Project no: 269317

nSHIELD

new embedded Systems arcHItecturE for multi-Layer Dependable solutions

Instrument type: Collaborative Project, JTI-CP-ARTEMIS

Priority name: Embedded Systems

D6.2 Prototype validation and verification

Due date of deliverable: M20 –2013.04.30

Actual submission date: M20-2013.04.30

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

Organisation name of lead contractor for this deliverable:

Selex ES, SES

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)	X
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Authors and Approvals

Authors		Date	Signature
Name	**Company**		
Andrea Morgagni	SELEX ES		
Andrea Fiaschetti	UNIROMA1		
UNIROMA1 Team	UNIROMA1		
Kiriakos Georgouleas	HAI		
Nikolaos Pappas	HAI		
George Dramitinos	ISD		
Balazs Berkes	S-LAB		
Gergely Eberhardt	S-LAB		
Lorena de Celis	AT		
Jacobo Domínguez	AT		
Carlo Pompili	TELC		
Kostas Fysarakis	TUC		
George Hatzivasilis	TUC		
Kostas Rantos	TUC		
Alex Papanikolaou	TUC		
Harry Manifavas	TUC		
Kresimir Dabcevic	UNIGE		
Christian Gehrmann	SICS		
Viktor Do	SICS		
Hans Thorsen	T2DATA		
Antonio Di Marzo	SESM		
Antonio Bruscino	SESM		
Ester Artieda	INDRA		
Kyriakos Stefanidis	ATHENA		
Andreas Papalambrou	ATHENA		
Panagiotis Soufrilas	ATHENA		
Paolo Azzoni	ETH		
Stefano Gosetti	ETH		
Reviewed by			
Name	**Company**		
Andrea Fiaschetti	UNIROMA1		
Nikolaos Pappas	HAI		
Approved by			
Name	**Company**		

Applicable Documents		
Issue	**Date**	**Description**
[01]	TA	nSHIELD Technical Annex
[02]	D3.1	SPD Nodes Technologies Assessment
[03]	D4.1	SPD Network Technologies Assessment
[04]	D5.1	SPD Middleware and Overlay Technologies Assessment
[05]	D3.2	Preliminary SPD Nodes Technologies Prototype
[06]	D4.2	Preliminary SPD Network Technologies Prototype
[07]	D5.2	Preliminary SPD Middleware and Overlay Technologies Prototype
[08]	D2.2	Preliminary Systems Requirements and Specifications
[09]	D4.3	Preliminary SPD network technologies prototype report

Modification History		
Issue	**Date**	**Description**
Draft 0	21/01/13	First version of ToC
Draft 1	16/02/13	Definition of purpose and contents of deliverable
Draft 2	17/02/13	Node Prototypes Validation and Verification – Audio surveillance system
Draft 3	27/02/13	Node and middleware Validation and Verification
Draft 4	28/02/13	Network Prototypes Validation and Verification - Reputation-Based Secure Routing prototype
Draft 5	13/03/13	Middleware Prototypes Validation and Verification- SHIELD Middleware Protection Profile
Draft 6	19/03/13	Node prototype – Secure power (&) communication cape
Draft 7	04/04/13	Network Prototypes Validation and Verification – SPD-driven Smart Transmission Layer
Draft 8	19/04/13	Node prototype – nS-ESD-GW
Draft 9	24/04/13	Link layer security prototype
Final	25/04/13	Contributions to Chapter 3, 4, 5, 6. Final Issue

Executive Summary

The purpose of this document is to present the plan and methodologies driving the validation and verification activities for the nSHIELD prototypes. The document is structured as follows:

- Chapter 1 – provides a brief introduction on V&V methodologies and activities
- Chapter 2 – presents the SHIELD taxonomy
- Chapter 3 – is related to the Validation and Verification of Node layer prototypes
- Chapter 4 – is related to Validation and Verification of Network layer prototypes
- Chapter 5 – is related to Validation and Verification of Middleware and Overlay layer prototypes
- Chapter 6 – draws the conclusions

Contents

Figures

Tables

Glossary

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

1 Introduction

The final objective of the SHIELD project is to develop innovative SPD functionalities and to build a framework that is able to dynamically compose them. This objective is reached in four steps. Up to now, two steps have been almost done:

- WP2 has defined the SHIELD system and scenario requirements that drive the design
- WP3-4-5 have designed and developed the technological enablers and the enriched components that are the building blocks of the SHIELD framework

At this point, it is necessary to put these components together and to demonstrate, in a significant environment, the effectiveness of the results. So two further steps are needed:

- WP6 is in charge to verify and validate the behaviour of the **individual SHIELD prototypes** and then to integrate them together in a **common** (still generic) **platform**, that is validated and verified as well.
- WP7 is finally in charge to tailor and refine the generic platform for the specific needs of the application scenarios (**demonstrators**) and to validate the final behaviour.

These two steps are depicted in Figure 1-1.

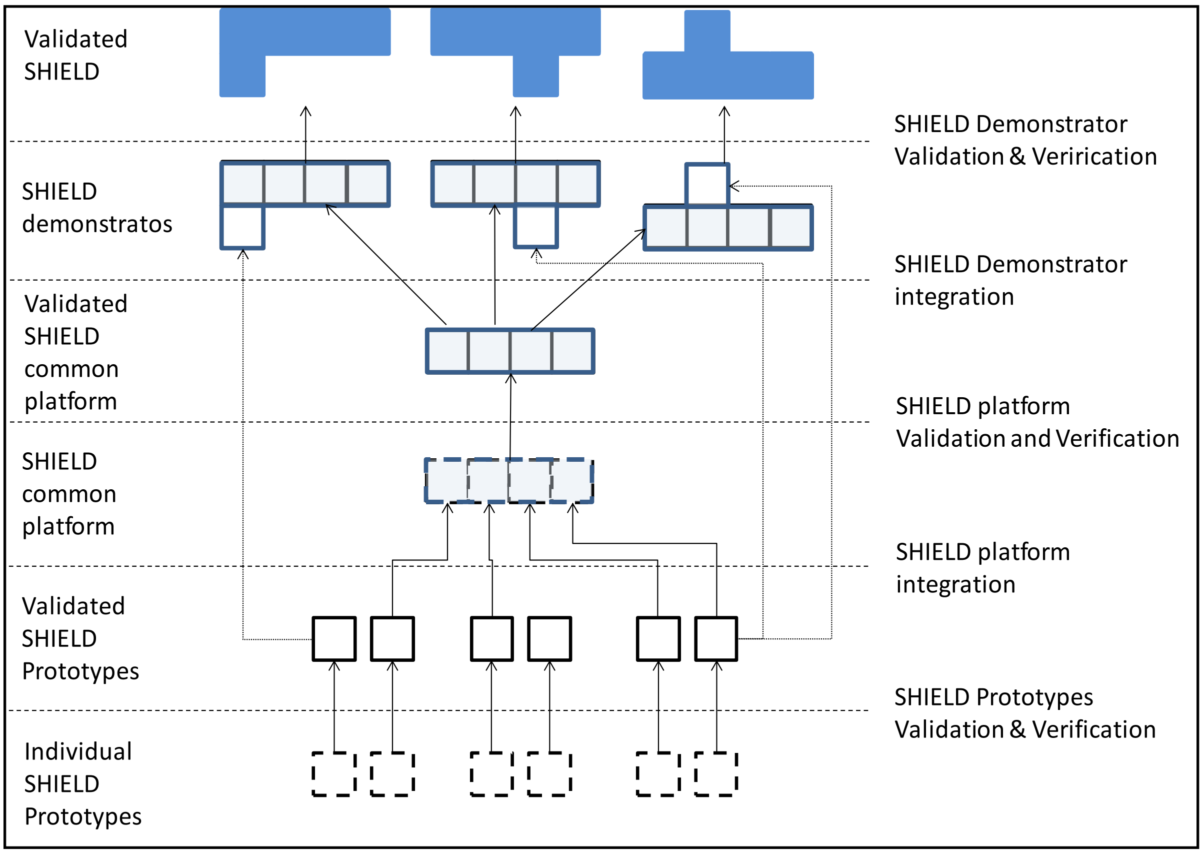
Validated SHIELD
SHIELD Demonstrator Validation & Verirication
SHIELD demonstratos
SHIELD Demonstrator integration
Validated SHIELD common platform
SHIELD platform Validation and Verification
SHIELD common platform
SHIELD platform integration
Validated SHIELD Prototypes
SHIELD Prototypes Validation & Verification
Individual SHIELD Prototypes

Figure 1-1: nSHIELD Integration, Validation and Verification approach

This deliverable addresses the first part of step three (corresponding to the lowest phase of Figure 1-1): the validation and verification of the individual prototypes, some of which will be selected for integration in the common platform or in the final demonstrators.

In particular, as well described in the corresponding technical deliverables, the complete list of prototypes developed in the framework of the nSHIELD project is the following:

Table 1-1: nSHIELD prototypes

ID	Prototype name	Owner
0	Elliptic Curve Cryptography	UNIGE
1	Lightweight Cyphering	TUC
2	Key Exchange Protocol	TUC
3	Hypervisor	SICS
4	Secure Boot	T2D
5	Secure Power (&) Communication cape	AT/TELC/TUC
6	Smart Card	TUC
7	Facial Recognition	ETH
8	GPU Hase	TUC
9	Smart Transmission	SES/UNIGE
10	Anonymity	TUC
11	Automatic Access Control	TUC
12	DDoS Attack Mitigation	ATHENA
13	Recognizing DoS	ATHENA
14	Dependable Distributed Computation Framework	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	Multimetrics	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
37	ETH SecuBoard	ETH

These prototypes are highly heterogeneous and in this deliverable it has been decided to split them depend on the layer, from node to overlay layer:

- Node Layer prototypes
  1) Audio Surveillance System (Prototype 34)
  2) Secure Boot (Prototype 04)
  3) SICS Hypervisor (Prototype 03)
  4) BeagleBoard-Xm prototype for SICS Hypervisor (Prototype 35)
  5) Smart-Card based services (Prototype 06)
  6) Secure Power (&) Communication cape (Prototype 05)
  7) Gateway nS-ESD-GW (Prototype 21)
  8) Automatic Access Control (Prototype 11)
  9) Face recognition (Prototype 07)

- Network Layer prototypes
  1) Reputation-Based Secure Routing prototype (Prototype 16)
  2) SPD-driven Smart Transmission Layer prototype (Prototype 09)
  3) Link Layer Security prototype (Prototype 23)
  4) DoS attack Defence (Prototype 12)
  5) Network Layer Security prototype (Prototype 24)
  6) Anonymity & Location Privacy service (Prototype 10)

- Middleware and Overlay layers prototypes
  1) SHIELD Semantic Model (Prototype 26)
  2) SHIELD Secure Discovery (Prototype 32)
  3) SHIELD Security Agent (Prototype 33)
  4) SHIELD Control Algorithms (Prototype 20)
  5) SHIELD Intrusion Detection System (Prototype 22)
  6) SHIELD Policy-based Access Control (PBAC) & Policy Based Management (PBM) (Prototype 19)
  7) SHIELD Middleware Protection Profile (Prototype 31)
  8) Adaptation of Legacy Systems (Prototype 29)

As it has been indicated previously, all these prototypes are highly heterogeneous. Some of them are software components, some are hardware components and some others are algorithms or models. For this reason, a common methodology for validation and verification activities is not provided, giving the choice of the most suitable mean of verification to the experts of the different layers.

However the macro areas of validation and verification should be chosen among the following standard procedures:

[A] **Analysis**: the verification that the requirement is covered is performed by means of a dedicated analysis

[D] **Design**: the verification that the requirement is achieved by means of a specific design choice

[I] **Inspection**: the verification that the requirement is covered is obtained by means of a visual inspection of the element

[T] **Test**: the verification that the requirement is satisfied is done through dedicated tests, which are well described and documented.

[R] **Review**: the verification that the requirement is covered is achieved by the review of project's documentation

Last, but not least, a distinction should be done between validation and verification activities. Commonly, verification is done with respect to a local or a functional property or requirement, while validation is done with respect to a wider end-to-end behaviour.

In the SHIELD system, the end-to-end behaviour is SPD behaviour, so the adopted rationale could be:

- To verify all the requirements that can be classified as functional and are necessary to make the prototype "work"
- Then, to validate the remaining SPD requirements and consequently the SPD behaviour of the prototype.

Since this distinction is not available in the requirements document, it will be done case by case.

Just to provide an example, let's consider as prototype, a network module for communication encryption; the validation and verification of the SPD functionality "secure communication exchange" could be treated as follows:

- A "ping" command is used to verify that the communication is enabled and messages could be delivered (no matter which is the content of the message)
- A brute force attack is done on an exchanged message to validate the security of the communication

Obviously one prototype could cover more than one SPD functionality, so there could be several validation procedures (one per SPD functionality) with the associated test cases. Verification procedures are most likely common to several validation procedures.

In the prosecution of the document the three SHIELD layers, with the associated prototypes (only the prototypes that were actually realised, not those which are in the study phase) are analysed with respect to V&V results.

1.1 Security Evaluation methodology supplementing Validation and Verification of secure technologies

The requirements developed in the work package WP2 in the nSHIELD project contain several security-relevant requirements, where validation and verification against these requirements would require evaluation of the security in the components.

1.1.1 Security evaluation methodology - MEFORMA

MEFORMA is a security evaluation methodology developed by SEARCH-LAB. The methodology was created to provide a framework for commercial evaluation projects, and it has been used (and further refined) in more than 50 evaluation projects over twelve years. The methodology was designed for manual security evaluation of (software and/or hardware) products.

SEARCH-LAB carries out security evaluation of some of the security-relevant components according to the MEFORMA methodology for assisting the validation and verification of those components against the requirements that have been identified.

1.1.2 Components of MEFORMA

1.1.2.1 Evaluation milestones

A MEFORMA evaluation consists of three to four phases.

- Preparation Phase: The test environment is established, and threat modelling is performed on the TOE to specify what exactly should be performed during the Evaluation Phase.
- Evaluation Phase: The test cases defined in the Preparation Phase are executed, confirming whether the originally-identified threats are viable or not. New findings are reported to the client regularly during the evaluation.
- Documentation Phase: The evaluators collect the findings of the Evaluation Phase, list all threats collected during the Evaluation Phase, and perform a risk analysis. Recommendations are given to deal with each threat.
- Review Phase: During this optional phase, a fixed version of the TOE can be re-evaluated to determine whether the threats identified during evaluation had been adequately addressed.

1.1.2.2 Deliverables

- Evaluation Plan: Contains the definition of the scope, the identified security objectives in the TOE, the threat model of the TOE, and the list of test cases to be executed during the Evaluation Phase.
- Weekly Status Reports: Sent to the client every week, these reports contain the current progress with the evaluation, schedule information, and any relevant findings.
- Evaluation Report: Delivered at the end of the Evaluation Phase, it contains all results of the evaluation along with a list of found threats, recommendations, and a risk analysis.
- Review Report: Delivered at the end of the (optional) Review Phase, it contains the results of the re-evaluation along with a revised list of found threats, recommendations, and a residual risk analysis.

1.1.2.3 MEFORMA evaluation process

1.1.2.3.1 Scope definition

Before performing a MEFORMA evaluation, the TOE (Target of Evaluation) must be identified, and the scope of the evaluation must be specified. The definition of the evaluation's scope is a co-operative effort between the Evaluator (e.g. SEARCH-LAB) and the client (prototype owner).

Basically there are three main aspects of planning an audit: scope, depth of analysis and the audit risk. If we limit the scope of the evaluation, important issues may not be addressed even if we increase the depth of the analysis. Contrarily, if we want to keep the scope as wide as possible and evaluate the whole system, limits in the available resources will imply limitations in the depth of the analysis. In both cases it is important to be aware of those remaining risk factors that the investigations would not cover.

In the current case of evaluating nSHIELD prototypes, efforts are allocated based on the experiences of previous audit projects and based on the time estimations of the proposed tasks. If the proposal is accepted in a review by the prototype owner, the Evaluator takes on to execute the specific tasks as long as the scope of the evaluation remains the same (minor problems are included).

1.1.2.3.2 Identification of security objectives

As a first step, the security objectives towards the TOE are specified. These are collected by first identifying the important assets within the system that need to be protected, and then determining which particular aspect of security from the industry-standard CIA triad (Confidentiality, Integrity, and Availability) must be applied to them. The assets are further grouped into categories appropriate to the specific evaluation (e.g. a software evaluation would likely have 'software assets', 'data assets', and 'other assets' categories).

If the client provides their own list of security requirements, those are used to refine these further, prioritizing the security of the assets the client finds to be the most important. In the case of nSHIELD prototypes, the relevant nSHIELD requirements identified form the basis of the objectives.

