented as OSGi bundles. Each OSGi bundle has its own dependencies, provides a set of functionalities, requires a set of functionalities and is characterized by a specific SPD level. Each bundle can be registered in the Service Registry to advertise itself, to maintain updated its status in order to be discovered. Each bundle can also store its description in the Semantic Database, to be semantically composed. Each bundle interfaces the rest of the architecture providing a set of functionalities and requiring a set of functionalities, exactly as a software component does. More in particular each bundle is decoupled into two parts: the interfacing part (API) and its implementation part (IMPL). This separation between API and IMPL ease the substitution at runtime of a specific bundle, to change from one implementation to another. This substitution can be due, as an example, to the necessity to strengthen the SPD level of a specific functionality.

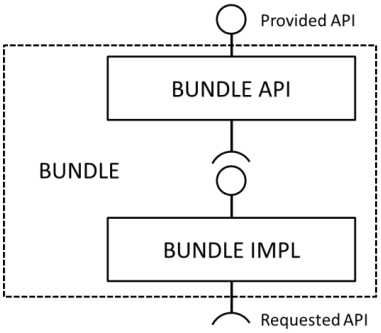

Figure 5-3: Bundle architecture

This framework will be installed on a Laptop and interconnected to the railway server to drive the security functions. At a preliminary analysis, the interaction will be done by means of Ethernet interfaces and through the Intrusion Detection Bundle that is just in the middle between the final system and the OSGI.

5.10 Secure Agent

The Security Agent was delivered as a preliminary prototype in D5.2 [5], and is described in detail in D5.3 [6]. This bundle represents the core of the SHIELD Overlay, since it implements the intelligent control functions and mechanisms to perform the SPD composability. In particular the security agent is able: i) to collect the inputs coming from the two information sources (ontologies and domain library), ii) to extrapolate a model of the system on which the controller is built and iii) to implement the control algorithms that produce the solution for the specific “composability” problem.

This bundle is the major novelty and key element of the SHIELD architecture (Figure 5-3).

Secure Agent SPD features

The inclusion of the Security Agent into the demonstrator is transparent, since it will be a module embedded in the Middleware (OSGI framework) itself. It will interact in a seamless way with the discovery and composition primitives to implement the clock.

5.11 Secure Discovery

The Secure Service Discovery Module was delivered as a preliminary prototype in D5.2 [5], and is described in detail in D5.3 [6]. The main aim of a service discovery protocol, architecture, or session is to find the highest number of services conforming to the query’s requests: for this reason, most of the commonly diffused service discovery protocols (e.g. Service Location Protocol – SLP; Universal Plug and Play – UPnP, etc.) were designed to fulfil this particular task, without paying particular attention to security-related aspects. In other words, most of the commonly diffused service discovery protocols are intrinsically unsecure, which mean they are completely opened to common attacks such as Denial Of Service attacks, Men in the Middle attacks, just to indicate some of them.

Luckily, for some of these protocols security extension have been proposed (e.g. for the SLP), but for some others the debate is still open even to identify if security extension have sense to be considered at all, or, at least, a proper solution was still not proposed for standardization (e.g.: UPnP).

Secure Discovery SPD features

A typical service discovery architecture could be depicted in Figure 5-4:

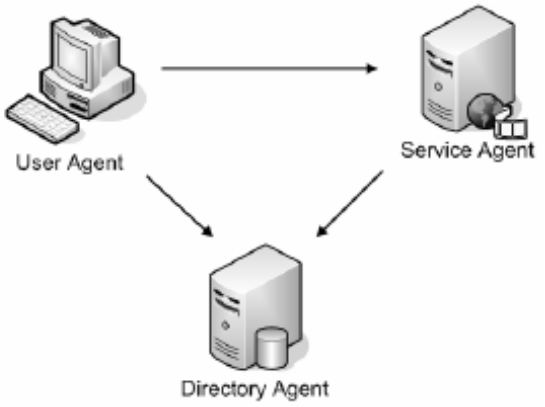

Figure 5-4: Service discovery architecture

In the picture three main entities could be identified, that need to be also present in the demonstrator:

- a Service Discovery Client (User Agent), which initializes the service discovery process: it is the entity interested in finding a certain service
- a Service (Service Agent), which, further being the service (one of the services) to be discovered by the client, is interested to be discovered, in case properly “advertising” itself
- a Service Repository (Directory Agent), which is a sort of database containing all the services that have published themselves in a certain scenarios and which discovery client could actually find.

It should be noticed that some protocols (e.g. UPnP) do not require the presence of a Service Repository, relying on multicast messages or equivalent solutions to perform both the discovery and the advertising phases: in this case the Service and the Service Repository could be considered as “collapsed” in a single entity.

In the considered scenario, securing service discovery means securing the exchange of messages among the Service Discovery Clients and the Service Repositories and the ones among the Services and the Service Repositories. Note that messages exchanged among the Client and the Service are outside of the scope of Service Discovery: when “found”, a certain Service could be used, so that service discovery is ended and messages regarding the usage of the service are not to be considered part of the service discovery process.

This solution is already implemented in the OSGI framework (that includes the Secure SLP Bundle) so the integration with the final system is very simple, and is based on the activation of this bundle in a transparent way. Many candidates protocols for Secure Discovery could be activated in principle, since the architecture is independent from the current implementation.

5.12 Automatic Access Control

Access control mechanisms are in charge of preventing malicious entities to access the physical resources of a network node. Nodes utilize asymmetric cryptography to verify access requests. A DoS attack can be performed if a large number of access requests are sent to exhaust node’s resources. Automatic access control embodies some lightweight features to easy the verification process and avoid the DoS attack. Such features include hash functions, matrix multiplication, pseudorandom number generators and cyclic redundancy check (CRC). For example, a client sends a hashed secret that is already known to the server. The server keeps a map for the hashed secrets for all its clients. Then the server simply looks up for the secret to quickly verify if the node is legitimate and counter the DoS attack efficiently. The failure of some access control requests could be an indication that the system is under attack. The relevant information can be reported to an IDS.

Several automatic access control protocols have been proposed. They try to provide security properties like mutual authentication, forward security and anonymity. Also they try to counter DoS, replay and de-synchronization attacks to the protocol steps as well as link-ability of different communications of the same user.

Gossamer[13] is a protocol for preventing DoS attacks on RFID systems. It belongs to the UMAP (Ultra Lightweight Mutual Authentication Protocol) family of protocols. A reader uses index-pseudonyms to retrieve tag information. Readers and tags share sub keys which are part of a single key. Then these sub keys are used to build the messages exchanged in the mutual authentication process. The protocol is based on bitwise logical operations such as XOR, OR, and AND. The reader generates a pseudorandom numbers and tags use them to create messages. Gossamer is vulnerable to a DoS attack by de-synchronization.

For nSHIELD dependable self-x technologies, we propose the Gossamer protocol. We will apply the protocol in BeagleBones/BeagleBoards. A BeagleBoard acting as a server and a small number of BeagleBones acting as tags will connect to the board. The protocol will be used for node and network protection against DoS attacks.

Automatic Access Control SPD features

Gossamer [13] is an ultra-lightweight protocol for mutual authentication. It makes use of pseudorandom numbers and simple bit operations to effectively provide data confidentiality, tag anonymity, mutual authentication, data integrity, forward security, robustness against replay attacks and DoS attack prevention. A tag and a server use 96-bit keys and nonces (to counter replay attacks). The session data is a triple of an index-pseudonym and two keys. It is updated at each session to achieve forward security. Each tag maintains two triples for the previous and the current session, overall 576-bits, to counter de-synchronization issues. The two entities exchange 4 messages, overall 424-bits, to achieve the aforementioned properties. They result in an index-pseudonym for the tag (96-bits) in the current session that is used as the lightweight feature for the automatic access control by the server.

The individual SPD level of this automatic access control mechanism is static. When it is applied by a network it can efficiently counter a set of DoS attacks. Otherwise, the network is vulnerable. It provides a moderate level of SPD and the overall DoS resilience has to be deduced in conjunction with other protecting mechanisms against these attacks.

Table 5-6: Reputation scheme SPD levels

SPD Level	Functionality
1 (lowest)	No protection against DoS attacks
2 (medium)	The Gossamer protocol
3 (high)	In conjunction with other protecting mechanisms for DoS attacks

5.13 Policy Based Access Control

The SHIELD secure policy-based access control (PBAC) framework facilitates the control of access to devices and their resources via security policies residing on resource-rich infrastructure nodes. It consists of several components that run on different nodes of the nSHIELD architecture. These components are the **Policy Enforcement Points (PEP)**, the **Policy Administration Point (PAP)**, the **Policy Decision Points (PDP)** and the **Policy Information Point (PIP)**. A node, depending on its capabilities and the available resources, might include one or more of these functional components.

The PBAC framework is Devices Profile for Web Services (DPWS) -compliant, utilizing the relevant specifications and existing work to provide message-level security and fine-grained security policy functionality while maintaining interoperability with the standard. The DPWS is the "UPnP[2] for the Internet of Things"; a unified protocol platform developed because of the need to implement dynamic and secure discovery of devices and Web Services (including messaging, description, interactions, event-driven changes etc.) on resource constrained devices and supported by Microsoft and other industry leaders. While UPnP and DLNA (Digital Living Network Alliance) are favored for home entertainment scenarios, DPWS is recommended for enterprise and vertical applications. By adopting a DPWS-compliant mechanism, the PBAC framework offers seamless integration (discovery, access etc.) of new devices into the ecosystem and good scaling.

The solution adopted for secure policy-based access control is based on eXtensible Access control Markup Language (XACML) policies. XACML is an XML-based general-purpose access control policy language used for representing authorization and entitlement policies for managing access to resources and, moreover, an access control decision request/response language. As such, it can be used to convey policy requirements in a unified and unambiguous manner, hence interoperable and secure, if

[2] Universal Plug and Play

appropriately deployed. The above fit well into the model of a network of heterogeneous embedded systems where access to resources is provided by nodes as a service, and into the management architecture developed by IETF Policy Framework. This typical policy based access control architecture combined with XACML is mapped to a Service Oriented Architecture (SOA) network of nodes to provide protected access to their distributed resources.

By combining the above technologies, the PBAC framework allows for fine-grained, policy-based control of all resources (e.g. DPWS-enabled cameras or sensors or control stations) from remote locations, via any compatible app developed for the purpose or even typical browsers and off the shelf mobile phones. Said access may be used to access the resources provided (e.g. sensor data or video stream), update settings or even receive alerts (e.g. in case of emergencies), all based on what the active policy dictates, while various metrics can be reported to the overlay.

Figure 5-5 depicts the integration of the PBAC framework into the Railway scenario System Security Architecture.

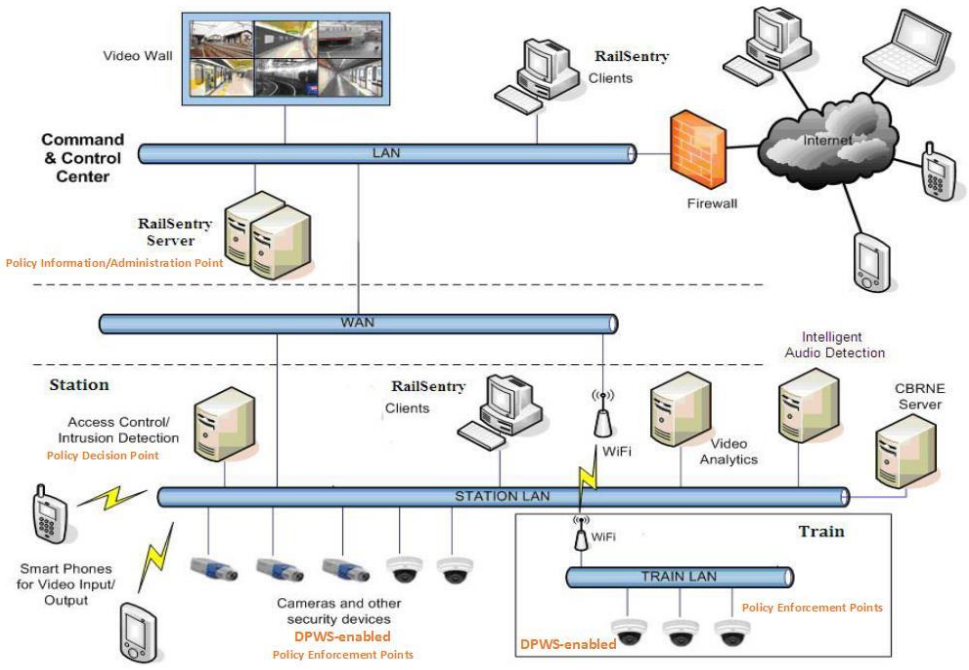

Figure 5-5: The SHIELD secure policy-based access control

Policy Based Access Control SPD features

The aforementioned framework addresses the critical issue of controlling access to nSHIELD resources. Its main features are that (1) it is policy based, hence allows the dynamic change of privileges and the SPD levels, based on the stakeholders' needs and decisions and (2) it provides the capability to directly access nodes' resources and address access requests to them, with no need to be aware of the system's details. In that sense policy based access control only authorized access satisfying the corresponding requirements.