1.1.2.3.3 Threat modelling

Based on the security objectives, the evaluators perform threat modelling. This can be done following two different approaches.

- Attack tree modelling consists of drawing conceptual diagrams of perceived threats to a system. This process is performed by identifying an attacker goal, and then modelling the various ways these goals can be achieved. E.g. an attacker may have a goal of obtaining a web application's administrator password which is also stored on the user's computer. He could do this by deploying malware on the user's computer, eavesdropping communication between the user's computer and the web server, or even by exploiting vulnerability in the web application to obtain the password.
- Misuse cases are similar to the use case UML formalism, but instead of describing ways to use system functionality, they present ways on how to misuse it. These diagrams are useful for eliciting security requirements in the early stages of the software development lifecycle, and therefore MEFORMA uses misuse cases for design review projects. Misuse cases are derived from the normal use cases of the TOE – e.g. for a normal 'shutdown' use case there may be several misuse cases defined where the system shuts down.

Optionally, attacker profiling may also be performed in this stage. This identifies several different types of attackers that may have different goals when it comes to attacking the security of the TOE, and may also have different resources and expertise at their disposal. Example profiles are insider, exploiter, misuser, and thief.

1.1.2.3.4 Test case specification

During the threat modelling, many potential threats will be identified. Some of these threats may be considered out of scope for the evaluation due to being unfeasible (such as the vendor's secret key being leaked) or trivial (e.g. a particular aspect of the system is insecure by design). However, most of the threats will require investigation to confirm their feasibility.

To that end, the evaluators group relevant threats together, and specify test cases – evaluation of specific aspects of the TOE – in an effort to determine whether the relevant threats identified during the threat modelling process are feasible.

1.1.2.3.5 Evaluation

The test cases defined during the test case specification step are executed. Actual evaluation of a test case can consist of black-box / white-box / grey-box testing, or source code review. Throughout the evaluation, we use several symbols to denote the results of individual tests within a test case. These symbols are as follows:

✓: Normal operation. The outcome of the test indicates that the implementation is correct.

💣: Problem. The outcome of the test has clearly identified a security problem.

😐: Potential / possible problem. The outcome of the test does not clearly indicate a security problem, but may lead to unexpected or abnormal operation. This symbol is also used if a security issue is suspected, but could not be verified (e.g. a necessary interface wasn't available during the test).

1.1.2.3.6 Threat documentation

We compile a list of threats based on the results of the evaluation. These may be threats that were identified during the threat modelling step, or completely new threats altogether.

For each threat, we perform a preparedness evaluation: the resources and expertise required by the attacker to realize it.

1.1.2.3.7 Risk analysis

We estimate the risk each discovered threat poses to the system. This is done by specifying the severity (damage that can be potentially done by realizing the threat) and likelihood (the difficulty of realizing the threat) of each threat. The risk is the product of these two values using the standard likelihood × severity risk calculation:

Table 1-2: Likelihood x severity risk calculation

Likelihood / Severity	Low	Medium	High
Low	Very Low	Low	Medium
Medium	Low	High	Very High
High	Medium	Very High	Catastrophic
Very High	High	Catastrophic	Catastrophic

The risk value of each threat can take the following levels:

- *Very Low* (VL): The threat has a very minor – but not negligible – effect on the security of the asset.
- *Low* (L): The threat has a minor effect on the security of the asset.
- *Medium* (M): The threat has a noticeable effect on the security of the asset.
- *High* (H): The threat significantly endangers the asset.
- *Very high* (VH): The threat significantly endangers the asset or the system as a whole.
- *Catastrophic* (C): The threat presents a critical risk to the system as a whole; if not mitigated, its effects could put the entire business process at risk.

1.1.2.3.8 Recommendations

For each threat, we specify recommendations that could be used to reduce its risk or eliminate the threat entirely. In each case, we aim to reduce either the likelihood or severity of the threat to zero.

After finishing the recommendations, the Evaluation Report is sent to the client, who can then make the appropriate steps to address the issues.

1.1.2.3.9 Review

Following the evaluation, a several weeks long review phase may be requested by the client. If so, we receive a newer version of the TOE, and re-run all of the test cases on the new TOE to verify if the threats have been appropriately dealt with.

After this re-evaluation, we create a residual risk analysis, showing the threats that still remain in the system after the review. If new threats are discovered during the review, those are listed as well.

The final review will describe the validation and verification results against relevant nSHIELD requirements.

[reference accepted best practices/standards]: Cert C++ source code analysis, CISA auditing methodology

2 Terms and definitions

Audit	Involves recognizing, recording, storing, and analysing information related to SPD relevant activities. The resulting audit records can be examined to determine which SPD relevant activities took place.
Authorised User	A user who possesses the rights and/or privileges necessary to perform an operation
Class and Family	The CC has organized the components into hierarchical structures: Classes consisting of Families consisting of components. This organization into a hierarchy of class - family - component - element is provided to assist consumers, developers and evaluators in locating specific components
Common Criteria	The Common Criteria for Information Technology Security Evaluation (abbreviated as Common Criteria or CC) is international standard (ISO/IEC 15408) for computer security certification. It is currently in version 3.1. Common Criteria is a framework in which computer system users can specify their security functional and assurance requirements, vendors can then implement and/or make claims about the security attributes of their products, and testing laboratories can evaluate the products to determine if they actually meet the claims. In other words, Common Criteria provides assurance that the process of specification, implementation and evaluation of a computer security product has been conducted in a rigorous and standard manner.
Composability	Is the possibility to compose different (possibly heterogeneous) SPD functionalities (also referred to as SPD components) aiming at achieving in the considered system of Embedded System Devices a target SPD level which satisfies the requirements of the considered scenario.
Cryptographic Algorithms	Algorithms to hiding the information, to provide security and information protection against different forms of attacks
Discovery	Provide to the pSHIELD Middleware Adapter the information, raw data, description of available hardware resources and services in order to allow the system composability
Life-Cycle support elements	It is the set of elements that support the aspect of establishing discipline and control in the system refinement processes during its development and maintenance. In the system life-cycle it is distinguished whether it is under the responsibility of the developer or the user rather than whether it is located in the development or user environment. The point of transition is the moment where the system is handed over to the user.
Overlay Layer	The “embedded intelligence” that drives the composition of the pSHIELD components in order to meet the desired level of SPD. This is a software layer as well.
Personal Area Network	Computer used for communication among computer devices, including telephones and personal digital assistants, in proximity to an individual's body.
PAN Coordinator	The ZigBee device which is responsible for starting the formation of a ZigBee network. The ZigBee PAN coordinator chooses the PAN ID. There is only one ZigBee PAN Coordinator in any ZigBee network; its ZigBee address is always 0.
TinyOS	This operating system (OS) is a free and open source operating system and platform that is designed for WSNs.
User session	A period of interaction between users and SPD functional components.
Wireless Sensor Network	It consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to a main location.

3 Node Prototypes validation and verification

3.1 SHIELD Node requirements

An initial node requirements assessment has been conducted in nSHIELD deliverable D2.2: Preliminary System Requirements and Specifications. In this section, the requirements are recited and their approach is updated through the -up to now- available input from the technical developments of nSHIELD Node Layer design (WP3) and related work.

In the following table the requirements that are addressed by the prototypes developed for the SHIELD node are reported, with the indication of the mean of verification (A, D, I, T) and the applicability scope (validation or verification). Only requirements covered by test are extensively investigated.

Table 3-1: Requirements relevant against validation and verification of Node layer prototypes

REQ_ND01 - Code execution An nSHIELD node should verify that only authorized code (booting, kernel and application) runs on the system. Carried out for SICS Hypervisor through a Security Evaluation with MEFORMA methodology [A, D, T]
REQ_ND02 - Data Freshness An nSHIELD node should include data freshness checks to avoid replay attacks. This requirement depends on network protocol. Some cryptographic primitives/low level mechanisms are supported by several chips (e.g. MRF chip). [A, D]
REQ_ND03 - Digital Signatures An nSHIELD node should be able to verify digital signatures even in cases where a trusted third party is not available. This is important to allow flexible secure operation in highly mobile scenarios. Different signature verifications can be done using JavaCard applets (like implementation of ECDSA java card applet on the smart card) [A, D]
REQ_ND05 - TPM Low Power mode An nSHIELD node's TPM shall be able to enter into a low power state without compromising its security. Resource-constraints may lead to a device running out of power. In that scenario the system must ensure it won't compromise the network's security or the security of stored data on the node itself. In this case, this requirement depends on the TPM module selected. [A, D]
REQ_ND09 - Virtualization An nSHIELD power node should employ virtualization techniques to allow concurrent virtual nodes to run independently onto the system. Carried out for SICS Hypervisor through a Security Evaluation with MEFORMA methodology [A, D, T]
REQ_ND11 - Node – Physical/tamper resilience An nSHIELD node shall be designed to not compromise the privacy of the contained information in the case of a malicious user gaining physical possession of the device. The device shall be resilient to tampering, micro-probing and reverse-engineering. Nodes often deployed in hostile and/or not monitored environments, thus owner's physical control over the device is not always an option (i.e. malicious users might have access to the actual node without being detected). This feature is a requirement that needs to be considered during design stage and it will depend on the final environment where the node will be work. [A, D, T]
REQ_ND12 - Low computation authentication schemes An nSHIELD node should include an optimized hardware implementation for an ECC-based public-key authentication algorithm. ECC will be one of the main lightweight cryptographic primitives utilized by nSHIELD nodes, both for encryption and digital signatures. Some high performance TPM chips include features such acceleration. Other way to cover this requirement is by means a software solution like using JavaCard/JCOP smart cards (built-in functionality in JavaCard). [A, D]

REQ_ND17 - Privacy in different trust domains An nSHIELD node shall feature the necessary mechanisms for security token exchange to enable the issuance and dissemination of credentials within different trust domains.
REQ_ND23 - Hardware/Software co-design An nSHIELD node should incorporate hardware-software co-design techniques to substantially increase application (e.g. public-key cryptography) performance with minimal device surface area and cost increase. Some chips include minimal hardware acceleration of cryptographic primitives, e.g. the MRF chip, as dictated by the IEEE 802.15.4-2003 Standard. [A, D]
REQ_ND27 - Accommodations for future energy sources An nSHIELD node should have provisions for future alternative power sources including super-capacitors and wireless power schemes. Utilization of advances in the field of energy sources which would help alleviate some of the resource-constraints or offer fail-safe alternatives. 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. [A, D]
REQ_ND28 - Power management In the nSHIELD system the hardware shall provide frequency dividers and variable supply voltage, in order to meet the most appropriate energy/performance trade-off. This requirement manages any system power supply risk, which might affect to the node behaviour. It is useful at this level a continuous power supply source, without any cut in time neither in the power, voltage or current levels, to correctly bias the devices. 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. [A, D]
REQ_ND29 - Power management interface In the nSHIELD system the hardware shall provide an appropriate interface, such as I/O ports or memory mapped registers, in order to allow the software to drive the frequency dividers and the supply multiplexers. The power management policies should be realizable as software routine, to allow flexibility and upgrading. [A, D, T]
REQ_SH07 Secure node deployment A new node shall be identified as a trusted node prior to be accepted in the system.
REQ_SH10 Secure execution A system node should provide an isolated protected execution environment that can offer secure execution for the most security sensitive computing tasks. Carried out for SICS Hypervisor through a Security Evaluation with MEFORMA methodology [A, T]
REQ_SH11 Secure Boot All nodes should provide a secure boot process. Carried out for Secure Boot and SICS Hypervisor through a Security Evaluation with MEFORMA methodology. [A, D, T]

3.2 SHIELD Node prototypes overview

In this section a brief overview of the node layer prototypes is given, that shows, at high level, how these requirements have been addressed in the design and development activities.

3.2.1 Audio Surveillance System (Prototype 34)

ISD has been designing a novel audio based surveillance infrastructure that aims to overcome the most important limitations of non-military grade systems currently utilized in acoustic based research by providing correlated data acquisition from a large number of overlapping sensors. It will be the only system able to deliver to the main memory of a single host synchronized uncompressed audio streams from up to 768 microphones with zero CPU load. Its hierarchical structure is fully extensible and able to support any number of microphones. The targeted implementations will support from 8 up to 768 sensors in multiples of 8 units.

3.2.2 Secure Boot (Prototype 04)

The firmware should be an integral part of the CPU core to prevent tampering. The secure boot is the first software executed after a reset. Prior to transfer control to next layer of software, the contents of the payload are verified to assure integrity of the node.

The payload is divided into individually signed sections. The first section address memory configuration, the following sections contain header carrying information where to locate the image in memory. The design supports any operating system to be loaded, as well as hypervisor. The prototype implementation is not integrated into the core CPU, but designed to operate in a constrained environment.

The modular design has been applied to BeagleBone and BeagleBoard as well as AMD SC2200 X86 SOC.

3.2.3 SICS Hypervisor (Prototype 03)

As part of the nSHIELD prototyping efforts, SICS has developed a hypervisor that aims to enhance security in embedded systems by guaranteeing isolation and secure interaction between co-existing open software components and closed trusted security critical components.

The isolation properties have been achieved by implementing the hypervisor to be the only software to run at the most privileged level of the CPU, giving it full access to the system. All guest execution environments have also been modified to run exclusively in the CPU's user mode, which required paravirtualization of the guest OS. This requires some effort, however the advantages are many. Besides having the possibility to run multiple execution environments on the hardware with secure access policies, the trusted computing base also becomes imminently smaller by not having vast amounts of OS kernel code running in the privileged mode. The performance overhead of a well-designed hypervisor is low enough that the performance trade away is well worth the increased security of the system.

3.2.4 BeagleBoard-Xm prototype for SICS Hypervisor (Prototype 35)

For the prototype, SICS is using the BeagleBoard-XM rev C running on the OMAP3530 SOC which includes an ARM Cortex A8 single core CPU. The Linux kernel 2.6.34.3 have been paravirtualized to run on top of the hypervisor in user privileged mode, co-existing with a security application that offers security services running in a different execution environment isolated from the Linux OS. Communication between the trusted application and the Linux OS can only occur through a well-defined interface that the hypervisor provides. The hypervisor also maintains isolation between the Linux kernel and its user processes, just like in the normal case of an unmodified Linux kernel.

Currently, only the most important peripherals have been mapped to the Linux kernel such as the interrupt controller, timer and UART. No other device mappings are possible but will be supported in the near future. A solution for virtualizing the DMA has also been developed but is not supported on the current BeagleBoard version of the hypervisor. There is no graphical support, and all communication with the OS is done through the UART. A bash shell console is available, in where you can navigate, explore and execute programs in the init ramdisk filesystem.

3.2.5 Smart-Card based services (Prototype 06)

To build trust among different type of nodes on the nSHIELD architecture we can exploit the benefits of smart cards and the cryptographic schemes they implement. Considering, the nSHIELD architecture where decentralized components are interacting not only with each other but also with centralized ones, depending on the type of the device and the employed scenario; there is a need for integrating security and interoperability. In this context, we developed a module for build building secure communication channels among different devices that supports the following security services:

- Allow secure key management required for establishing secure channels between different nodes.
- 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.

3.2.6 Secure Power (&) Communication cape (Prototype 05)

This nSHIELD node prototype is composed of different subsystems that are directly related to different partners' expertise. In deliverables D3.2 and D3.3 the BeagleBone/BeagleBoard devices have been presented as the selected by several partners as the base platform to support the implementation of their nSHIELD solution. This prototype has been designed as a BeagleBone cape. This cape could be used by any partner working with BeagleBone as reference board.