There is a strong relation between the SPD levels of the PBAC framework and the defined policies and the corresponding rules which can be very strict or relaxed based on the system owner's requirements. Note that this policy can be defined either on a node basis, set of nodes, or for the whole system or it can event target specific subjects and resources.

On top of these policy-based levels there are some features available that affect the protection of the framework itself and therefore its effectiveness. These mechanisms are the encryption of messages and

their authentication. Unprotected messages can disclose access control related messages and make them subject to unauthorised modifications. Therefore, the SPD levels of the Policy Based Access Control mechanism are shown in Table 5-7.

Table 5-7: PBAC SPD levels

SPD Level	Functionality
1 (low)	No encryption – No Authentication-
2 (medium)	Encryption or Authentication
4 (high)	Encryption + Authentication

5.14 Reputation Based Secure Routing

To meet the technical necessities of effectively securing the communication framework in nSHIELD network layer, we are implementing a novel module reputation-based scheme that can act as a general purpose scheme for a wide range of applications, where wireless PANs and WSNs are involved. The objective is to increase network's SPD levels, optimize its performance and strengthen system's protection against malicious perpetrators and network security attacks, such as Denial of Service. Thus, part of the security requirements imposed by the composite heterogeneous nSHIELD network serving the railway application scenario, will be successfully accomplished by the Reputation-based Security scheme.

The abstract backdrop is a Secure Routing Framework, based on a number of interconnected components. This is followed by the description of a Trusted Routing scheme, implementing Direct and Indirect Trust over a geographical routing protocol (Greedy Perimeter Stateless Routing, GPSR) and an Intrusion Detection System (IDS) implemented on a wireless sensor node distributed architecture. Direct Trust scheme is the simplest form of Trust establishment, as it takes into account only first-hand observations of the cooperative interactions (e.g. packet forwarding for secure routing), which can lead to increased detection time of attacks. Indirect Trust scheme adds Reputation calculations from other nodes, which can lead to a more complete view of node behaviour and shorten the time needed to learn a node's trustworthiness compared to the scheme implementing only Direct Trust.

Reputation Based Secure Routing SPD features

Reputation based IDS employing a Bayesian formulation, specifically a Beta reputation system for reputation representation can be considered the solution providing the highest SPD level, using only third party information of high integrity and broadcasting alerts, when non-trustworthy behaviour is detected. The three achieved SPD levels are presented in the Table 5-8.

Table 5-8: Reputation scheme SPD levels

SPD Level	Functionality
1 (lowest)	-
2 (low)	DT (Direct Trust)
3 (medium)	Weighted DT (Direct Trust) + ID (Indirect Trust)
4 (high)	Reputation based IDS algorithm

The reputation scheme applies in the scale of wireless Nano or Micro nodes, whether these are cameras (e.g. image sensors) or sensors (e.g. motion detection sensors). The integration process requirements are application specific, while in parallel there are technical prerequisites ensuring the compatibility of a WSN island and its reputation scheme software, in the use case environment. The application needs for interoperability will determine the sensor cloud's interconnection with the rest of the system. More specifically, if the sensors need to be accessed directly or they will constitute a monitoring island that will transmit information through dedicated gateways. In order for any nodes to be integrated and communicate, in each time nSHIELD system abstraction, they have to share at least the basic protocol stack layer components (physical, data link and network).

The reputation-based scheme is expected to advance nSHIELD platform's functionality, especially in the areas concerning wireless communications between medium to small devices. First priorities are network security and secure data transmission. A trust management structure assists nodes to configure and protect themselves against neighbouring topology changes. This is ensured with the avoidance of malicious nodes and the guarantee of a trustworthy route to each destination. Reversely and on the positive side, trusted routing helps nodes in selecting trusted neighbours with which they can share a trustworthy link to route packets. Consequently, network reliability is optimized. As explained above, selecting between one of the three reputation structure implementations, enables us to adjust the desired SPD level of the respective nSHIELD subsystem, according to conditions or each time needs. The reputation module contributes in access control and intrusion detection, primary SPD objectives of the railway scenario smart surveillance system. The former can be performed with several types of RFID equipment, while the latter can be achieved with the selection of the appropriate sensing modalities like volumetric sensors and active/passive infrared detectors.

The Figure 5-6 depicts the coexistence of surveillance cameras and access control sensors (and their connection to nSHIELD control workstations and middleware) in example architecture, in a monitored station area.

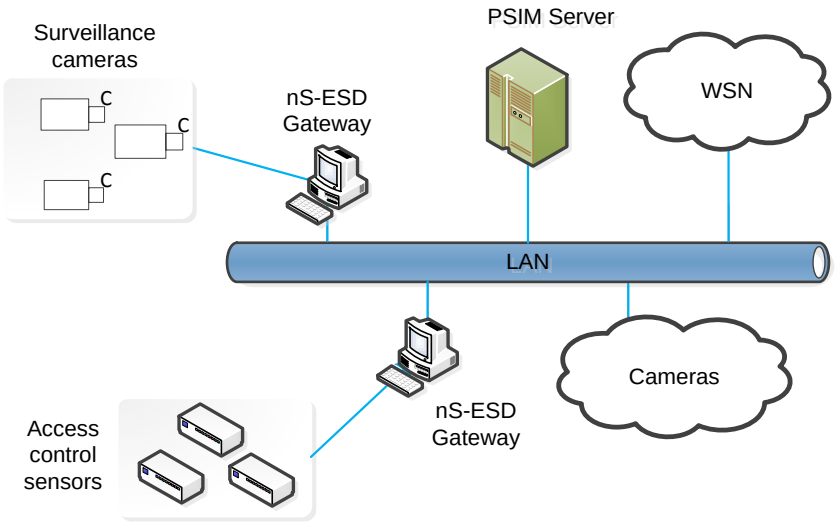

Figure 5-6: WSNs and Cameras connected to nSHIELD Middleware

5.15 Smart card based security services

To build trust among different types of nodes on the nSHIELD architecture we can exploit the benefits of smart cards and the cryptographic schemes they implement. In the nSHIELD architecture where decentralized components are interacting not only with each other, but also with centralized ones, there is a need for integrating security and interoperability. In this context, we exploit the advantages of smart cards to enable different types of nodes to provide the following security services:

- Allow secure key management required for establishing secure channels between different nodes.
- "Anonymous" Authentication e.g., between the sensor and central or other distributed components in the train network.
- Protecting message integrity, for sensor data in the train network among the node and the central system.

This way, the smart card component can be used to eliminate the following general threats:

- Repetition (a message is received more than once)
- Insertion (a new message is implanted in the message stream)
- Re-sequencing (messages are received in an unexpected sequence)
- Corruption (the information contained in a message is changed, casually or not)
- Masquerade (a non-authentic message is designed thus to appear to be authentic)

More particularly, the smart card module can be used to minimize the risk of the railway information system as illustrated in Table 5-9.

Table 5-9: Reputation scheme SPD levels

Railway Component	Threat	Security service for threat mitigation using smart card
Ethernet, WiFi Camera	Unauthorized network access	Authentication Integrity
Application server	Unauthorized access Loss of data integrity	Integrity Authentication
Client operator / video wall	Unauthorized network access Data alteration	Authentication Integrity
Mobile client	Unauthorized network access due to data destruction/alteration	Authentication/Integrity

Note that using smart card for nodes in the train network where no security mechanism have been deployed correspond to no modifications in the current infrastructure.

6 nSHIELD Railway security use cases

Starting from the description of the reference architecture for the case-study, in this section we provide some use cases proposals to be discussed for possible demonstrations.

6.1 Scenario n. 1

Let us refer to cameras used to monitor the main entrance of a shelter. In order to improve reliability, two cameras (CAM A and CAM B) are installed for redundancy in a cluster-like architecture, with identical position and SPD characteristics (Figure 4-1).

The following is a possible use case:

1. CAM A works properly.

   SHIELD set Reliability_level=X

2. Cam A fails

3. SHIELD allows to automatically detect the failure and reconfigure the system such that

   a. Cam B replaces Cam A in a transparent manner (e.g. the IP to which the box on the video-wall points changes so quickly that the user hardly realizes).

   b. System dependability level decreases, **Reliability_level=X-delta**

Note: if CAM B fails while CAM A is still working, no reconfiguration happens but administrator is notified of the decrease in the dependability level.

6.2 Scenario n. 2

This scenario refers to video surveillance from a control room of a vehicle (Figure 4-1) and proposes the need to adapt to different wireless network connections (Wi-Fi, GSM/GPRS, UMTS, etc.) and signal conditions as the vehicle moves through the city, since no fixed infrastructure with assured QoS is assumed. Furthermore, there will be wired connections in the depot for daily downloading of log-files and recordings (for back-up or long term video archiving). Different technologies and configuration/encryption options (WEP, WPA, etc.) will obviously feature different SPD requirements.

In such a scenario SHIELD would reconfigure SPD properties depending on locations (depot, tunnel, etc.) and other measured parameters (network connection, estimated bandwidth and S/N ratio) considering that public networks (mobile or wireless) do not assure QoS or SPD.

Of course the intra-vehicle (wired) network connecting on-board devices is dedicated and segregated, therefore featuring potentially higher and controllable QoS and SPD.

Example of demonstration of Middleware services

In this section will be shown an example of Scenario 2 steps to demonstrate Middleware functionality:

1. The vehicle is connected by wire and has high SPD level (secured connection)

2. Nodes communicate with middleware services via IDS (communications are known-to-be-good, so overlay functionality adds endpoints to the white-list of IDS, e.g. based on Secure Discovery)

3. Vehicle starts, changes to public network (e.g. UMTS), SPD level decreases (Middleware now connects to a network from which malicious connections can originate as well)

4. Previously known nodes should repeat discovery because of change of network addresses

5. IDS operates normally and forwards all requests as long as server load is acceptable (compared to a preset value)

6. Middleware services / Overlay still recognize trusted nodes based on discovery and continue whitelisting (based on changed network address)

7. Then, malicious requests start flooding Middleware from network (DoS / DDoS)

8. IDS recognizes Critical Load on Server, and changes status, notifies Overlay (SPD level may change accordingly)

9. IDS may switch to white-listed operation, dropping all requests originating from unknown addresses while server load is critical

10. *OR* IDS may blacklist addresses which cause overload (based on request statistics), and drop blacklisted requests while server load is critical

7 Railway security System demonstrator integration

The general integration methodology adopted can be found in the D6.3 [12] in which is described the integration plan for the nSHIELD framework. In particular for the Railway Security Scenario, there are some prototypes involved and the integration plan provide the interface to be implemented

In Figure 7-1 (reported again only for convenience) is possible to see the interfaces between the different components of scenario architecture. The links are in red and black. The black link are interfaces already existed and implemented, the red link are new interfaces that will be implemented in order to construct the final demonstration scenario.

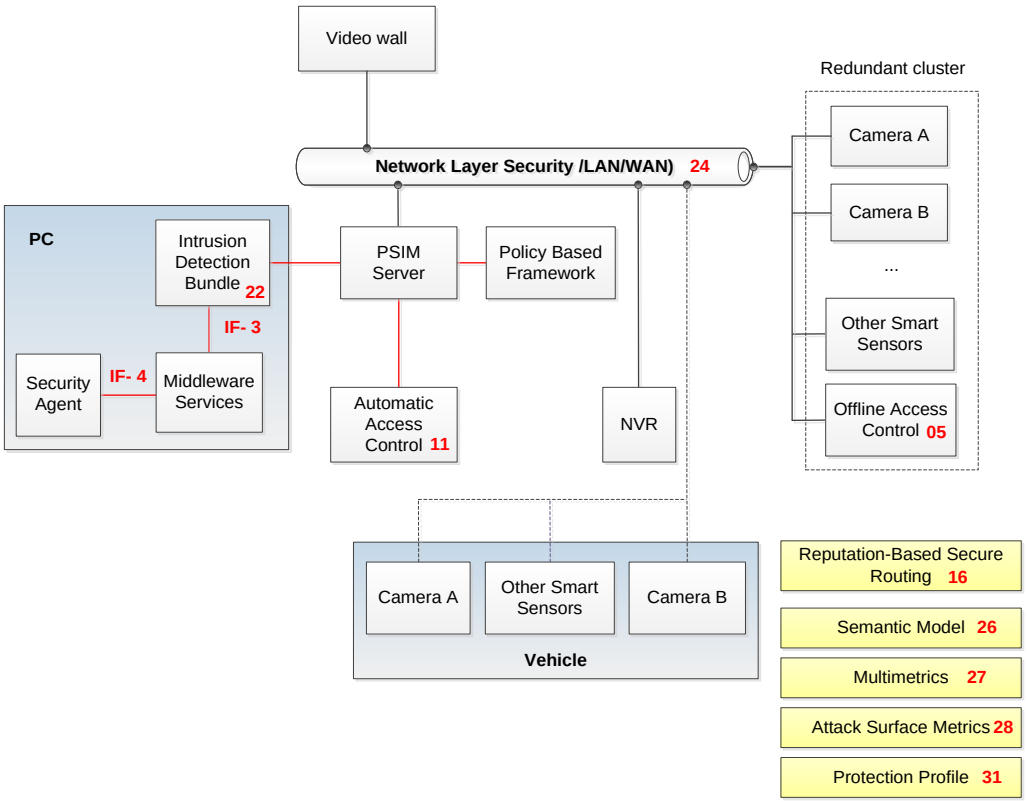

Figure 7-1: Interfaces of reference architecture

In the following list will be described the planning for some interfaces implementation.