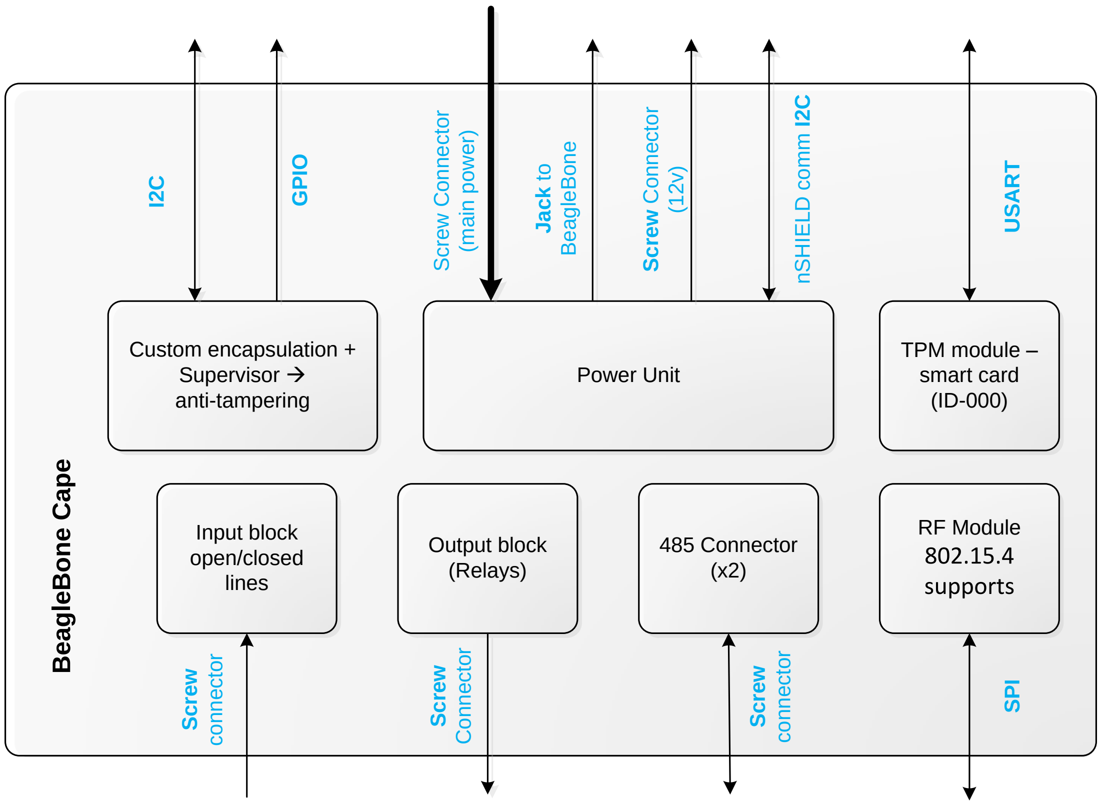

Figure 3-1: BeagleBone Cape prototype

Designed features:

1. **Custom encapsulation + Supervisor and anti-tampering** The physical protection will depend on the needs of the board and it can be vary from a simple box that include a tamper detection switch to a custom encapsulated, more costly mesh, that offer higher security levels (i.e. when there whole enclosure integrity needs to be protected). However in the second case a custom redesign should be performed of the board to be protected (board dimensions, number, location and size of external connectors…) in order to be able to apply a convenient protection mesh at reduced cost. If the TPM module provides the required security level and only the stored secure keys need to be protected it is not necessary to add a secure mesh encapsulation to the whole target board. In this prototype the first option has been implemented.
2. **Power unit** for the BeagleBone board and third-party boards
3. **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).
4. **RF Module** that supports the 802.15.4, based on the MRF24J40 that provides a wireless communication link.
5. **Other features**:
    a. Additional RS-485/RS-232 external interfaces (driver + connector) will be available in the cape.
    b. RTC signal will be provided.
    c. Two relays
    d. Several digital inputs

The table below summarizes the list of node requirements addressed by this prototype.

Table 3-2: Prototype 05 - Node Requirements addressed

Prototype ND06: BeagleBone Cape	Mean of Verification
REQ_ND02 - Data Freshness	A, D
REQ_ND03 - Digital Signatures	A, D
REQ_ND05 - TPM Low Power mode	A, D
REQ_ND11 - Physical/tamper resilience	A, D, T
REQ_ND12 - ECC Authentication	A, D
REQ_ND23 - Hardware/Software co-design	A, D
REQ_ND28 - Power management	A, D
REQ_ND29 - Power management interface	A, D, T

3.2.7 Gateway nS-ESD-GW (Prototype 21)

SESM has been developing the nS-ESD-GW (nSHIELD Embedded System Device Gateway) that is used to interconnect the nSHIELD Middleware and the node layers. It provides enhanced capabilities in terms of security and dependability to the cluster where it will be integrated in. The hardware computing platform of the Gateway consists of a hybrid architecture composed by dual-core Cortex ARM A9 and FPGA. The choice of using this kind of platform confers to this node high performance thanks to the coexistence of hardware-software developing techniques for the same device. As reported by the Reference System Architecture Design D2.4, the nS-ESD-GW includes different modules whose synergy will foster the integration of legacy embedded systems into the nSHIELD architecture as well as they will confer specific capabilities in terms of composability and SPD. The Figure 3-2 shows a logical view of the internal architecture of the Gateway.

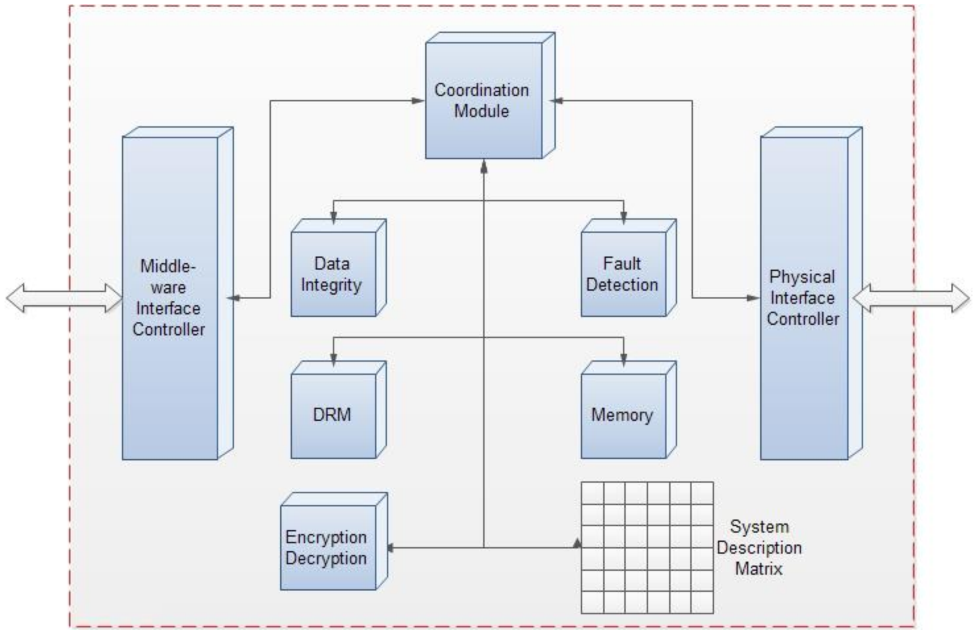
Coordination Module
Middle-ware Interface Controller
Data Integrity
Fault Detection
Physical Interface Controller
DRM
Memory
Encryption Decryption
System Description Matrix

Figure 3-2: nS-ESD-GW Logical View

The development of the gateway has been done performing a modular approach. This enables the tight partitioning and isolation between internal components involved to implement security, communication and monitoring functions.

Design and development activities:

- blocks referred as Middleware and Physical Controllers are responsible for communication, data conversion and proxy services. In particular, the physical interface controller, used to connect legacy nodes, provides a subset of common ready-to-use interfaces as: ETH, SPI and I2C.
- the Coordination Module represents resources, both hardware and software, executing balancing and safety algorithms as well as nSHIELD specific operations.
- the component identified as DRM (Dynamic Reconfiguration Module) is in charge of managing the reconfiguration of the processing block and software applications based on the nSHIELD request. The purpose of this functional block is to increase security and dependability aspects of the node allowing the device to switch between different operational modes.
- the Fault Detection module encompasses the nSHIELD algorithms. It consists of several software modules and some processing blocks.
- the accuracy and consistency of data exchanged among the nSHIELD middleware and legacy nodes is assured by the Data Integrity module. Moreover, the confidentiality and security of long-term data storage is ensured by the Encrypt/Decrypt controller. Due to the fact that cryptographic algorithms are processing consuming, this block is implemented as a FPGA-based module, providing a software-like flexibility with hardware-like performances.

3.2.8 Automatic Access Control (Prototype 11)

For automatic access control functionality, TUC implement the Gossamer protocol. Gossamer utilizes pseudorandom numbers and simple bit operations and is an ultra-lightweight protocol for mutual authentication. It furthers provide data confidentiality, tag anonymity, 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. When the protocol fails to recognize a legitimate user, it may lead to the conclusion that the system is under attack. The two entities communicate through the network with sockets. The implementation is in C++ and is applied on Memsic IRIS and BeagleBone devices.

3.2.9 Face recognition prototype (Prototype 07)

The face recognition prototype is an all-in-one solution created to practically demonstrate the functionalities and potentialities of the face recognition system for people identification. It represents a proof of concept of the technologies adopted for the face recognition and it will be used to develop the final prototype. This prototype belong to the “Face and voice recognition” application scenario and has been developed during the first part of nSHIELD project. During the second part of the project it will finalized developing the embedded camera for face recognition that can be used in a real environment.

From a software point of view, the approach identified in the assessment phase has been implemented and satisfies the technical requirements for face recognition. The prototypes, both the Windows and the Linux versions, are based on the Eigenface method. This method is based on the idea of extracting the basic features of the face: the objective is to reduce the problem to a lower dimension maintaining, at the same time, the level of dependability required for this application context. The core of this solution is the extraction of the principal components of the faces distribution, which is performed using the Principal Component Analysis (PCA) method. This method is also known in the pattern recognition context as Karhunen-Loève (KL) transform. The principal components of the faces are eigenvectors and can be computed from the covariance matrix of the face pictures set (faces to recognize). Every single eigenvector represents the feature set of the differences among the face picture set. The graphical representations of the eigenvectors are also similar to real faces and, for this reason, they are called eigenfaces. The PCA method is autonomous and therefore is particularly suggested for unsupervised and automatic face recognition systems. This software solution has been developed in C++ and has been compiled for Windows, (all-in-one demonstrator) and for Linux-ARM (final embedded camera prototype).

3.3 SHIELD Node prototypes verification and validation

In this section a description of the verification and validation activities is reported, with particular attention to those prototypes whose associated requirements are marked with a T (i.e. a test is needed or verifies that the requirement is met).

3.3.1 Audio Surveillance System (Prototype 34)

1 Verification Procedure

i. Verification of the functionality of an individual system port [I]. A microphone is attached to one system port and is exposed to a known audio signal. Audio capturing is activated for several seconds. An audio file is generated by the system and its contents are played back by an operator who verifies that the captured data match the audio signal to which the microphone was exposed. The experiment is repeated for all system ports and for all sampling frequencies.

ii. Verification of synchronous audio capture [I]. Several microphones are mounted on system ports and placed in a linear topology. Capturing is activated for several seconds and an audio signal is produced at a distant location. The captured data are analysed by an operator verifying that the time offset of each signal peak in the captured data is analogous to the distance between the respective microphone and the audio source.

2 Validation Procedure

i. REQ_ND01 Code execution [D]. The firmware is not upgradeable in the field avoiding by design the execution of malicious code.

ii. REQ_ND02 Data Freshness [D]. A monotonically increasing serial number is attached to each audio sample captured ensuring the ability to perform data freshness checks.

iii. REQ_ND07 Situational-aware SPD [D]. The system reports the availability of data acquired from each sensor and it’s able to operate even when most of its sensors have been destroyed by an attack due to its inherent redundancy.

iv. REQ_ND11 Node - Physical/tamper resilience [D]. If required by an application scenario, some of the sensors can be used in order to detect any attempt for unauthorized access to the system itself

3.3.2 Secure Boot (Prototype 04)

Verification and Validation of the Secure Boot prototype was carried out through the Security Evaluation of this technology using the MEFORMA methodology (see chapter 0) by SEARCH-LAB. The actual test cases were described in the Evaluation Plan, provided based on the relevant nSHIELD requirements. The Evaluation Plan and Evaluation Report were submitted to the prototype owner T2DATA.

Test cases identified:

Table 3-3: Prototype 04 - Secure Boot identified test cases

Test Case	Means of verification	nSHIELD requirements	Description
Source code analysis of the Secure Boot	A, D, T	**REQ_SH07 Secure node deployment** **REQ_SH11 Secure Boot**	The Secure Boot is responsible to load and verify the Kernel and the Hypervisor before it would be started. In this test case we checked whether the main function and the high level logic of the Secure Boot were implemented correctly. We also checked whether the implementation of the Secure Boot was free from typical security flaws such as buffer overflows, integer overflows, logic flaws and memory handling issues.
Security of the signature verification process	A, D	**REQ_ND01 Code execution**	The signature verification process is the most important part of the Secure Boot, which ensures that only trusted Kernel and Hypervisor will be able to execute on the device. The aim of this test case was to check whether the image validation was implemented correctly and whether it was free from typical security flaws such as buffer overflow, integer overflow, logic flaws, memory handling issues and cryptographic issues. We paid attention to the following specific issues: • Implementation of the cryptographic algorithms such as the possibility of the Bleichenbacher attack. • Protection of the used public key.
Source code analysis of other libraries	A, D	**REQ_SH11 Secure Boot** **REQ_SH10 Secure execution** **REQ_ND01 Code execution**	To access hardware devices and load the boot image from the MMC, the Secure Boot should use additional libraries, which may contain security flaws. In this test case we checked whether these libraries (MMC, VFAT, etc.) were free from typical security flaws such as buffer overflows, integer overflows, logic flaws and memory handling issues.

The review document provided results for all tests validating and/or verifying if the identified requirements were met. The evaluation was carried out with the following goals:

- Validate that the design of the Secure Boot provides the necessary features to fulfil nSHIELD requirements
- Verify that the design is correct to implement the security implied by the nSHIELD requirements
- Verify that the implementation is correct with regard to the design, i.e. it is free of implementation errors regarding security

The evaluation milestones 1-3 (see chapter 2.1.2.1) were carried out in the process of the validation and verification of the Secure Boot, while milestone 4 (review phase) will be reached at the time of the integration of this prototype. In the evaluation and documentation phases the following findings were identified, which need correction in order to fulfil related nSHIELD requirements:

- It was possible to bypass the integrity protection mechanisms of the boot loader due to logic and design flaws. This could allow an attacker to start any kernel or hypervisor on the system.
- Several vulnerabilities in the boot loader could be potentially exploited, allowing an attacker to run arbitrary code.

The Evaluation Report contained detailed error description with annotated code parts related to the problem, to make it straightforward to correct the findings. There were recommendations given for each threat discovered during the evaluation. Of these, the most important recommendations are:

- Make sure that all data in the boot image is verified or validated before using this data.
- Use appropriate input validation and error handling in all functions dealing with external input.

3.3.3 SICS Hypervisor (Prototype 03)

Verification and Validation of the SICS Hypervisor prototype is carried out through the Security Evaluation of this technology using the MEFORMA methodology (see chapter 0) by SEARCH-LAB. The actual test cases were described in the Evaluation Plan, provided based on the relevant nSHIELD requirements. The Evaluation Plan and Evaluation Report were submitted to the prototype owner SICS. Test cases identified:

Table 3-4: Prototype 03 - Hypervisor identified test cases

Test Case	Means of verification	nSHIELD requirements	Description
Source code analysis of hypercalls	A, D, T	**REQ_ND17 Privacy in different trust domains**	The guest operating system can communicate with the Hypervisor via hypercalls. The aim of this test case was to check whether the hypercalls were implemented correctly and whether it was free from typical security flaws such as buffer overflow, integer overflow or logic flaws. Since hypercalls can be called only from the kernel, we supposed that an attacker could execute code from the kernel during this test case.
Protection of virtual guest modes	A, D	**REQ_ND09 Virtualization** **REQ_SH10 Secure execution**	Virtual guest modes are set by the Hypervisor based on the various hypercalls initiated by the Kernel. These guest modes provide the separation mechanism between the kernel, trusted applications, user applications and the interrupt handlers. In this test case we checked whether state of the virtual guest modes were protected well and whether the implementation was free from any logical flaw.
Protection of Trusted Application	A, D	**REQ_ND17 Privacy in different trust domains**	Trusted Application is running in a completely separated environment from the guest. Applications in the guest can communicate with the Trusted Application only via RPC hypercalls. The aim of this test case was to verify whether these RPC calls was not decrease the overall security of the system and whether it was possible to bypass the Hypervisor and access the Trusted Application in any other ways.
Protection of hardware devices	A, D	**REQ_SH10 Secure execution**	The Hypervisor should prevent mapping of hardware devices for the guest system. The aim of this test case was to check whether this restriction was implemented correctly and whether any way existed to bypass this protection.

The review document provided results for all tests validating and/or verifying if the identified requirements were met. The evaluation was carried out with the following tasks goals:

- Validate that the design of the Hypervisor provides the necessary features to fulfil requirements

- Verify that the design is correct to implement the security implied by the requirements
- Verify that the implementation is correct with regard to the design, i.e. it is free of implementation errors regarding security

The evaluation milestones 1-3 (see chapter 2.1.2.1) were carried out in the process of the validation and verification of the Hypervisor, while milestone 4 (review phase) will be reached at the time of the integration of this prototype. In the evaluation and documentation phases the following findings were identified, which need correction in order to fulfil related nSHIELD requirements:

- A vulnerability in one of the hypercall handlers within the hypervisor could allow an attacker to write data anywhere in memory, potentially resulting in arbitrary code execution.
- A design weakness in the example Trusted Application's signature verification function could allow an attacker to forge a valid signature relatively easily.