Intrusion Detection Bundle and PSIM Server/Middleware Services: this is a typical network interfaces for receiving and forwarding requests that are to be filtered. The IDS prototype will receive and optionally forward messages without altering their content or re-encapsulating them. It is a homogeneous interface between the Middleware services and the PSIM Server. It is however anticipated that TAP / TUN virtual network interfaces could be used to physically separate and protect internal (Middleware services) and external (other components and networks besides Middleware) network domains. These changes could mostly be implemented in a transparent manner for the system components using middleware services, but may impact how connection methods towards middleware services should be implemented. The design of the network domains and the connection methods used will be studied at the time of integration with other Middleware components.

Reputation based Secure Routing: For the integration of reputation-based secure routing prototype in the railway scenario, network interfacing is the first issue needed to be solved. This of course is true in the case that secure routing is running on 802.15.4 wireless network of embedded device as is the case of Prototype 16. From an architectural point of view the kind of network that secure routing prototype runs

can be considered a legacy network and a gateway device must be included in order to communicate with the L-ESD devices.

Offline Access Control: Telcred (TELC), Acorde (AT), and SICS are collaborating on developing a secure micro node, which can be used as a lock controller. A custom "cape" for a standard BeagleBone low end Linux computer is being developed. This cape will provide features such as tamper detection, backup power, secure storage of cryptographic keys, and a real time clock. This system will be used as an offline physical access control and the secure lock controller will operate offline/standalone. This means that, at the beginning, no interfaces with other nSHIELD devices/ components will be implemented.

Network Layer Security: The network layer prototype has been implemented in the Contiki OS and is meant for securing communication between nano nodes and other nSHIELD nodes. The implementation has taken place within the Contiki's uIP stack, which is responsible for handling the incoming/outgoing traffic of a node.

Metrics Approach: This integration approach will be held by incorporating the aggregation formula to OSGI Middleware governing nSHIELD Overlay, and in particular by embedding it in the semantic model used in the SHIELD framework. This middleware will enable a container for aggregation formula and/or algorithm. However it must be understood that both approaches have to be tuned by operator experts, so that integration will be finished with both perspective: this one which will be automatically addressed and one more manual one with the opinion of experts.

Automatic Access Control: Its function interfaces are implemented in C++ and implements the ultra-lightweight protocol for automatic access control and mutual authentication – Gossamer. The code is tested under the operating system Linux. The core functionality is implemented in the Linux kernel and rest functionality in the user space. The compilation creates a module in the Linux framework Netfilter, which is responsible for manipulating the out/incoming network traffic of a node.

Policy Based Management Framework: Refer to section 4.12 in D 6.3.

Semantic Model: The integration of this prototype in the common platform is done in two ways:

- By providing the components' responsible with the "guidelines" to write down the Ontology model for their component as well as the Domain Dependent Library
- By codifying this ontology into an xml file that can be parsed by the OSGI to extrapolate relevant information.

For the sake of simplicity, the demonstrator could be set up with the semantic data bases already initialized (in the real system this "learning" phase will be done at the "switch on").

OSGI Middleware: Knopflerfish (hereafter referred as to KF) is a component-based framework for Java in which units of resources called bundles can be installed. Bundles can export services or run processes, and have their dependencies managed, such that a bundle can be expected to have its requirements managed by the container. Each bundle can also have its own internal classpath, so that it can serve as an independent unit, should that be desirable. All of this is standardized such that any valid Knopflerfish bundle can be installed in any valid OSGi container (Oscar, Equinox or any other).

All the major nSHIELD Middleware services have been translated into specific Bundles (including the Security Agent) so that the platform is representative enough of the final system.

On a deployment point of view, this framework is installed into a Notebook that is interfaced directly with the Intrusion Detection Bundle and consequently with the rest of the railways demonstrator through a network (most likely an Ethernet LAN). The interfaces with the Secure Discovery Bundle (in charge of populating the service databases) and the Security Agent are internal and implemented directly in Java Language.

Security Agent: The Security Agent is one of the main OSGi bundle and is responsible of interfacing the control algorithms with the discovery module, i.e. it represent the “embedded intelligence” of the SHIELD framework. Since the Security Agent is an OSGI bundle, no integration issues are foreseen, since it is native in the middleware prototype itself.

Secure Discovery: The service discovery client and repository will be implemented in the OSGI framework, while the service agent will be installed into the railways demonstrator PSIM server. The specific discovery protocol adopted for the demonstration purposes is the SLP protocol.

Protection Profile: Considering this PP definition it is evident that it is a particular type of prototype which is completely divorced from the speech of the integration of the prototypes. On the contrary, it makes the rules or rather the SPD requirements that must be met by prototypes Integration that make up an embedded system aiming to be SHIELD compliant (as indicated above, at this time are shown only the SPD requirements that the middleware of the system must meet).

8 Railway security System demonstrator validation and verification

8.1 Validation and Verification methods

The IEEE Guide to the Project Management Body of Knowledge[3] defines validation and verification as follows:

- "Validation. The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with verification."
- "Verification. The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with validation."

The proposed method of Validation and verification is applying relevant parts of the V-Model of project management used by several national standards [9]. This model clearly identifies the distinct steps during all stages of the development process – from requirements and design specification to implementation, testing, and operation.

In the case of nSHIELD project, the following methodology is proposed. Activities are identified together with deliverables relevant to that stage, where outcomes are described. As per the V-Model, the overall architecture must be designed to be testable. Specifically, all design elements and acceptance tests must be traceable to design requirements – similarly, each design requirement must be addressed by at least one design element and a corresponding acceptance test.

Demonstrator Validation and Verification plan (the scope of the current document) deals with the following activities:

- Validation of presented demonstrator scenarios against High level requirements for scenarios – ensuring that the original requirements were adequately covered and demonstrated by use cases.

  Stakeholders involved in Validation and Verification activities:

  - Demonstrator owner as responsible for requirements and scenarios

  Means of validation / verification:

  - Analysis of requirements vs. Demonstrator scenarios

  Justification of prototype, integrated prototype, and platform level validation and verification results in the scope of the current Demonstrator – ensuring that the Demonstrator components were developed as planned and all the lower level requirements were properly implemented. Means of validation / verification:

  - Analysis of former (lower-level) validation and verification results, tracing results from prototypes and relevant functionality

- Verification of Demonstrator scenario execution – verifying that the specified use cases (derived from Demonstrator Owner use cases relevant to high level requirements) were executed according to specifications.

[3] IEEE Guide--Adoption of the Project Management Institute (PMI®) Standard A Guide to the Project Management Body of Knowledge (PMBOK® Guide)--Fourth Edition". p. 452. doi:10.1109/IEEESTD.2011.6086685. (http://ieeexplore.ieee.org/servlet/opac?punumber=6086683)

Stakeholders involved in Validation and Verification activities:

- Demonstrator owner and partner(s) responsible for scenarios
- Prototype owners and integrators responsible for implementation of relevant functionality

Means of validation / verification:

- Comparison of Test results vs. description of scenario (expected behaviour)

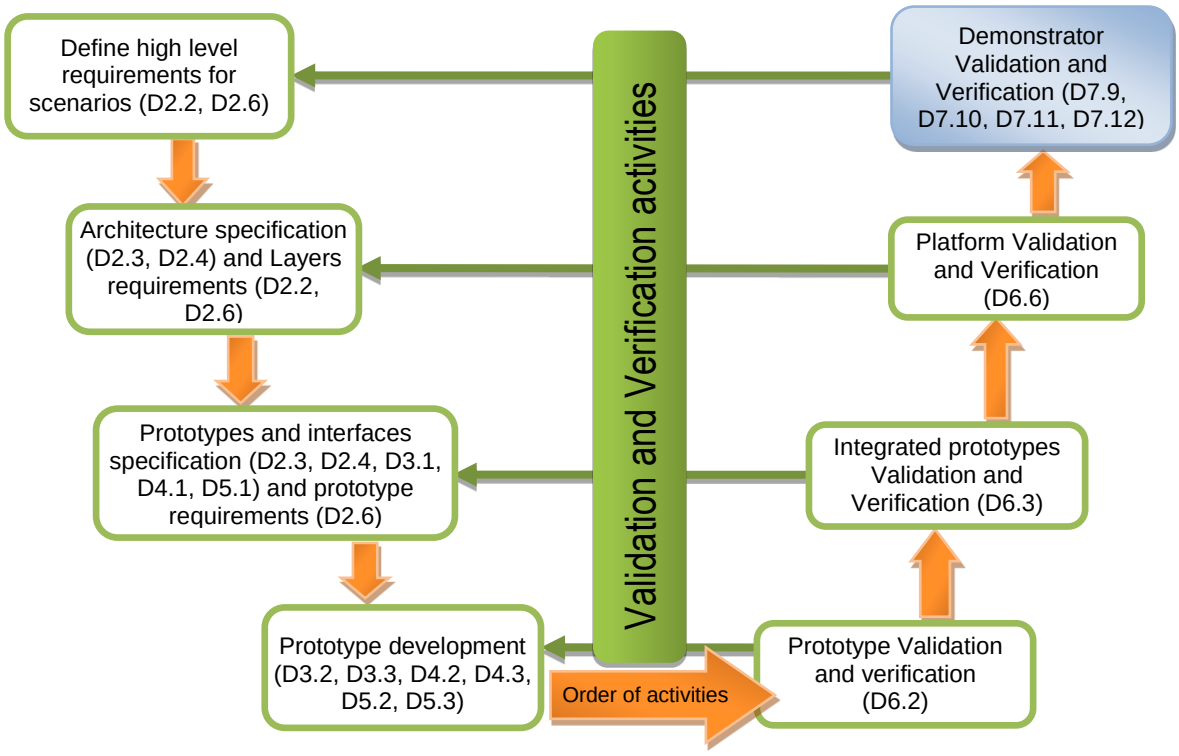

Figure 8-1: Validation and verification activities

The following chapter's list information that is available at the current design and development stage about the activities as listed above. The Validation and Verification results will be described in detail in the nSHIELD deliverable D7.9 Railways Security - Validation and Verification Report.

8.2 Example of Validation process for demonstrator scenarios

In this section is shown an example of validation process for scenario 2. The same can be considered for other scenarios. The requirements for each scenario are listed in D2.2 chapter 5.1, this is only an example, that can be developed in the future reports.

Scenario n.2

Description of relevant requirements from [8].

Table 8-1: Relevant requirements of scenario n.2

High Level Requirements for Scenario	Use case steps (from Chapter 6)	Description
REQ_RW09 Denial of service	Scenario 2 steps 7-10	Demonstration for protection of integrity of Middleware services by IDS prototype in DoS situation

8.3 Justification based on prototype and platform Validation and Verification

8.3.1 Validation and verification results for prototypes

In this section will be summarized how results of D6.2 validate that prototypes included in current demonstrator are fit for the purpose of the demonstrator.

8.3.1.1 (20) Control Algorithm

See chapter 6.3.4 in nSHIELD, D6.2: Prototype validation and verification Plan [7]

Relevant requirements being validated from nSHIELD, D2.2: Preliminary System Requirements and Specifications [8]:

- REQ_MW15 Configurations definition
- REQ_MW16 Configurations quantification
- REQ_MW17 Configurations selection

8.3.1.2 (22) IDS prototype

See chapter 5.3.5 in nSHIELD, D6.2: Prototype validation and verification Plan [7].

Relevant requirements being validated from nSHIELD, D2.2: Preliminary System Requirements and Specifications [8]:

- REQ_SH02 Information transmission integrity
- REQ_SH12 SPD level assignment
- REQ_SH33 Automated testing tools
- REQ_MW7 Information filtering for intrusion detection
- REQ_MW10 Interoperability

The relevant requirement towards IDS prototype for the Railway Scenarios in [8] is REQ_RW09 Denial of service, which can be traced to Middleware requirement REQ_MW7 Information filtering for intrusion detection, which is validated for the prototype.

8.3.1.3 (24) Network Layer Security

See chapter 4.2.6 in nSHIELD, D6.2: Prototype validation and verification Plan [7].