The Evaluation Report contained detailed error description with annotated code parts related to the problem, to make it straightforward to correct the findings. There were recommendations given for each threat discovered during the evaluation. Of these, the most important recommendations are:

- Use appropriate input validation and error handling in all functions dealing with external input.
- The signature verification function in the example Trusted Application should use the appropriate method for comparing binary data objects.

All the above identified SICS hypervisor prototype implementation vulnerabilities have been addressed and a new updated more secure release of the hypervisor is now available.

3.3.4 BeagleBoard-Xm prototype for SICS Hypervisor (Prototype 35)

The validation of this prototype was linked with the validation of the software environment executed on it, namely the validation of Secure Boot and SICS Hypervisor prototypes was carried out in the context of the BeagleBoard-Xm software environment. The hardware platform was not validated as such, but the protection of the relevant assets was taken into account. See chapters 4.2.2 and 4.2.3 for further details.

3.3.5 Smart-Card services module (Prototype 06)

The proposed module will be verified through an experimental specific test bed. Particularly, the module will verify the provided requirements as described in the following Table.

Table 3-5: Prototype 06 - Verification procedures

Relevant nSHIELD requirements	Means of Verification	Description
REQ_ND17 Privacy in different trust domains	T, D	Valid as well as invalid smart cards will be used in order to demonstrate solution's ability to exchange the security tokens and disseminating credentials correctly among different trust domains.
REQ_ND02 - Data Freshness	T, D	A node in the network that will use previous values to demonstrate the ability of the solution to avoid replay attacks.
REQ_ND11 - Physical/tamper resilience	D	Smart Cards are by design physical/tamper resilience.

3.3.6 Secure Power (&) Communication cape (Prototype 05)

The Smart Power unit has been design as a BeagleBone cape, and some additional functionality has been added in order to enrich the prototype.

Three verification procedures for the communication with the different modules included in the prototype have been developed for prototype 06:

Table 3-6: Prototype 05 - Verification procedures

Verification Procedures for powering and communication prototype	**Test Description**
Test Nr 3.3.6.1	Communication and power BeagleBone-SPC cape
Test Nr 3.3.6.2	Communication - SPC cape (TPM module)
Test Nr 3.3.6.3	Communication - SPC cape (Wireless module)

For the validation that the operation of Prototype 05 meets node requirements (T) of Table 3-2, the following validation procedure was developed.

Table 3-7: Prototype 05 - Validation procedure

Validation Procedure for powering and communication prototype	**Test Description**
Test Nr 3.3.6.4	Tamper detection and countermeasures actions

Table 3-8: Prototype 05 - Verification test #1

Verification Test Nr.: 3.3.6.1		**Written by:** AT	**Conducted by:**	**Date:**	**Test Category:** Node powering
Software and Hardware Configuration Details		• BeagleBone • Secure Power Communication cape (BeagleBone Cape) • PC running telnet program • Multimeter			
Test Name:		***Communication and power supply BeagleBone-Smart Power Unit***			
Purpose:		Verify that that the power module powers the BeagleBone and third party chips.			
Modules/Interfaces/Code Tested:		Node layer module Power manager and tamper module			
Step	**Action**		**Expected Result**	**Pass/Fail**	**Remarks**
1	Stack the Secure Power Communication cape to the BeagleBone		-		
2	Connect power cable between cape and main board.		-		
3	Provide external source power to the cape board and switch it on.		Beagle bone is powered, led status is ok and communication is working. BeagleBone power signal is stable at 5V.		
4	Communication between BeagleBone PC is functional		Control commands return valid responses.		
5					
6					
7					
8					

Table 3-9: Prototype 05 - Verification test #2

Verification Test Nr.: 3.3.6.2	**Written by:** AT	**Conducted by:**	**Date:**	**Test Category:** Node Security
Software and Hardware Configuration Details	• BeagleBone • Secure Power Communication cape (BeagleBone Cape) • PC running telnet program			
Test Name:	***Communication - Secure Power (&) Communication cape (TPM module)***			
Purpose:	Verify the communication and main functionality between the Smart Card (TPM module) with the BeagleBone board.			
Modules/Interfaces/Code Tested:	Node layer module TPM module			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Stack the Secure Power Communication cape to the BeagleBone	-		
2	Connect power cable between cape and main board.	-		
3	Provide external source power to the cape board and switch it on.	Beagle bone is powered, led status is ok and communication is working. BeagleBone power signal is stable at 5V.		
4	Communication between BeagleBone PC is functional	Control commands return valid responses.		
5	Communication between BeagleBone and Smart Card module is functional.	Control commands return valid responses. It will be use a test applet to verify the correct functionality. Main command subset to test: 1) Request JCOP version from card manager 2) Select test applet and provide applet PIN 3) Send command to test applet and receive expected response		

Table 3-10: Prototype 05 - Verification test #3

Verification Test Nr.: 3.3.6.3	**Written by:** AT	**Conducted by:**	**Date:**	**Test Category:** Node Communication
Software and Hardware Configuration Details	• 2 x BeagleBone • 2 x Secure Power Communication cape (BeagleBone Cape) • PC running telnet program			
Test Name:	***Communication -Smart Power Unit (Wireless module)***			
Purpose:	Verify the communication between the wireless module and the BeagleBone, and also the wireless communication between two BeagleBone boards with the Secure Power Communication cape (BeagleBone Cape).			
Modules/Interfaces/Code Tested:	Node layer module Wireless module			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Setting up two BeagleBones loading the correct kernel	-		
2	Initialize the interface of the wireless chip	Communication between the BeagleBone and Wireless chip is initialized		
3	Wireless Communication between the two BeagleBones (with Smart power unit Cape) running a simple ping-pong test between them			
4				
5				
6				
7				
8				

Table 3-11: Prototype 05 - Validation test #1

Validation Test Nr.: 3.3.6.4	**Written by:** AT	**Conducted by:**	**Date:**	**Test Category:** Node Security
Software and Hardware Configuration Details	• BeagleBone • Smart Power Unit (BeagleBone Cape) • External Li-on battery for Smart Power Unit • PC running telnet program • Multimeter			
Test Name:	***Tamper detection and countermeasures actions***			
Purpose:	Test Validation of nSHIELD Network requirements: • REQ_ND011 - Node – Physical/tamper resilience • REQ_ND029 - Power management interface			
Modules/Interfaces/Code Tested:	Node layer module Power manager and tamper module			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Stack the Smart Power unit to the BeagleBone	-		
2	Connect power cable between cape and main board.	-		
3	Provide external source power to the cape board and switch it on.	BeagleBone is powered, led status is ok and communication is working.		
4	Remove external source power and check that the external battery provides energy.	BeagleBone should not experience any anomalies in its functional state while running on battery power.		
5	Tamper event is triggered manually	Tamper event is detected by BeagleBone Supervisor removes sensitive information from its internal secure memory.		
6	Modify external voltage in the 0V-12V range	External measured voltage matches configured value		
7	Progressive lower external load to reach the 500mA limit	External current is provided until 500mA, the current is limited at that value and no damages are suffered by the cape board.		

3.3.7 Gateway nS-ESD-GW (Prototype 21)

The following table reports requirements covered by the nS-ESD-GW.

Table 3-12: Prototype 21 - nS-ESD-GW identified test cases

ID	nSHIELD Requirement	Mean of verification	Description
T.1	**REQ_ND02 Data Freshness**	A	The nS-ESD-GW encompasses two distinct Data Freshness methods. Stored data can be updated periodically or upon system requests. The update time interval is controlled by a configurable parameter. The system is endowed by a set of registers that contain information about the data freshness. [A] – A number representing the amount of time since data have been stored is saved into a specific register of the Gateway and it is available as output.
T.2	**REQ_ND03 Digital Signatures**	A, T	A mechanism of Digital Signature check is applied to communicate with legacy nodes. [A] – Once a legacy node is elected as trusted, its *status* is written into the data memory. [T] –A non-trusted node, or a node that is not capable of sharing a trusted digital signature, will be detected by the Gateway and any attempt to open a communication channel will refused.
T.3	**REQ_ND04 Policy updates**	D	The nS-ESD-GW is endowed by several communication policies. These policies specify and regulate the interrogations (interval, priority, broadcast, unicast, etc), the messages to be dispatched and accepted (message alive time, timeout, etc) and the Gateway modes. [D] – Policies are not upgradeable, avoiding by design the risk of malicious attacks.
T.4	**REQ_ND14 Storage of private information**	A, D	The confidentiality and the security of private information is ensured through the adoption of encryption/decryption blocks for the data writing. A non-volatile memory is the support for the storage of long-term information. [A] – During the secure operational mode execution all data that are stored into the memory are encrypted.
T.5	**REQ_ND21 Dynamic security behaviour**	A	In accordance to the policies of the nSHIELD framework, the nS-ESD-GW is able to evaluate the SPD level provided by the Middleware and to change its operational mode. This means that the Gateway is able to: • Increase/decrease the rate of the messages requests sent to legacy nodes; • Enable/disable cryptographic modules; • Increase/decrease the writing of the log file concerning the audit function. [A] – As consequence of the operational mode changing, log files are stored into the memory with a different rate. Likewise, data saved into the non-volatile support are encrypted if the status of the Gateway requires a more accurate behaviour.
T.6	**REQ_ND23 Hardware/Software co-design**	D	The platform used to develop the nS-ESD-GW prototype is a chip that integrates a hybrid architecture compose by a dual-core ARM Cortex A9 and a 7-Series Xilinx FPGA. This allows the possibility to use a specific design flow that speed up the entire development process and permits the coexistence, on the same device, of hardware-software co-design techniques.
T.7	**REQ_ND24 Situational-aware and context-aware SPD**	D	With the aim to optimize Gateway's performances, a coordination module is able to provide a services balancing according to the SPD level. [D] – A module that provides fault detection and SPD evaluation encompasses the nSHIELD algorithms. It consists of several software modules and some FPGA-based IP processing blocks.

3.3.8 Automatic Access Control (Prototype 11)

Verification procedure

Verify the proper operation of the Gossamer protocol. We initialize the Gossamer protocol for a server and a client. Then, we execute the protocol and prompt the communication messages – verifying that all messages are correct. Finally, we check that the two entities have correctly calculated the relevant data for the next session (forward security) and aren't desynchronized.

Verify the proper operation of the Gossamer protocol under a DoS attack. We initialize the server. Then, send random messages and observe if the server drops them instantly. We check if the server calculates the SPD metrics properly (e.g. transaction rate, failed authentication) and if it detects the DoS attack and arises an alarm.

Validation procedure

1. REQ_ND02 - Data Freshness (A, D): the Gossamer protocol by design enforces data freshness procedures to avoid replay attacks.

3.3.9 Face recognition prototype (Prototype 07)

The following table describes the requirements covered by the face recognition prototypes.

Table 3-13: Prototype 07 - Face Recognition identified test cases

ID	nSHIELD Requirement	Mean of verification	Description
VF.1	**REQ_VF01 Secure identification through face recognition**	T	The system should allow secure identification of end users through face recognition. The biometric profile of a person is acquired using face recognition and is used as a mean to securely identify end-users in the system. [T] – A set of synthetic and real tests is set up to understand the recognition capabilities of the system. The tests include synthetic images, real images and real pictures taken with a camera. These inputs include accredited biometric profiles, unknown biometric profiles, false positive and false negative.
VF.2	**REQ_VF03 Protection of biometric data and metadata storage**	D, A	Confidentiality of biometric data and metadata storage should, due to security and privacy reasons, be protected. The face recognition prototype doesn't allow an intruder to break the security of people identification process by extracting stored biometric data from the system. [D] – Biometric profiles are stored in a secure way. They are encrypted, both when they are passed from one module to the other and when they are stored. At the end of the recognition process these data are permanently deleted. [A] – The analysis of the recognition procedure allows the identification of possible weak points where the data confidentiality is compromised.
VF.3	**REQ_VF04 Biometric data privacy**	D, A	The acquired image must be deleted after analysis and cannot be transferred in any way outside the embedded device used for the acquisition and recognition. The illegal usage or diffusion of biometric data is prevented in the system due to privacy reasons. [D] – The biometric profile is not stored anyway on the system. It is represented in an internal data structure, encrypted and stored in memory. This data structure is deleted immediately after the result of the recognition process is available. [A] – The analysis of the recognition algorithm ensures that the biometric data are deleted after being used.

VF.4	**REQ_VF05 Biometric data and metadata confidentiality**	A, D	Biometric data and metadata shall be encrypted when used or transmitted for any purposes. In order to prevent an attacker to gain access to biometric data, the information acquired from the camera is encrypted when sent to or used by the processing unit. [D] – The recognition algorithm encrypts all the sensible data used during the recognition process. [A] – The analysis of the recognition algorithm can confirm that confidentiality is achieved thanks to encryption.
VF.5	**REQ_VF06 Face recognition quality**	T	Secure identification based on face recognition shall provide high quality results in order to reduce the risk of false identifications. The probability of false detection satisfies the ICAO standard. [T] – A set of specific tests has been conceived to measure the quality of the recognition. The tests and the datasets have been organized in an evaluation framework.
VF.6	**REQ_VF07 Biometric metadata privacy**	A, D	The metadata obtained from the analysis of images must be used respecting privacy. Although the metadata obtained from images are less sensible than their corresponding analogue sources, they always represent important information, especially when contextualized and, for this reason, they are treated respecting privacy. [D] – The biometric metadata are not stored anyway on the system. They are represented in an internal data structure, encrypted and stored in memory. This data structure is deleted immediately after the result of the recognition process is available. [A] – The analysis of the recognition procedure ensures that the biometric metadata are deleted after being used.
VF.7	**REQ_VF09 System fault tolerance**	D, T	The components of the recognition system shall use periodic keep alive messages, independent from the acquisition process, in order to inform each other of any fault occurred in the system. To ensure the dependability of the overall recognition system, the embedded device sends period keep alive message to the other components of the recognition system. [D] – The recognition system has been designed to enter in an alarm state if one or more components are, for any reason, not functioning properly. This emergency situation occurs when one or more components don't receive the expected keep alive messages. [T] – It is possible to recreate this faulty situation stopping one of the components or blocking it manually.
VF.8	**REQ_VF10** **Recognition process determinism**	A, T	The recognition process is self-learning and requires a training procedure. For its intrinsic nature it doesn't provide a binary result in terms of recognition but only a matching score and could be non-deterministic. In order to guarantee the reliability of the recognition process and the overall system, it must be "forced" to be deterministic. [A] – The analysis of the algorithm can demonstrate that it is deterministic. [T] – A specific set of test has been conceived to verify that the algorithm is deterministic. The test includes images that stresses the algorithm to the conditions that make the recognition fail.

VF.9	**REQ_VF11** **Onsite recognition**	A, D	The recognition (not the identification) of human faces in the input stream must be entirely performed by the embedded system on site. The features detection and recognition of face is performed entirely onsite and there is no need to send these sensible data to another computing unit over the network, and this increase security and privacy. [A] – The analysis of the feature detection component can verify this aspect. [D] – The feature extraction module is autonomous and doesn't need to communicate to the other modules, except to pass the results of the feature extraction. The module is installed on the embedded camera.
VF.10	**REQ_VF12** **Transmitted biometric information integrity**	D	Metadata transmitted over wireless or cable networks shall be protected in terms of integrity. It is not possible to modify biometric metadata that contain critical information used for user identification during any kind of data transmission. [D] – Metadata obtained by the recognition process are encrypted, both when they are used by the software modules and when they are transmitted.
VF.11	**REQ_VF13** **Secure execution**	D, T	The embedded system in charge of recognition should provide an isolated and protected execution environment that can guarantee the security during the recognition and identification process. It is not possible to interfere in any way with the system during the recognition and identification process. [D] – The components of the recognition system have been designed to strictly respect the presence of keep alive messages, timeouts and a handshaking based procedure. [T] – A set of tests, that alters this procedure, stops one or more modules, alter the handshaking timeline, has been used to verify this requirement.

4 Network Prototypes validation and verification

4.1 nSHIELD Network Requirements

An initial network requirements assessment has been conducted in nSHIELD deliverable D2.2: Preliminary System Requirements and Specifications. In this section, the requirements are recited and their approach is updated through the -up to now- available input from the technical developments of nSHIELD Network Layer design (WP4) and related work. The following table recapitulates nSHIELD network specifications under the view of validation and test processes. However it is useful to bear in mind the basic categorization criteria and dependency parameters that constitute the context of requirements analysis and design:

1. Significance: some requirements ensure the compliance with fundamental networking and security principles, where others are more loosely related with prerequisite actions. However, the vast majority of the 23 network requirements presented here fall in the first category.
2. Application: some requirements are more application specific than others or differently, each scenario (4 in nSHIELD) is best served by its own sheaf.
3. Functionality: the category includes requirements that explain what the system should do.
4. SPD level: different requirement types and levels are specified at various stages of nSHIELD subsystems operation.
5. Complexity: it would be desirable for a system to be designed in detailed compliance with each and every requirement. This, however, would result in losses in other fields that can be roughly summarized as “resources”: increased information overhead and demands for computational power are great consumers of energy and memory.

The reference notation of D2.2 is retained, whereas an indication declaring the current level of verification (A, D, I, T) is added, as these levels were introduced in the Introduction of this document and will be implemented in this first stage of Integration, Validation and Demonstration activities.

Table 4-1: Requirements relevant against validation and verification of Network layer prototypes

REQ_NW01 Confidentiality - REQ_NW08 Network Security Cryptographic Support A basic principle for a secure network implemented through data encryption. It is possibly applicable only on the most powerful of nSHIELD nodes. As for the specific cryptography scheme, in our preliminary analysis the employment of both symmetric and asymmetric cryptography is foreseen. However, this is a broad technical issue dependent on the aforementioned parameters (especially 2, 4 and 5) and possibly concerning a second stage of specific application management or specific component design (D,T).
REQ_NW02 Integrity Integrity refers to the network prerequisite of transmitting intact data packets. Applies with the trade-off of the previous one (meaning that is an important network feature but also resource consuming) (D, T).
REQ_NW03 Secure Routing - REQ_NW11 Reputation-Based Secure Routing One method to ensure Secure Routing in nSHIELD Network layer is through a Reputation-based scheme, which can build evidence of trustworthy cooperation fast by taking into account third-party ratings of trust beside first-hand observations. However the reputation-based trust mechanism can be also a target of attacks, the most important of which are: • Bad mouthing attack: an attacker node propagates false reputation values for malicious nodes, in order to increase their trust values. • On-Off attack: A malicious node behaves successively bad or well, aiming to confuse the trust calculation scheme. • Conflicting behaviour attack: a malicious node can follow a contradictory behaviour pattern towards two nodes. This will probably result in the two nodes estimating a low trust value for

each other, whereas there is no reason to do so. The verification for the requirement will be provided through testing (T).
REQ_NW04 Fault Tolerance - REQ_NW07 Availability - REQ_NW13 Fault Recovery The three requirements are grouped here to represent a desired property of an always available network. This is translated in a redundant network having the capability to adjust and overcome some nodes failing. (D).
REQ_NW05 Self-Management and Self-Coordination A node’s capability to organize itself is a very useful property. However, it is quite unfeasible to prescribe such a requirement in an overall network scale. Once more, it is the more powerful nodes that will implement more automated and autonomous functions, but selected self-management actions can be conducted by small nodes also. As an example we can refer to the propagation of trust metrics between nano/micro nodes, contributing in the formulation of trust tables and therefore more secure routing (A, D).
REQ_NW06 Multiple Protocol Support Since one of the core concepts of nSHIELD is service composability, component heterogeneity is highly probable and therefore the need for the network layer to provide support for a variety of protocols. Indicatively, application domains interconnecting PC formed data units, servers or workstations (running IP based protocols) with “clouds” of sensors (running WSN routing protocols) are foreseen (T).
REQ_NW09 Traceability That the nSHIELD network should provide traceability information on each transmitted packet is an intriguing requirement difficult to be accomplished in resource constrained sensor nodes with limited memory (A).
REQ_NW10 Audit nSHIELD network shall maintain and provide data and statistics reflecting information such as network performance and network status updates. A simple example is keeping log files in fixed intervals which show network’s reaction in the same or different conditions over time. This enables comparisons and assists in planning future actions. The requirement concerns mainly non-constrained nodes, having a centralized and supervisory role (D).
REQ_NW12 Reputation-Based Intrusion Detection The requirement will be covered by the IDS system described in nSHIELD D4.3, based on cooperation between nodes and fully distributed architectures. This module is part of the general nSHIELD Reputation scheme. Furthermore, intrusion detection can be implemented as a simple extension of Trusted Routing Scheme (T).
REQ_NW14 Application-Based Dependable Connectivity Each application presents specific communication needs that should be met to ensure dependable connectivity (T).
REQ_NW15 Dependable Authentic Key Distribution Mechanisms - REQ_NW23 Key Exchange Interfaces The requirements are listed here as they are related to Confidentiality. As referred previously, the adopted encryption mechanism and corresponding processes (e.g. key distribution methods, distributor’s ID, key propagation, man-in the middle attack and possible key alterations) are specialized topics that could be objectives of an application specific dedicated study. (A, D).
REQ_NW16 Reliable Transmission Methodologies The nSHIELD network layer shall provide waveform-agile and reliable transmission methodologies. The requirement will be addressed by the SDR radio platform, enabling smart SPD transmission and will be verified with SDR prototype (T).
REQ_NW17 Anonymity - REQ_NW18 Location Privacy The selection to fulfil this requirement and provide users with source anonymity comes along with computation overhead that may be prohibiting for the majority of nSHIELD nodes (A).

REQ_NW19 Application-Based Configurability Ensures that nSHIELD platform will be capable to adapt its operation and provide the SPD level each application demands (T).
REQ_NW20 Low Network Delay nSHIELD network designer will may have to balance between quick response times and cost, security or reliability (D, T).
REQ_NW21 Information Capacity The efficiency of available bandwidth depends (among others) on the applications needs, the topologies and technologies used (D, T).
REQ_NW22 Secure Channel Establishment Interfaces The nSHIELD network shall provide well-defined interfaces for establishing secure channels. The requirement seeks to ensure secure system scalability (D).

4.2 SHIELD Network prototypes overview

In this section a brief overview of the prototypes is given, that shows, at high level, how these requirements have been addressed in the design and development activities.

nSHIELD network prototypes are independent development efforts able to verify and validate that the network requirements identified in WP2 and summarized in the above section have been met. These prototypes can work as a proof of concept that the system behaves accordingly to the network specified rules providing a specific SPD level functionality and can be also considered as an input source to the nSHIELD application scenario demonstrators.

4.2.1 Reputation-Based Secure Routing prototype (Prototype 16)

The purpose of this prototype is to validate that fault-tolerant network connectivity can be guaranteed using a distributed reputation-based trust scheme running in each node of a wireless ad-hoc network. Two mechanisms have been taken into account during the design phase of the prototype to secure routing and protect the undisrupted network operation:

- Packets Overhearing: The purpose of this mechanism is to enable each node to obtain an evidence that every packet forwarded to the next node has been properly forwarded from this node towards the end network point in case that the next node is not the final destination itself. Implementing this mechanism can provide immunity to network-layer attacks starting from either malicious nodes (Black-hole, Gray-hole attacks) or from selfish behaviour.
- Validation of reputation-based information: using a modified Bayesian approach only second-hand reputation information that is not incompatible with the current reputation rating is accepted. This is important for every reputation-based scheme as malicious nodes can alter the ratings of their collaborating nodes forming attacks knows as bad mouthing attacks that can severely degrade the usefulness of a reputation-based scheme.

The table below summarizes the list of network requirements addressed by Prototype 16.

Table 4-2: Prototype 16 - Network Requirements addressed

Prototype 16: Reputation-Based Secure Routing prototype	Mean of Verification
REQ_NW03 Secure Routing	T
REQ_NW11 Reputation-Based Secure Routing	T
REQ_NW04 Fault Tolerance	T
REQ_NW05 Self-Management and Self-Coordination	T
REQ_NW07 Availability	T
REQ_NW19 Application-Based Configurability	T

4.2.2 Reputation-Based Secure Routing Prototype #2 (Prototype 16)

In WSNs, due to the open medium and the dynamic entrance of new nodes, there must be a way to establish trust relationships to avoid malicious entities. Trust and reputation-based schemes are used in wireless networking to provide secure routing functionality. A common approach for implementing secure routing functionality is the integration of a routing protocol with a reputation scheme.

For nSHIELD network layer, we implement a novel module reputation-based scheme that can act as a general purpose scheme for a wide range of applications. We identify the common components of reputation-based schemes and provide an abstract framework. We conclude in eleven components where each one of them serves a specific functionality. For every component we propose a set of features that implements the component's functionality. The segmentation of the scheme into components enables the dynamic deployment and extension of the scheme.

The network manager selects which components are active and the exact set of features that implements them. During the selection process, a designer could model the combination of more than one feature for some components. The designing options can range from ultra-lightweight schemes to heavily secure ones. The selection decision will be either static at deployment time or dynamic at run time, if such operation is supported. Also, heterogeneous nodes could utilize different features for some components. To make our proposal more applicable and acceptable, we have pre-set the configuration options for implementing the decision making process of well-known trust and reputation schemes for secure routing. The properties of the pre-defined schemes are known and each one of them protects the system against a set of specific attacks.

The trust and reputation system is implemented in C++ and extends the routing protocol DSR. It will be applied in BeagleBone and BeagleBoard devices.

4.2.3 SPD-driven Smart Transmission Layer prototype (Prototype 09)

Purpose of the prototype is demonstrating the provision of secure and robust communication in critical and hostile channel conditions. This was done via means of the simulated RF test-bench, consisting of 3 Software Defined Radio (SDR) handheld terminals, each interconnected with the OMBRA v2 (a representative of the nSHIELD Power node) as the processing platform.

The prototype allows for emulating various types of interferers in the channel, as well as creating and deploying appropriate counter-mechanisms.

The following table summarizes the list of network requirements addressed by Prototype 09.

Table 4-3: Prototype 09 - Network Requirements addressed

Prototype 09: SPD-driven Smart Transmission Layer	Mean of Verification
REQ_NW01 Confidentiality	A
REQ_NW02 Integrity	T
REQ_NW04 Fault Tolerance	T
REQ_NW05 Self-Management and Self-Coordination	D,T
REQ_NW07 Availability	D,T
REQ_NW13 Fault Recovery	D,T
REQ_NW16 Reliable Transmission Methodologies	D,T
REQ_NW19 Application-Based Configurability	T
	T

4.2.4 Link Layer Security prototype (Prototype 23)

Purpose of the prototype is demonstrating the provision of secure communication on the link layer, and to do so the protocol to test should provide access control, message integrity, message confidentiality and replay protection.

The following table summarizes the list of network requirements addressed by Prototype 23.

Table 4-4: Prototype 23 - Link Layer Security requirements addressed

Prototype 23: Link Layer Security prototype	Mean of Verification
REQ_NW01 Confidentiality	T
REQ_NW02 Integrity	T

4.2.5 DoS attack Defence (Prototype 12)

The purpose of this prototype is to demonstrate the operation of the developed algorithms. The prototype consists of the cooperation of two algorithms that run as software processes in the system, the Statistical analysis algorithm and the Pattern matching algorithm. The Statistical analysis algorithm communicates with all four input modules, reads and processes that information and the Pattern matching algorithm communicates with the network traffic module. Its task is to sample network packets and compare them with the signature database.

Initially, requirements of the network layer have been considered in the design process of the algorithms. The self-management requirement was considered and as a result the algorithms can operate in single-node mode, processing and correlating information that relates to the single node they operate. As the analysis algorithm gets positive detections, these are stored in the node memory and re-used by the matching algorithm. These allows for the algorithms to be self-managed without information from other nodes. The audit requirement is embedded in the design of the algorithms since all positive detections are logged for the operation of the algorithm and can later be reviewed together with all the input information that resulted in the positive detection. The Low Network delay requirement was considered in the design of the algorithms. The mechanism by which this is achieved is the sampling of the network data which can is adjustable and can be set to values that don’t cause any load in the exchange of data in the network.

Table 4-5: Prototype 12 - DoS Attack Defence requirements addressed

Prototype 12: DoS attack Defence	Mean of Verification
REQ_NW05 Self-Management and Self-Coordination	D, A
REQ_NW10 Audit	D, A
REQ_NW20 Low Network Delay	D, A

4.2.6 Network Layer Security prototype (Prototype 24)

The network layer security prototype deals with the ability to provide message protection at the network layer. The deployed protocol provides end-to-end confidentiality and integrity satisfying applications’ requirements. It is an adaptation of the standardized IPsec protocol, specifically designed to satisfy the restricted nodes of the nSHIELD network.

The following table *summarizes the list of network requirements addressed by Prototype 24.*

Table 4-6: Prototype 24 - Network Requirements addressed

Prototype 24: Network Layer Security prototype	Mean of Verification
REQ_NW01 Confidentiality	D, T
REQ_NW02 Integrity	D, T
REQ_NW06 Multiple Protocol Support	A,D
REQ_NW08 Network Security Cryptographic Support	D
REQ_NW19 Application-Based Configurability	D,T
REQ_NW20 Low Network Delay	D, T

4.2.7 Anonymity & Location Privacy service (Prototype 10)

This prototype is intended for applications where personal location privacy must be preserved while enabling the system to provide location monitoring services as well as users' access to location-based services. The mechanism implemented is lightweight, relies on the K-anonymity privacy concept that aims to make a user indistinguishable from "K" of her neighbours.

The table below summarized the list of network requirements addressed by Prototype 10.

Table 4-7: Prototype 10 - Network Requirements Addressed

Prototype 10: Anonymity & Location Privacy service	**Mean of Verification**
REQ_NW17 Anonymity	A, D, T
REQ_NW18 Location Privacy	A, D, T

4.3 SHIELD Network prototypes verification and validation

In this section a description of the verification and validation activities is reported, with particular attention to those prototypes whose associated requirements are marked with a T (i.e. a test is needed or verifies that the requirement is met).

4.3.1 Reputation-Based Secure Routing prototype (Prototype 16)

This prototype testing is based on Reputation-Based Trust module developed from HAI for Memsic IRIS sensor nodes which are IEEE 802.15.4 compliant at physical and MAC layers. The programming environment used is TinyOS 2.x. The routing engine implements Greedy Perimeter Stateless Routing (GPSR) a geographical-type routing algorithm with precompiled position coordinates. For the testing purposes the topology of the figure below was used in a controlled neighbourhood manner – each node rejects as neighbours other nodes than those depicted in the figure, even in cases where data link connectivity is feasible. On top of the routing module a reputation-based trust module is implemented which interacts with packet overhearing to rate the forwarding behaviour of the next node and take countermeasures in cases of routing attacks like Black-hole and Gray-hole attacks selecting a well behaving node for packet forwarding. To build trust evidence more quickly, a reputation based scheme which takes into account indirect trust was build. The system is tested against attacks that can degrade reputation performance in the form of bad mouthing attack where a node advertises deliberately fault trust values for nodes that interact with.

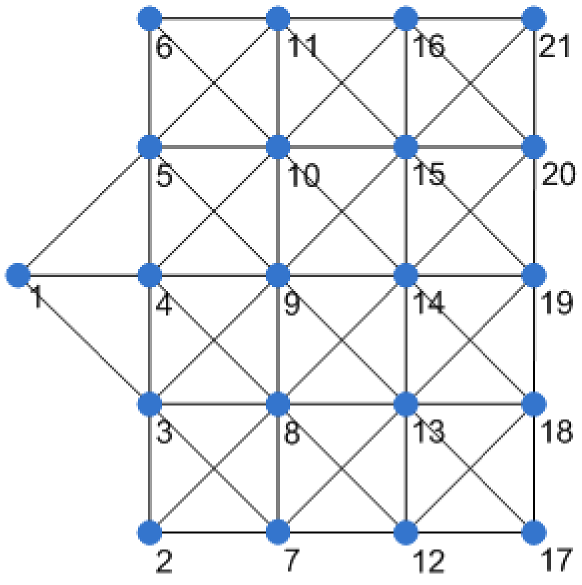

Figure 4-1: Test bed composition and nodes' connectivity (reputation-based secure routing)

Four verification procedures for various types and combinations of routing attacks have been developed for prototype 16:

Table 4-8: Prototype 16 - Verification procedures

Verification Procedures for Reputation Based Secure Routing	**Test Description**
Test Nr 4.3.1.1	Security Verification against Black-hole attack
Test Nr 4.3.1.2	Security Verification against Gray-hole attack
Test Nr 4.3.1.3	Security Verification against Bad mouthing attack
Test Nr 4.3.1.4	Security Verification against several simultaneous attacks

For the validation that the operation of Prototype 16 meets all security requirements of Table1, the following validation procedure was developed.