Relevant requirements being validated from nSHIELD, D2.2: Preliminary System Requirements and Specifications [8]:

- REQ_NW01 Confidentiality
- REQ_NW02 Integrity
- REQ_NW06 Multiple Protocol Support

- REQ_NW08 Network Security Cryptographic Support
- REQ_NW19 Application-Based Configurability
- REQ_NW20 Low Network Delay

8.3.1.4 (25) OSGI Middleware

No specific requirements have been foreseen for the OSGI framework, since it is a consolidate heritage of the pSHIELD project, so there was no need to specify it again as a precondition.

8.3.1.5 (05) Secure Power (&) Communication cape

See chapter 3.3.6 in nSHIELD, D6.2: Prototype validation and verification Plan [7].

8.3.1.6 (31) Protection Profile

See chapter 5.2.7 in nSHIELD, D6.2: Prototype validation and verification Plan [7].

8.3.1.7 (26) Semantic Model

See chapter 6.3.1 in nSHIELD, D6.2: Prototype validation and verification Plan [7].

Relevant requirements being validated from nSHIELD, D2.2: Preliminary System Requirements and Specifications [8]:

- REQ_MW6 Information retrieving
- REQ_MW9 Data management
- REQ_SH16 Data backup
- REQ_SH17 Data storage redundancy
- REQ_SH18 Data storage integrity
- REQ_SH19 Data storage confidentiality

8.3.1.8 (27) Multi-metrics

No specific requirements

8.3.1.9 (28) Attack Surface Metrics

No specific requirements

8.3.1.10 (33) Secure Agent

See chapter 6.3.6 in nSHIELD, D6.2: Prototype validation and verification Plan [7].

Relevant requirements being validated from nSHIELD, D2.2: Preliminary System Requirements and Specifications [8]:

- REQ_MW5 Orchestration and choreography
- REQ_MW3 Composition
- REQ_MW4 Secure/Trusted Composition
- REQ_MW8 Enforcement)

8.3.1.11 (32) Secure Discovery

See chapter 6.3.2 in nSHIELD, D6.2: Prototype validation and verification Plan [7].

Relevant requirements being validated from nSHIELD, D2.2: Preliminary System Requirements and Specifications [8]:

- REQ_MW1 Discovery
- REQ_MW2 Secure Discovery
- REQ_MW11 Non-repudiation of origin for secure service discovery, composition and delivery protocols
- REQ_MW12 Non-repudiation of receipt for secure service discovery, composition and delivery protocols

8.3.1.12 (11) Automatic Access Control

See chapter 3.3.8 in nSHIELD, D6.2: Prototype validation and verification Plan [7].

Relevant requirements being validated from nSHIELD, D2.2: Preliminary System Requirements and Specifications [8]:

- REQ_ND02 Data Freshness

8.3.1.13 (19) Policy Based Access Control

See chapter 5.2.6 in nSHIELD, D6.2: Prototype validation and verification Plan [7].

Relevant requirements being validated from nSHIELD, D2.2: Preliminary System Requirements and Specifications [8]:

- REQ_MW13 Access Control Policies for middleware components
- REQ_MW14 Access Control Functions for middleware components

8.3.1.14 (16) Reputation based secure routing

See chapter 4.3.2 in nSHIELD, D6.2: Prototype validation and verification Plan [7].

Relevant requirements being validated from nSHIELD, D2.2: Preliminary System Requirements and Specifications [8]:

- REQ_NW03 Secure Routing
- REQ_NW04 Fault Tolerance
- REQ_NW05 Self-Management and Self-Coordination
- REQ_NW07 Availability
- REQ_NW11 Reputation-Based Secure Routing
- REQ_NW12 Reputation-based intrusion detection
- REQ_NW19 Application-Based Configurability
- REQ_NW20 Low network delay

8.4 Verification of Demonstrator scenario execution

8.4.1 Tools and platforms for execution of Demonstrator scenarios

8.4.1.1 Validation and Verification tools for IDS prototype

Chapter 5.3.5 in nSHIELD, D6.2: Prototype validation and verification Plan [7] lists validation steps that are carried out using code for automated testing of Intrusion Detection and Filtering Module:

- Basic functionality test (SendReceiveTest)
- load generation (CriticalLoadTest)
- Information filtering tests (BlackListTest and WhiteListTest)

The testing code referred here is described in [6] and included as source code in [5]. These test cases are available for use in the Railway Demonstrator scenarios developed, or may be used as templates to implement Validation and Verification test cases executed automatically in the Scenarios.

8.4.2 Other HW and SW resources for execution of Demonstrator scenarios

Please list resource to be used in verification of the Demonstrator scenarios, such as measurement, automated testing and input generation, logging, etc. tools, devices, and software.

- IDS prototype uses logging functionality provided by the OSGI Framework (Java package ‘org.knopflerfish.log’)

9 Conclusions

In this document, we have presented the integration and validation plan for the Railway Security demonstrator.

Starting from the SPD requirements of a reference railway security application, we have described the nSHIELD solution proposed to address the real-world issues related to the reference application. For the sake of a full comprehension of the SPD issues, we have also described in details the prototypes involved in the demonstrator and their SPD functionalities.

After describing the real-world use-cases for the railway security scenario demonstration, we have presented the integration and V&V plan, in order to demonstrate and validate the nSHIELD framework in the reference application.

10 References

[1] European Commission - Guide to successful communications, 2004 http://ec.europa.eu/research/science-society/science-communication/index_en.htm

[2] Aniketos FP7-funded project: http://www.aniketos.eu. For project description, see project presentation at http://aniketos.eu/content/project-presentation-slides

[3] PROSPER: Provably Secure Execution Platforms for Embedded Systems, http://www.sics.se/projects/prosper

[4] Global Platform, http://www.globalplatform.org/

[5] nSHIELD, D5.2: Preliminary SPD Middleware and Overlay technologies prototype

[6] nSHIELD, D5.3: Preliminary SPD Middleware and Overlay technologies prototype Report

[7] nSHIELD, D6.2: Prototype validation and verification Plan

[8] nSHIELD, D2.2: Preliminary System Requirements and Specifications

[9] V-Model - http://en.wikipedia.org/wiki/V-model

[10] nSHIELD Middleware Protection Profile - nSHIELD Project - PP1.0 - 24.5.2013

[11] nSHIELD, D2.5: Preliminary SPD Metrics Specification

[12] nSHIELD D6.3: Prototype integration Report

[13] Deepak Tagra, Musfiq Rahman, Srinivas Sampalli: Technique for Preventing DoS Attacks on RFID Systems. In 2010 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pages 6-10. (2010)