Table 4-9: Prototype 16 - Validation procedure

Validation Procedure for Prototype 16	**Test Description**
Test Nr 4.3.1.5	Validate that for Prototype 16 all network requirements of Table 1 are met

Table 4-10: Prototype 16 - Verification test #1

Verification Test Nr.: 4.3.1.1	**Written by:** HAI		**Conducted by:**	**Date:**	**Test Category:** Network Security
Software and Hardware Configuration Details	• 20 Memsic IRIS nodes with mda100 sensor-boards • Nodes compiled with T-GPSR (v X.X) • 1 Base-station IRIS node (connected to pc) compiled with Base-Station (v X.X) • PC running GUI application monitoring routing paths using a Sniffing node • 1 node performing Black-Hole attack				
Test Name:	***Secure Routing Verification against Black-hole attack***				
Purpose:	Verify that the routing layer of the sensor node performs correctly under a black-hole attack				
Modules/Interfaces/Code Tested:	Network layer module Reputation-based trust management module				
Step	**Action**		**Expected Result**	**Pass/Fail**	**Remarks**
1	Compile all nodes comprising the island so that both distance and trust weighting factor are taken into account during routing decision. Compile one node as black-hole attacker dropping all forwarding messages. Only node 1 sends data packets in order to keep a clear track of the routing path.		-		
2	Switch on all nodes.		Network layer control messages (Beacons) are exchanged among nodes. Routing tables are filled according to trust-enabled GPSR algorithm. Multihop routing is enabled. Sniffer overhears all messages exchanged among the nodes.		
3	Switch on Base-Station node.		Base-Station receives beacons, data packets, etc exchanged among the nodes.		
4	Start transmitting application-layer messages from node 1.		Messages received at BS (write down the route, number of hops). The route and the number of hops can be monitored from the PC application connected to sniffing node.		
5	Replace one of the nodes on the route with one compiled to perform as a black-hole attacker.		Messages received at BS via a different route, bypassing the BH node (write down the route, number of hops). The route and the number of hops can be monitored from the PC application connected to sniffing node.		
6	Repeat the test placing black-hole attacker to another place.		The network is able to bypass the BH attacker in every network place.		
7	Switch off all nodes of the island.				

Table 4-11: Prototype 16 - Verification test #2

Verification Test Nr.: 4.3.1.2	**Written by:** HAI	**Conducted by:**	**Date:**	**Test Category:** Network Security
Software and Hardware Configuration Details	• 20 Memsic IRIS nodes with mda100 sensor-boards • Nodes compiled with T-GPSR (v X.X) • 1 Base-station IRIS node (connected to pc) compiled with Base-Station (v X.X) • PC running GUI application monitoring routing paths using a Sniffing node • 1 node performing Gray-Hole attack			
Test Name:	***Secure Routing Verification against Gray-hole attack***			
Purpose:	Verify that the routing layer of the sensor node performs correctly under a gray-hole attack			
Modules/Interfaces/Code Tested:	Network layer module Reputation-based trust management module			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Compile all nodes comprising the island so that both distance and trust weighting factor are taken into account during routing decision. Compile one node as gray-hole attacker randomly dropping some forwarding messages. Only node 1 sends data packets in order to keep a clear track of the routing path.	-		
2	Switch on all nodes.	Network layer control messages (Beacons) are exchanged among nodes. Routing tables are filled according to trust-enabled GPSR algorithm. Multihop routing is enabled. Sniffer overhears all messages exchanged among the nodes.		
3	Switch on Base-Station node.	Base-Station receives beacons, data packets, etc. exchanged among the nodes.		
4	Start transmitting application-layer messages from node 1.	Messages received at BS (write down the route, number of hops). The route and the number of hops can be monitored from the PC application connected to sniffing node.		
5	Replace one of the nodes on the route with one compiled to perform as a gray-hole attacker.	Messages received at BS via a different route, bypassing the gray-hole node (write down the route, number of hops). The route and the number of hops can be monitored from the PC application connected to sniffing node.		
6	Repeat the test placing gray-hole attacker to another place	The network is able to bypass the gray-hole attacker in every network place.		
7	Switch off all nodes of the island.			

Table 4-12: Prototype 16 - Verification test #3

Verification Test Nr.: 4.3.1.3	**Written by:** HAI	**Conducted by:**	**Date:**	**Test Category:** Network Security
Software and Hardware Configuration Details	• 20 Memsic IRIS nodes with mda100 sensor-boards • Nodes compiled with T-GPSR (v X.X) • 1 Base-station IRIS node (connected to pc) compiled with Base-Station (v X.X) • PC running GUI application monitoring routing paths using a Sniffing node • 1 node performing Bad-mouthing attack			
Test Name:	***Secure Routing Verification against Bad-mouthing attack***			
Purpose:	Verify that the routing layer of the sensor node performs correctly under a Bad-mouthing attack			
Modules/Interfaces/Code Tested:	Network layer module Reputation-based trust management module			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Compile all nodes comprising the island so that both distance and trust weighting factor are taken into account during routing decision. Compile one node as bad-mouthing attacker advertising deliberately false trust ratings in its reputation messages. Only node 1 sends data packets in order to keep a clear track of the routing path.	-		
2	Switch on all nodes.	Network layer control messages (Beacons) are exchanged among nodes. Routing tables are filled according to trust-enabled GPSR algorithm. Multihop routing is enabled. Sniffer overhears all messages exchanged.		
3	Switch on Base-Station node.	Base-Station receives beacons, data packets, etc. exchanged among the nodes.		
4	Start transmitting application-layer messages from node 1.	Messages received at BS (write down the route, number of hops). The route and the number of hops can be monitored from the PC application connected to sniffing node.		
5	Replace one of the nodes on the route with one compiled to perform as a bad-mouthing attacker.	When reputation message is received from the attacker, forwarding from the most trusted path must be continued neglecting false rating advertised by bad-mouthing attacker. Messages received at BS (write down the route, number of hops). The route and the number of hops can be monitored from the PC application connected to sniffing node.		
6	Repeat the test placing bad-mouthing attacker to another place.	The network is able to bypass the BH attacker in every network place.		
7	Switch off all nodes of the island.			

Table 4-13: Prototype 16 - Verification test #4

Verification Test Nr.: 4.3.1.4	**Written by:** HAI	**Conducted by:**		**Date:**	**Test Category:** Network Security
Software and Hardware Configuration Details	• 20 Memsic IRIS nodes with mda100 sensor-boards • Nodes compiled with T-GPSR (v X.X) • 1 Base-station IRIS node (connected to pc) compiled with Base-Station (v X.X) • PC running GUI application monitoring routing paths using a Sniffing node • 1 mote performing black-hole, 1 mote performing gray-hole and 1 mote performing bad-mouthing attacks				
Test Name:	***Secure Routing Verification against several attacks***				
Purpose:	Verify that the routing layer of the sensor node performs correctly under a number of network-layer attacks				
Modules/Interfaces/Code Tested:	Network layer module Reputation-based trust management module				
Step	**Action**		**Expected Result**	**Pass/Fail**	**Remarks**
1	Compile all nodes comprising the island so that both distance and trust weighting factor are taken into account during routing decision.		-		
2	Switch on all nodes.		Network layer control messages (Beacons) are exchanged among nodes. Routing tables are filled according to trust-enabled GPSR algorithm. Multihop routing is enabled. Sniffer overhears all messages exchanged.		
3	Switch on Base-Station node.		BS receives beacons, data packets, etc. exchanged among the nodes.		
4	Start transmitting messages from node 1.		Messages received at BS (write down the route, number of hops). The route and the number of hops can be monitored from the PC application connected to sniffing node.		
5	Replace one of the nodes on the route with a node compiled to perform as a black-hole attacker.		Messages received at BS via different route, bypassing the black-hole attack (write down the route, number of hops).		
6	Replace one of the nodes on the new route with a node compiled to perform as a gray-hole attacker.		Messages received at BS via different route, bypassing the gray-hole attack (write down the route, number of hops).		
7	Replace one of the nodes on the new route with a node compiled to perform as a bad-mouthing attacker.		Messages received at BS via the most trusted network path (write down the route, number of hops).		
8	Switch off all nodes of the island.				

Table 4-14: Prototype 16 - Validation test #1

Validation Test Nr.: 4.3.1.5	**Written by:** HAI	**Conducted by:**	**Date:**	**Test Category:** Network Security
Software and Hardware Configuration Details	• 20 Memsic IRIS nodes with mda100 sensor-boards • Nodes compiled with T-GPSR (v X.X) • 1 Base-station IRIS node (connected to pc) compiled with Base-Station (v X.X) • PC running GUI application monitoring routing paths using a Sniffing node • 1 mote performing black-hole, 1 mote performing gray-hole and 1 mote performing bad-mouthing attacks			
Test Name:	***Test Validation of network requirements for Prototype 16***			
Purpose:	Test Validation of nSHIELD Network requirements: • REQ_NW05 Self-Management and Self-Coordination • REQ_NW04 Fault Tolerance • REQ_NW07 Availability • REQ_NW03 Secure Routing • REQ_NW11 Reputation-Based Secure Routing • REQ_NW19 Application-Based Configurability			
Modules/Interfaces/Code Tested:	Network layer module Reputation-based trust management module Security configuration module			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Compile all nodes comprising the island so that both distance and trust weighting factor are taken into account during routing decision.	-		
2	Switch on all nodes.	Network layer control messages (Beacons) are exchanged among nodes. Routing tables are filled according to trust-enabled GPSR algorithm. Multihop routing is enabled. Sniffer overhears all messages exchanged.		Validation of Self-Management and Self-Coordination of network layer operation.
3	Switch on Base-Station node.	BS receives beacons, data packets, etc. exchanged among the nodes.		

4	Start transmitting messages from node 1.	Messages received at BS (write down the route, number of hops). The route and the number of hops can be monitored from the PC application connected to sniffing node.		
5	Switch off a network node.	Network must be available and fault tolerant routing traffic from a different path.		Validation of Network availability and fault tolerance.
6	Replace one of the nodes on the route with a node compiled to perform as a black-hole attacker.	Messages received at BS via different route, bypassing the black-hole attack (write down the route, number of hops).		
7	Replace one of the nodes on the new route with a node compiled to perform as a gray-hole attacker.	Messages received at BS via different route, bypassing the gray-hole attack (write down the route, number of hops).		
8	Replace one of the nodes on the new route with a node compiled to perform as a bad-mouthing attacker.	Messages received at BS via the most trusted network path (write down the route, number of hops).		Validation of Reputation-Based Secure Routing which counteracts against a number of security attacks (Steps 6-9).
9	Configure from the PC application the security level of a node.	Node network security is configured to the desired SPD level: 1. Trust disabled 2. Direct Trust enabled 3. Weighted Direct Trust and Indirect Trust enabled 4. Direct Trust + Beta distribution Indirect Trust enabled		Validation of network security level configuration according to application requirements. Quantitative performance comparison of different approaches will be conducted.
10	Switch off all nodes of the island.			

4.3.2 Reputation-Based Secure Routing Prototype #2 (Prototype 16)

The following table summarizes the verification and validation process.

Table 4-15: Prototype 16 (#2) – Verification

Prototype 16: Reputation-Based Secure Routing prototype #2	Verification	Validation
REQ_NW03 Secure Routing	T	Validate that the scheme is resilient against attacks on the pure routing protocol (e.g. black whole attack)
REQ_NW04 Fault Tolerance	D	The scheme provides fault tolerance mechanisms
REQ_NW05 Self-Management and Self-Coordination	A, D	The scheme is designed as a self-management and self-coordination system
REQ_NW07 Availability	D	The scheme provides protection against several attacks and ensures availability under them
REQ_NW11 Reputation-Based Secure Routing	T	Validate that the scheme is resilient against attacks on the pure routing protocol (e.g. black whole attack) and the trust & reputation scheme (e.g. badmouthing attack)
REQ_NW12 Reputation-based intrusion detection	T	Detection of attacks on the routing protocol and the trust & reputation scheme
REQ_NW19 Application-Based Configurability	T	The scheme can be configured to comply with the applications requirements
REQ_NW20 Low network delay	D, T	The pre-defined schemes produce specific network delay and computation overhead. For a configured scheme the aforementioned parameters must be evaluated

4.3.3 SPD-driven Smart Transmission Layer Prototype (Prototype 09)

This prototype testing is based on the test bed implementation consisting of 3 SWAVE HH Software Defined Radios, 3 OMBRA v2 nSHIELD Power nodes, a spectrum analyser, a vector signal generator and several auxiliaries allowing the components' interconnectivity.

The software environment includes a combination of:

- Proprietary C++ libraries, algorithms and functions, running on the Power node
- SOAP XML web service, running on the Power node
- Simple Network Management Protocol (SNMP) client, running on the Power node
- Software Communications Architecture-compliant operating system, running on SWAVE HH

By continuously receiving and interpreting the instructions from the Overlay regarding the desired SPD-level, SPD-driven STL provides several on-the-fly reconfigurability potentials of the radios to accommodate the current needs, such as:

- Power state of the radios (on/off)
- Transmit power of the radios
- Deployed waveform on the radios
- Transmit frequency on the radios

- Modulation type (waveform-dependant) on the radios
- Deployed cryptography

Vector signal generator will be used as the means for creating various types of interference.

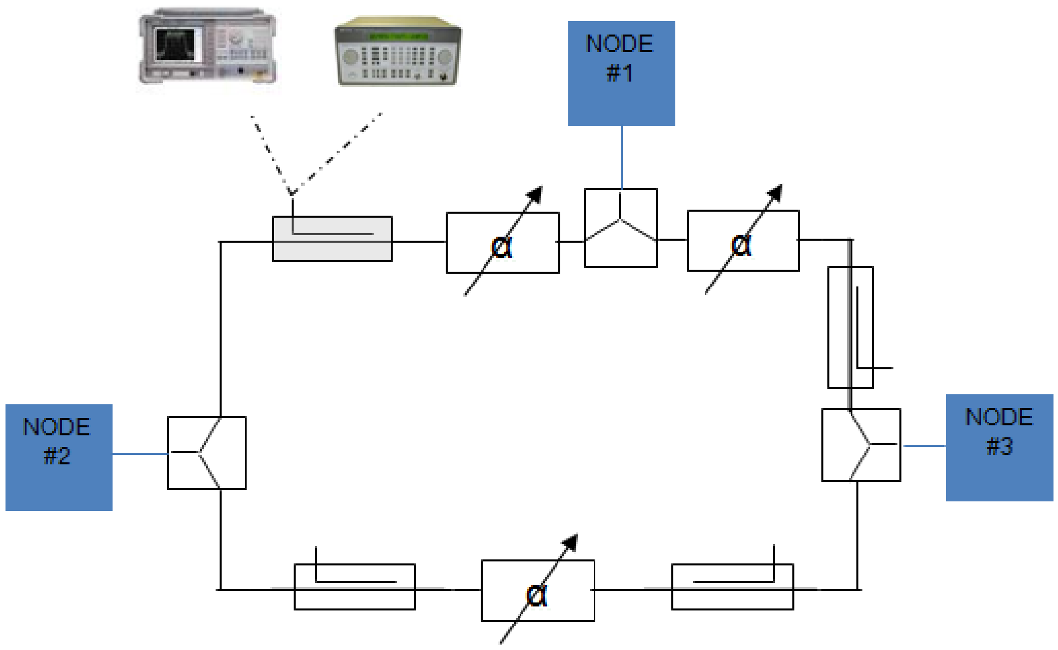

Figure 4-2: Test bed block diagram

Correct functioning of each of the components comprising the test bed needs to be verified on an individual basis. Then, a set of verification and validation methods of the prototype's developed functionalities-of-interest may be performed. These are described as follows:

Table 4-16: Prototype 09 - Verification procedures

Verification Procedures for Prototype 09	Test Description
Test Nr 4.3.3.1	Individual verification of the correct functioning of each of the test bed's components
Test Nr 4.3.3.2	Verification of the remote control functionality of the radios
Test Nr 4.3.3.3	Verification of the spectrum sensing data acquisition functionality of the radios
Test Nr 4.3.3.4	Verification of the waveform identification network functionality
Test Nr 4.3.3.5	Verification of the developed jamming mitigation algorithms

For the validation that the operation of Prototype 09 meets all the defined security requirements, the following validation procedure was developed.

Table 4-17: Prototype 09 - Validation procedure

Validation Procedures for Prototype 09	Test Description
Test Nr 4.3.3.6	Validate that for Prototype 09 all of the described network requirements are met

Table 4-18: Prototype 09 - Verification test #1

Verification Test Nr.: 4.3.3.1		**Written by:** UNIGE	**Conducted by:**	**Date:**	**Test Category:** Assessment of individual components
Software and Hardware Configuration Details		• 3 SWAVE HHs • 3 OMBRA v2 Power Nodes running Windows CE • 3 RS232-RS485 converters • 2 dual directional couplers for the frequency range of interest • 3 programmable attenuators (min 30dB) • Vector signal generator • Spectrum analyser			
Test Name:		***Test bed's individual components assessment***			
Purpose:		Verify that each of the components is functioning as envisioned			
Modules/Interfaces/Code Tested:		???			
Step	**Action**		**Expected Result**	**Pass/Fail**	**Remarks**
1	Switch on SWAVE HHs.		Boot-up screen turns on, followed by the "username and password" menu.		
2	Ensure that the tested SWAVE HHs have a same waveform, modulation and carrier frequency set. Start transmitting several bursts of speech/data. Repeat procedure, pairing up all of the tested HHs.		Speech/data should be received at the observed HH terminals.		
3	Switch on OMBRA v2 Power Nodes.		Windows CE boots.		
4	Connect the vector signal generator to the spectrum analyser via the coaxial RF cable. Create a signal via the VSG and verify that it is correctly seen on the SA.		Whichever signals were created by the VSG should be visible on the SA.		
5	Set a pair of programmable attenuators to at least 30dB attenuation value, connect them on the ports of the directional coupler, and connect two SWAVE HHs. Start transmitting voice/data bursts. Repeat for different pairs of attenuators and HHs.		Voice/data bursts should be received without too many errors, even with the low transmit powers.		
6	Turn off all the components.				

Table 4-19: Prototype 09 - Verification test #2

Verification Test Nr.: 4.3.3.2	**Written by:** UNIGE	**Conducted by:**	**Date:**	**Test Category:** Node functionality
Software and Hardware Configuration Details	• 3 SWAVE HHs • 3 OMBRA v2 Power Nodes running Windows CE, with SNMP v3 clients installed • 3 Ethernet cables			
Test Name:	***Remote control of the radio***			
Purpose:	Verify that the remote control of the radio functions properly			
Modules/Interfaces/Code Tested:	???			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Via Ethernet cable, connect each SWAVE HH with an OMBRA v2 Power node (or a PC) with installed SNMP client. Make sure HH and PN/PC are on the same domain (might need to set the IP address manually), and try pinging the HH via cmd terminal.	Ping should invoke “reply” from the HH, i.e. sent packets should be received		
2	Open SNMP client, and load the appropriate MIB table. Set the appropriate parameters for the SNMP v3 protocol, and scan for devices	Scanning should recognize the HHs that wish to be targeted		
3	Invoke SET/GET/TRAP commands for the wanted radio parameters (from the MIB table)	Radio’s parameters may be read/altered/flagged, depending on the invoked command		
4	Turn off all the components.			

Table 4-20: Prototype 09 - Verification test #3

Verification Test Nr.: 4.3.3.3	**Written by:** UNIGE	**Conducted by:**	**Date:**	**Test Category:** Node functionality
Software and Hardware Configuration Details	• 3 SWAVE HHs • 3 OMBRA v2 Power Nodes running Windows CE, with serial terminals installed • 3 RS232-RS485 converters			
Test Name:	***Spectrum sensing data acquisition verification***			
Purpose:	Verify that spectrum sensing data is read and shown in a correct manner			
Modules/Interfaces/Code Tested:	???			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Connect each HH to the PN via serial interface. Due to different interfaces (RS232 vs. RS 485), a converter needs to be used.			
2	Open a serial terminal on the PN, and set the read speed to 115200 bps, 8/1. Turn on the data logger and store read data.	Every 20 seconds, 8192 words (2B each) should be transmitted in raw format for the band-of-interest.		
3	Convert the stored data to HEX values following instructions provided in D4.2 and D4.3	Each of the HEX values represents an FFT value at a certain frequency.		
4	Plot the data.	A representation of a current spectrum status for the band-of-interest should be plotted		
5	Turn off all the components.			

Table 4-21: Prototype 09 - Verification test #4

Verification Test Nr.: 4.3.3.4	**Written by:** UNIGE	**Conducted by:**	**Date:**		**Test Category:** Network security
Software and Hardware Configuration Details	• 3 SWAVE HHs • 3 OMBRA v2 Power Nodes running Windows CE • 3 RS232-RS485 converters • 2 dual directional couplers for the frequency range of interest • 3 programmable attenuators (min 30dB) • Vector signal generator • Spectrum analyzer • Waveform identification – proprietary software				
Test Name:	***Waveform identification verification***				
Purpose:	Newly detected waveform’s features are compared to those of allowed waveforms from the database, and categorized				
Modules/Interfaces/Code Tested:	???				
Step	**Action**	**Expected Result**		**Pass/Fail**	**Remarks**
1	Assemble the STL test bed and start the “waveform identification” executable on one of the PNs				
2	Transmit a waveform through one of the HHs.	By comparing its characteristics with the ones from the database, the waveform should be recognized and characterized as “known, non-malicious”			
3	Create a signal somewhere in the band-of-interest by a vector signal generator	Waveform should not correspond to a list of the waveforms from the database, and should be classified as “unknown, potentially malicious”			
4	Turn off all the components.				

Table 4-22: Prototype 09 - Verification test #5

Verification Test Nr.: 4.3.3.5	**Written by:** UNIGE	**Conducted by:**	**Date:**	**Test Category:** Network dependability
Software and Hardware Configuration Details	• 3 SWAVE HHs • 3 OMBRA v2 Power Nodes running Windows CE • 3 RS232-RS485 converters • 2 dual directional couplers for the frequency range of interest • 3 programmable attenuators (min 30dB) • Vector signal generator • Spectrum analyzer • Jamming mitigation – proprietary software			
Test Name:	***Jamming attack mitigation***			
Purpose:	Verify that spectrum sensing data is read and shown in a correct manner			
Modules/Interfaces/Code Tested:	???			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Assemble the STL test bed and start the “jamming mitigation” executable on the PNs which are to be used as receivers			
2	Start transmitting from the other HH at frequency *f*	Signal should be received and decoded without significant interference. This may also be verified by running the BER test on the HHs		
3	Using vector signal generator, create a significantly powerful interfering signal at frequency *f*	Interfering signal is observable on the spectrum analyser, and is superimposed on the original signal. Communication is significantly degraded (also verifiable by BER test), and the channel surfing/waveform switching/power allocation algorithm is initiated. Communication should be improved significantly, depending on which jamming mitigation algorithm was deployed.		
4	Alter the created interfering signal’s parameters to successfully jam the altered transmission.	- | | -		
5	Turn off all the components.			

Table 4-23: Prototype 09 - Validation test #1

Validation Test Nr.: 4.3.3.6	**Written by:** UNIGE	**Conducted by:**	**Date:**	**Test Category:** Network Security and Dependability
Software and Hardware Configuration Details	• 3 SWAVE HHs • 3 OMBRA v2 Power Nodes running Windows CE • 3 RS232-RS485 converters • 2 dual directional couplers for the frequency range of interest • 3 programmable attenuators (min 30dB) • Vector signal generator • Spectrum analyzer • Jamming mitigation – proprietary software • Modified SOAP library - interface with the Overlay Layer			
Test Name:	***Test Validation of network requirements for Prototype 09***			
Purpose:	Test Validation of nSHIELD Network requirements: - REQ_NW01 Confidentiality - REQ_NW02 Integrity - REQ_NW04 Fault Tolerance - REQ_NW05 Self-Management and Self-Coordination - REQ_NW07 Availability - REQ_NW13 Fault Recovery - REQ_NW16 Reliable Transmission Methodologies - REQ_NW19 Application-Based Configurability			
Modules/Interfaces/Code Tested:	???			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Assemble the STL test bed and start the “waveform identification” and “jamming mitigation” executable on the PNs which are to be used as receivers			
2	Test the communication of each of the terminals with the Overlay – a client-server communication is instantiated using the modified SOAP web service	Log file is created, describing whether the communication is successful, i.e. terminal can successfully receive commands from Overlay		

3	Set the initial SPD level for each of the nodes (SPD level is communicated through SOAP)	Each node should display a set of algorithms that the current SPD level supports.		
4	Create interfering signal as in test 4.3.2.5.	A set of security measures corresponding to the given SPD level will be invoked. E.g. for high SPD level, channel surfing may be the preferred anti-jamming method, whereas for low SPD level, power allocation may be preferred		
5	Change the SPD level for the nodes.	Each node should display a set of algorithms that the current SPD level supports.		
6	Modify the interfering signal to successfully degrade the re-establish communication	Different anti-jamming algorithm should be invoked in this case		
7	TBD			
8	TBD			
9	TBD			
10	Turn off all the nodes			

4.3.4 Link Layer Security prototype (Prototype 23)

This prototype testing is bases on Link Layer Security module developed by INDRA for Zolertia Z1 sensor nodes which are IEEE 802.15.4 compliant. The programming environment used is TinyOS 2.x. The security link layer protocols used according to IEEE 802.15.4 standard are CTR, CBC-MAC, CCM.

For the testing purposes we have used three nodes, two of them used as senders and the other one used as base station.

One of the senders will have a different key value than the base station and the other sender.

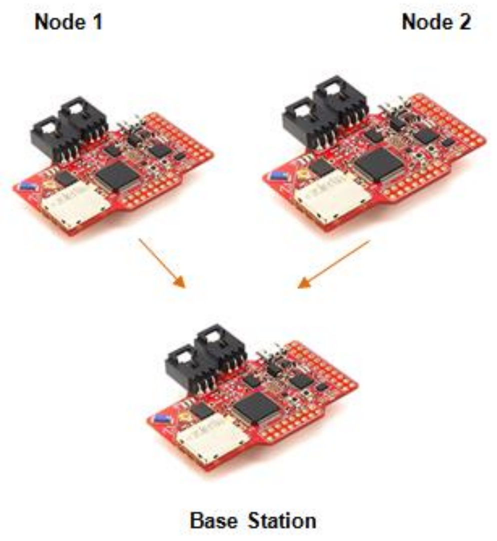

Figure 4-3: Test bed block diagram

Following tables summarizes the verification and validation procedures.

Table 4-24: Prototype 23 - Verification Procedure

Verification Procedures for Prototype 23	Test Description
Test Nr 4.3.4.1	Verification of integrity and authentication

Table 4-25: Prototype 23 - Validation Procedure

Verification Procedures for Prototype 23	Test Description
Test Nr 4.3.4.2	Validate that for Prototype 23 all of the described network requirements are met

Table 4-26: Prototype 23 - Verification test #1

Verification Test Nr.: 4.3.4.1	**Written by:** INDRA	**Conducted by:**	**Date:**	**Test Category:** Assessment of individual components
Software and Hardware Configuration Details	• 3 z1 motes, 1 as base station and two as senders with different key values • Sniffer			
Test Name:	***Test Link layer security***			
Purpose:	Verify that each of the components is functioning as envisioned			
Modules/Interfaces/Code Tested:	???			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Compile all nodes with tinyOS 2.x operative system. One receiver and sender with the same key value and the other sender with other key value			
2	Switch on senders	The senders begin to send data		
3	Switch on receiver	The receiver, as base station, sends received data to the serial port		
4	Print serial port information	If CTR algorithm has been used, the data is encrypted. If CBC-MAC algorithm has been used, MIC code is added to the frame providing integrity but data isn’t encrypted. If CCM algorithm has been used, MIC code is added to the frame and data is encrypted. With CBC-MAC and CCM packets processed with a different key will be discarded.		
5	Turn off all the components			

4.3.5 Dos Attack Defence (Prototype 12)

4.3.5.1 Verification Procedure

This was achieved by code review (A) against the relevant requirements.

4.3.5.2 Validation Procedure

Simulations were performed in the OMNET++ simulator which achieved the validation of basic functionality against requirements.

4.3.6 Network Layer Security prototype (Prototype 24)

This prototype testing is based on Network Layer Security module developed by TUC for Zolertia Z1 sensor nodes which are IEEE 802.15.4 compliant. The programming environment used is Contiki OS 2.6. The network layer security protocols used are compatible with the IEEE 802.15.4 standard.

For the testing purposes we have used three nodes, two of them used as legitimate communicating parties and the other one used as malicious/eavesdropping entity.

Correct functioning of each of the components comprising the test bed needs to be verified on an individual basis. Then, a set of verification and validation methods of the prototype's developed functionalities-of-interest may be performed. These are described as follows:

Table 4-27: Prototype 24 - Verification procedures

Verification Procedures for Prototype 24	Test Description
Test Nr 4.3.6.1	Verification of confidentiality
Test Nr 4.3.6.2	Verification of the integrity feature
Test Nr 4.3.6.3	Verification that an application can set the level of security provided
Test Nr 4.3.6.4	Verification that the mechanism introduces low delay

For the validation that the operation of Prototype 24 meets all the defined security requirements, the following validation procedure was developed.

Table 4-28: Prototype 24 - Validation procedure

Validation Procedure for Prototype 24	Test Description
Test Nr 4.3.6.5	Validate that for Prototype 24 all of the described network requirements are met

Table 4-29: Prototype 24 - Verification test #1

Verification Test Nr.: 4.3.6.1	**Written by:** TUC	**Conducted by:**	**Date:**	**Test Category:** Network Security
Software and Hardware Configuration Details	• 3 Z1 sensor nodes • Nodes running Contiki OS 2.6 and the IPsec protocol with AES in CCM* mode. • PC running GUI application monitoring communication using a Sniffing node			
Test Name:	***Verification of confidentiality among communicated messages***			
Purpose:	Verify that the network layer messages are encrypted (REQ_NW01).			
Modules/Interfaces/Code Tested:	Network layer module IPsec module			
Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Compile all nodes comprising the scenario. Only node 1 sends data packets in order to keep track of the communication.	-		
2	Switch on all nodes.	Network layer control messages are exchanged among nodes. Sniffer overhears all messages exchanged among the nodes.		
4	Start transmitting application-layer messages from node 1.	Packets exchanged between nodes are encrypted		
5	Switch off all nodes.			

Table 4-30: Prototype 24 - Verification test #2

Verification Test Nr.: 4.3.6.2	**Written by:** TUC	**Conducted by:**	**Date:**	**Test Category:** Network Security
Software and Hardware Configuration Details	• 3 Z1 sensor nodes • Nodes running Contiki OS 2.6 and the IPsec protocol with AES in CCM* mode. • PC running GUI application monitoring communication using a Sniffing node • 1 node injecting malformed packets, bearing the destination node's address			
Test Name:	***Verification of message integrity among communicated messages***			
Purpose:	Verify that the network layer messages are received correctly and any possible tampering (REQ_NW02).			
Modules/Interfaces/Code Tested:	Network layer module IPsec module			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Compile all nodes comprising the scenario. Only node 1 sends data packets in order to keep track of the communication.	-		
2	Switch on all nodes.	Network layer control messages are exchanged among nodes. Sniffer overhears all messages exchanged among the nodes.		
4	Start transmitting application-layer messages from node 1.	MIC is added to the packet to provide integrity.		
5	Inject malformed packets intended for node 2.	Packets should be rejected by receiving node.		
6	Switch off all nodes.			

Table 4-31: Prototype 24 - Verification test #3

Verification Test Nr.: 4.3.6.3	**Written by:** TUC	**Conducted by:**	**Date:**	**Test Category:** Network Security
Software and Hardware Configuration Details	• 3 Z1 sensor nodes • Nodes running Contiki OS 2.6 and the IPsec protocol with AES in CCM* mode. • PC running GUI application monitoring communication using a Sniffing node • 1 node injecting malformed packets, bearing the destination node's address			
Test Name:	***Verification that an application can set the level of security provided***			
Purpose:	Verify that the application has the ability to choose the level of protection provided by the underlying network layer security protocol (REQ_NW19).			
Modules/Interfaces/Code Tested:	Network layer module IPsec module			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Compile all nodes comprising the scenario. Only node 1 sends data packets in order to keep track of the communication.	-		
2	Switch on all nodes.	Sniffer overhears all messages exchanged among the nodes.		
4	Observe exchanged messages	The application has the ability to choose among the different options of provided security, regarding the use of encryption and/or message authentication.		
5	Switch off all nodes.			

Table 4-32: Prototype 24 - Verification test #4

Verification Test Nr.: 4.3.6.4	**Written by:** TUC	**Conducted by:**	**Date:**		**Test Category:** Network Security
Software and Hardware Configuration Details	• 2 Z1 or MSP430-compatible sensor nodes • Nodes running Contiki OS 2.6 and the IPsec protocol with AES in CCM* mode. • PC running GUI application monitoring communication using a Sniffing node • 1 node injecting malformed packets, bearing the destination node's address				
Test Name:	***Verification that the mechanism introduces low delay***				
Purpose:	Verify that the mechanism introduces low delay compared to an unprotected communication (REQ_NW20).				
Modules/Interfaces/Code Tested:	Network layer module IPsec module				
Step	**Action**	**Expected Result**		**Pass/Fail**	**Remarks**
1	Compile all nodes comprising the scenario. Only node 1 sends data packets in order to keep track of the communication.	-			
2	Switch on all nodes.	Network layer control messages are exchanged among nodes. Sniffer overhears all messages exchanged among the nodes.			
4	Observe exchanged messages	Compare unprotected and protected messages for the various levels of provided security and measure introduced delay.			
5	Switch off all nodes.				