[14] Bocchetti, G., Flammini, F., Pappalardo, A., Pragliola, C.: Dependable integrated surveillance systems for the physical security of metro railways. In: Proc. 3rd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2009), 30 August - 2 September, 2009, Como (Italy): pp. 1-7

[15] Flammini, F., Gaglione, A., Ottello, F., Pappalardo, A., Pragliola, C., Tedesco, A.: Towards Wireless Sensor Networks for Railway Infrastructure Monitoring. In: Proc. ESARS 2010, 19-21 October 2010, Bologna, Italy: pp. 1-6

[16] Casola, V., Esposito, M., Mazzocca, N., Flammini,F.: Freight Train monitoring: A Case-Study for the pSHIELD Project. In: IEEE Proc. Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012: pp. 597-602

[17] Flammini, F., Gaglione, A., Mazzocca, N., Pragliola, C.: Optimization of Security System Design by Quantitative Risk Assessment and Genetic Algorithms. In: International Journal of Risk Analysis and Management (IJRAM), Vol. 15, No. 2/3, 2011: pp. 205-221

Appendix A Interface Control Documents

This section reports the detail of the interfaces exposed by each component, identified in section 4 (a subsection for each ICD). A template is provided in the following examples.

A.1 Interface Control Document Automatic Access Control

A.1.1 Introduction

For nSHIELD, we implement the Gossamer protocol [13] in C++ and apply it on Memsic IRIS and BeagleBone devices. The two entities communicate through the network with sockets.

A.1.2 Protocol Formats

A tag and a server use 96-bit keys and nonces (to counter replay attacks). The session data is a triple of an index-pseudonym and two keys. It is updated at each session to achieve forward security. Each tag maintains two triples for the previous and the current session, overall 576-bits, to counter de-synchronization issues.

A.1.3 nS-DI – nSHIELD Data Interchange

The two entities exchange 4 messages, overall 424-bits. They result in an index-pseudonym for the tag (96-bits) in the current session that is used as the lightweight feature for the automatic access control by the server. The figure below illustrates the data interchange process of the protocol. 'π' has the value 0x3243F6A8885A308D313198A2. 'MixBits' is a very lightweight function that mixes bits and significantly increases the security of the protocol. 'ROT' is a function that performs left rotation. The 'n_*' are random nonces and the 'k_*' are keys. The n_1 and n_2 are computed by the server (the reader of the figure). The n_3, $n_{1'}$ and $n_{2'}$ are computed by both entities. The k1 and k2 are known in both entities prior to communication. The k_{1*}, k_{2*}, k_{1next} and k_{2next} are computed by both entities.

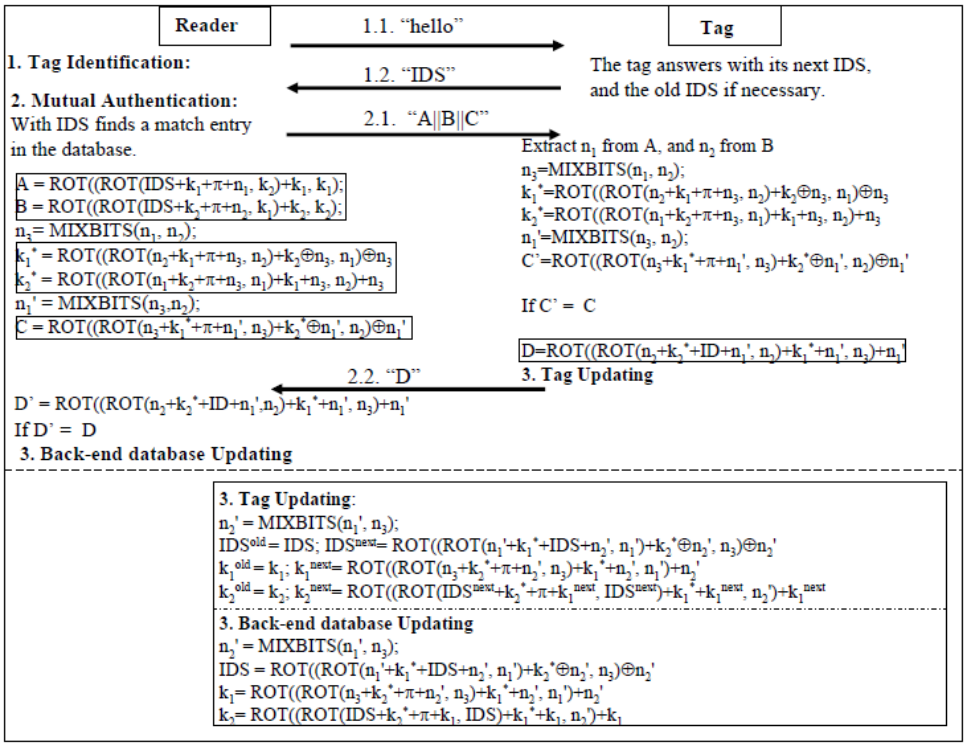

Figure A-1: The Gossamer protocol [13]

A.1.3.1 nS-DI Message Format

The two entities communicate through the network with sockets. The messages are sent/received based on the IP addresses and port numbers that each entity has bind for the protocol. The data of each packet contains the relevant information that is mentions in the above figure.

Message Types

For the identification phase, a 'hello' message (40-bits) and the old IDS (96-bits) are transmitted. For the authentication phase two messages containing the A||B||C (288-bits) and D (96-bits) are transmitted.

Destination/Source IDs

Every ID, key and nonce is a 96-bit value to comply with the EPCGlobal specification. Initially each tag possesses a static ID, an initial index-pseudonym (IDS) which is used by the server for indexing and two keys (k_1, k_2). Prior to the first communication both the server and the tag know these four values (the server maintains a database with all relevant information for all tags, the tag stores the value in its memory). If a communication is successfully completed, the IDS, k_1 and k_2 will be updated. Thus, we achieve forward security and prevent replay-attacks. Also, the current values of IDS, k_1 and k_2 are stored and indicated as 'old'. The old values can be used in the next communication in case of de-synchronization of the tag and the server.

A.1.3.2 Message Type Description

Based on the figure above, the message 1.1 is a simple message that contains the string 'hello' and denotes the initiation of a new communication process. The message 1.2 contains the current index-pseudonym (IDS) that will is used by the server to instantly recognize the tag. The message 2.1 is the concatenation of the values A, B and C (A||B||C). 'A' and 'B' are derived by the current values of k_1, k_2, n_1 and n_2. 'C' is derived by the new k_1*, k_2*, n_1' and n_3. The message 2.2 contains the 'D' parameter which is derived by A, B, and C.