4.3.7 Anonymity & Location Privacy prototype (Prototype 10)

The fulfilment of the pertinent to the Anonymity & Location privacy prototype requirements will be accomplished via a validation procedure which is detailed in the next table.

Table 4-33: Prototype 10 - Validation procedure

Validation Procedure for Prototype 10	Test Description
Test Nr 4.3.7.1	Anonymity & Location Privacy (A&LP) service validation

Table 4-34: Prototype 10 - Validation test #1

Validation Test Nr.: 4.3.7.1	**Written by:** TUC	**Conducted by:**	**Date:**	**Test Category:** Service functionality validation
Software and Hardware Configuration Details	• 2 nodes setup as Anonymity & Location privacy proxies • 10 nodes setup as service users • 1 power node or PC running as Server GUI on Knopflerfish OSGi platform			
Test Name:	***Anonymity & Location Privacy (A&LP) service validation***			
Purpose:	Validate that the service works as intended, thus satisfying pertinent requirements.			
Modules/Interfaces/Code Tested:	User module Anonymity Node (user & proxy) module Server module (OSGi bundle)			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Setup 2 nodes as A&LP service proxies. Setup 10 nodes acting as service users. Setup a power node or PC with Knopflerfish OSGi platform running the Server bundle. Set K=5.	-		
2	Switch on all nodes.	Proxy nodes are initialized and calculate their cloaked area based on set “K” value. GUI featuring service area is visible on Server.		
3	Have a client issue a Location-based service request	Request is forwarded to nearest proxy node, is anonymized and forwarded (via inter-proxy routing) to server.		
4	Examine request arriving to server	The request should be anonymized, i.e. the server’s interface should display requester’s pseudonym and a cloaked area with at least “K” users as the origin of the request.		
5	Request is handled and returned to user	Server tends to the request and sends reply to nearest proxy node. Request is routed to originating user.		
6	Switch off all nodes.			

5 Middleware and Overlay Prototypes validation and verification

5.1 SHIELD Middleware and Overlay requirements

An initial middleware and overlay requirements assessment has been conducted in nSHIELD deliverable D2.2: Preliminary System Requirements and Specifications. In this section, the requirements are recited and their approach is updated through the -up to now- available input from the technical developments of nSHIELD Middleware and Overlay Layers design (WP5) and related work.

In the following table the requirements that are addressed by the prototypes developed for the SHIELD middleware and overlay are reported, with the indication of the mean of verification (A, D, I, T) and the applicability scope (validation or verification). Only requirements covered by test are extensively investigated.

Table 5-1: Requirements relevant against validation and verification of Middleware and Overlay layers prototypes

REQ_MW01 – Discovery The nSHIELD middleware shall offer discovery functionalities. In order to collect information on available SPD functionalities, the nSHIELD middleware must be equipped with discovery functionalities that will be an OSGI bundle. **Mean of Verification**: [R, D]
REQ_MW02 - Secure Discovery It is recommended that the nSHIELD middleware discovery is performed in a secure way. This requirement underlines that the nSHIELD middleware discovery is a security functionality itself, so it must be performed in a secure way (see Discovery bundle) **Mean of Validation**: [R, D, T]
REQ_MW03 Composition The nSHIELD middleware shall be able to compose nSHIELD components and functionalities. This requirement specifies that, once collected the necessary information, the nSHIELD middleware must be able to use them to compose nSHIELD components and functionalities. This functionality is a specific Bundle operated by the Security Agent **Mean of Verification**: [R, D]
REQ_MW04 Secure/Trusted Composition It is recommended that the composition of nSHIELD components and functionalities is performed in a secure/trusted way. With respect to components/functionalities composition, this requirement specifies that the composition must be performed in a secure/trusted way (see composition bundle). **Mean of Validation**: [R, D]
REQ_MW05 Orchestration and choreography The nSHIELD middleware shall be able to orchestrate the composition of nSHIELD components orchestration may be driven by defined policies, rules or control algorithms. The objective of the SHIELD platform is to compose SPD functionalities, so the nSHIELD middleware shall be able to orchestrate the composition of nSHIELD components. This orchestration may be driven by defined policies, rules or control algorithms. This orchestration is done by a dedicated OSGI bundle, part of the framework **Mean of Verification**: [R, D]

REQ_MW06 - Information retrieving

The nSHIELD middleware shall be able to retrieve information and requests from nSHIELD components.

It is supposed that the nSHIELD middleware, in order to communicate and orchestrate the system, is able to retrieve information and requests from nSHIELD components.

Mean of Verification: [R, D, I].

REQ_MW07 - Information filtering for intrusion detection

The nSHIELD middleware shall be able to filter information and requests from/to nSHIELD components to prevent malicious attacks.

Mean of Validation: [R, D, I, T]

REQ_MW08 – Enforcement

The nSHIELD middleware shall be able to enforce decisions taken by the overlay into SHIELD components.

Once computed the optimal solution of the composition problem, the nSHIELD dedicated bundle must be able to enforce decisions taken by the overlay into SHIELD components.

Mean of Verification: [R, D]

REQ_MW09 - Data management

The nSHIELD middleware shall be able to manage both service description and semantic data related to the discovered nSHIELD components and to the specific application domain.

The nSHIELD middleware is aware of the system capabilities thanks to its discovery functionalities. This requirement imposes that the middleware must be able to manage both service description and semantic data related to the discovered nSHIELD components and to the specific application domain.

Mean of Verification: [R, D]

REQ_MW10 – Interoperability

The nSHIELD middleware shall be able to provide a mechanism that allows legacy devices to be integrated into nSHIELD framework and make use of nSHIELD services. This is done by using specific software adapters (OSGi bundles) that contain the semantic information necessary for the discovery/composition procedure and that are able to communicate them.

Mean of Verification: [R,I,T]

REQ_MW11 - Non-repudiation of origin for secure service discovery, composition and delivery protocols

The middleware components for service discovery, composition and delivery protocols should provide a method to ensure that a subject that receives information during a data exchange is provided with evidence of the origin of the information. This evidence can then be verified by either this subject or other subjects

In order to guarantee a secure behaviour of the SHIELD middleware, it is imposed a non-repudiation functionalities between entities involved both in service discovery, composition and delivery. In particular this requirement addresses the origin of information exchange.

Mean of Validation: [R, D, I]

REQ_MW12 - Non-repudiation of receipt for secure service discovery, composition and delivery protocols

The middleware components for service discovery, composition and delivery protocols should provide a method to ensure that a subject that transmits information during a data exchange is provided with evidence of receipt of the information. This evidence can then be verified by either this subject or other subjects.

In order to guarantee a secure behaviour of the SHIELD middleware, it is imposed a non-repudiation functionalities between entities involved both in service discovery, composition and delivery. In particular this requirement addresses the receipt of information exchange.

Mean of Validation: [R, D, I]

REQ_MW13 - Access Control Policies for middleware components Middleware components shall have policies that identify sets of the following entities for access control functions: the subjects of access control, the objects of access control and the operations between the object and the subject covered by the policy. This requirement specifies the content of the policies developed to perform access control **Mean of Validation**: [R, D, I]
REQ_MW14 - Access Control Functions for middleware components A mechanism to perform Access Control, based on defined Access Control Policies, shall be implemented in the Middleware component(s). This requirement imposes the presence of a module performing access control on the bases of the developed Access Control Policies. **Mean of Validation**: [R, D]
REQ_MW15 - Configurations definition The nSHIELD overlay shall be able to elaborate feasible system configurations. The elaboration shall use, as input, service descriptions and semantic information retrieved by the middleware. This requirement says that elaboration must use, as input, service descriptions and semantic information retrieved by the middleware knowledge data bases. **Mean of Verification**: [R, D]
REQ_MW16 - Configurations quantification The nSHIELD overlay shall be able to measure the SPD level associated to each system configuration. Composing SPD functionalities towards an SPD objectives means that one is able to measure the current and the desired level of SPD within the system. This requirement imposes that the nSHIELD overlay must be able to measure the SPD level associated to each system configuration. **Mean of Verification**: [R, T]
REQ_MW17 - Configurations selection The nSHIELD overlay shall be able to choose, among the feasible configurations, the one that satisfies the SPD requirements in terms of SPD level and pre-defined policies The composition decision is based on two steps: first of all a feasible configuration must be collected. Secondly, the best feasible configuration must be selected. This requirement specifies this second point, i.e. the ability of choosing, among the feasible configurations, the one that satisfies the SPD requirements in terms of SPD level and pre-defined policies **Mean of Verification**: [D, R, T]
REQ_SH02 – Information transmission integrity Integrity of data transmitted over wireless or cable networks shall be granted. This requirement imposes the presence of mechanism that assures the integrity of the information exchanged in the SHIELD system. **Mean of Validation**: [A, T]
REQ_SH12 – SPD level assignment Each component or set of component in the system must have an assigned SPD level. As before, this requirement is focused on the needs of having a module and a methodology to quantify the SPD level of each SHIELD component **Mean of Validation**: [I]
REQ_SH16 - Data backup The system shall provide data backup for most sensitive data. Since data are at the basis of the nSHIELD behaviour and decision, it must be assured data backup at least for most sensitive data. **Mean of Validation**: [R, D]

REQ_SH17 - Data storage redundancy

Critical system data shall be secured through redundant storage

In particular, the availability of critical system data must be assured by the presence of redundant storage

Mean of Validation: [R, D]

REQ_SH18 - Data storage integrity

Safety and/or security critical data shall be integrity protected at storage

Data are also a critical asset, so safety and/or security of critical data must be assured through integrity protection at storage level

Mean of Validation: [A, D, T]

REQ_SH19 - Data storage confidentiality

Security critical data shall be confidentiality protected at storage.

As before, the security of critical data must be assured by confidentiality protection at storage level.

Mean of Validation: [R, D]

REQ_SH32 - Deployment manual

nSHIELD partners developing system components SHOULD provide description on how to deploy those system components, and how to operate them safely

This optional requirement aims at improving the usability of the SHIELD framework.

Mean of Verification: [I]

REQ_SH33 – Automated testing tools

Assigned nSHIELD project partner(s) SHOULD create automated testing tool(s) to generate arbitrary (even potentially faulty or malicious) input to exercise and evaluate input/output and internal interfaces of the critical components.

This requirement is focused on the definition of a potential mean of verification of the SHIELD behaviour and security.

Mean of Validation: [I]

REQ_SH34 - Guidance documents

nSHIELD partners developing system components SHALL provide user documentation on how to use the system securely and safely. The guidance shall include warnings about actions that can cause errors and lead to faults or failures in the secure environment.

This requirement is focused on the necessity of well-defined user documentation to improve SPD characteristic of a generic SHIELD system.

Mean of Validation: [I]

REQ_SH35 – Vulnerability assessment

Using results from test, validation, and verification tools, assigned nSHIELD project partner(s) SHALL carry out security analysis on the system components for all scenarios in order to find potential threats that could leave the nSHIELD system vulnerable.

This requirement is focused on the usage of a defined and standardized methodology to obtain a definition of the potential menaces to SHIELD system.

Mean of Validation: [A]

5.2 SHIELD Middleware and Overlay prototypes overview

In this section a brief overview of the prototypes is given, that shows, at high level, how these requirements have been addressed in the design and development activities.

5.2.1 SHIELD Semantic Model (Prototype 26)

The SHIELD Semantic Mode is based on a separation paradigm: on one hand a set of scenario dependent DBs contain all the information necessary to tailor the system aspects to the specific application. On the other hand, the abstract model of the generic SPD functionality, derived from the "attack surface" logic, contains the quantification of the SPD capabilities of the component as well as the mapping between menaces and means of mitigation (the basic principles of security). This concept is depicted in Figure 5-1 and detailed in D5.2 and D5.3.

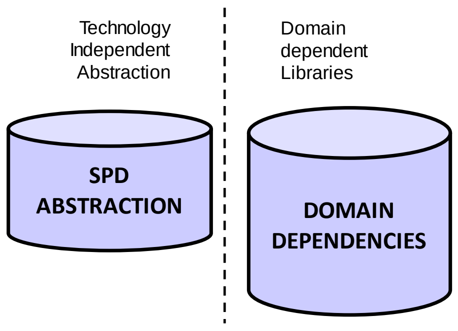
Technology Independent Abstraction
Domain dependent Libraries
SPD ABSTRACTION
DOMAIN DEPENDENCIES

Figure 5-1: nSHIELD Knowledge Bases

All the requirements related to data integrity and management are addressed by an adequate data storage mechanism (relational DB rather than global variable in system memory) with basic security functionalities implemented by the software environment.

5.2.2 SHIELD Secure Discovery (Prototype 32)

This prototype enable the SHIELD Middleware with the possibility of discovering the available SPD functionalities and services over heterogeneous environment, networks and technologies that are achievable by the nSHIELD Embedded System Device (nS-ESD) where it is running. Indeed the nSHIELD secure service discovery uses a variety of discovery protocols (such as SLP, SSDP, NDP, DNS, SDP, UDDI) to harvest over the interconnected Embedded System Devices (ESDs) all the available SPD services, functionalities, resources and information that can be composed to improve the SPD level of the whole system.

In order to properly work, a discovery process must tackle also a secure and dependable service registration, service description and service filtering. The service registration consists in advertising in a secure and trusted manner the available SPD services (thus satisfying the non-repudiation requirements). The advertisement of each service is represented by its formal description and it is known in literature as service description. The registered services are discovered whenever their description matches with the query associated to the discovery process, the matching process is also known in literature as service filtering.

This prototype is further detailed in D5.2 and D5.3

5.2.3 SHIELD Security Agent (Prototype 33)

The requirement focused on SPD composition and orchestration are mainly addressed by the Orchestrator and the Security Agent prototypes, two OSGI Bundles in charge of managing discovery, putting together i) the semantic models, ii) the knowledge bases, iii) the registered services and iv) the user needs and to elaborate a system configuration with a given (measurable) SPD level.

These different functionalities are reflected in the design of the Security Agent, whose detailed description is available in D5.2 and D5.3.

5.2.4 SHIELD Control Algorithms (Prototype 20)

The security agent is only a software routine and need a set of “algorithms” to take intelligent decision. In the scope of the SHIELD framework, several theoretical approaches have been investigated to better model the SPD composition. The most expressive candidate for the SHIELD purposes appeared to be the Colored Petri Nets, by which a model of system composition has been derived taking into account logical conditions that enables the system evolution through a set of controlled transitions. This prototype, in the form of theoretical model and simulation runs, shows that it is possible to drive the composability in an automatic way by leveraging the abstraction achieved with the derivation of semantic models.

The control is strongly based on the quantification of SPD (both for the component and for the overall system). This has been achieved through the implementation of the “attack surface” metrics, a deterministic approach (simple but effective) whose baseline is available in literature and that has been adapted to the project purposes with a cross fertilization with the Common Criteria.

These algorithms are described in detail in D5.2 and D5.3, while the metric approach is available in D2.5

5.2.5 SHIELD Middleware Intrusion Detection System (Prototype 22)

The Middleware Intrusion Detection System prototype is implemented to act as an input filter for other Core SPD Services with public interfaces, e.g. SHIELD secure discovery. The current module implements firewall functionality between other SHIELD components and the relevant middleware service interfaces.

The Middleware Intrusion Detection System provides filtering of messages received from other legitimate (SHIELD) or malicious components forwarding them towards the other middleware functionalities, and sending related responses back.

The following requirements formed core design rationale for the applied technical solution:

- **REQ_MW07 Information filtering for intrusion detection** - The nSHIELD middleware shall be able to filter information and requests from/to nSHIELD components to prevent malicious attacks.
- **REQ_MW10 Interoperability** - The nSHIELD middleware shall be able to interface with heterogeneous legacy component.

Document “Preliminary SPD middleware and Overlay technologies prototype Report” [6] contains design documentation as well as interface specification for the resulting Intrusion Detection and Filtering module, while document “Preliminary SPD middleware and overlay technologies prototype” [5] contains the preliminary version of implementation, as well as the developed test cases referred in 0.

5.2.6 SHIELD Policy-based Access Control (PBAC) & Policy-based Management (PBM) (Prototype 19)

The Policy-based Access Control framework implements access control based on policy restrictions imposed by the system owner, hence controlling data accessibility and authorization. The solution adopted for secure policy-based access control is based on eXtensible Access control Markup Language (XACML) policies. It consists of several components that run on different nodes of the nSHIELD architecture; these components are the Policy Enforcement Points (PEP), the Policy Decision Points (PDP), the Policy Administration Point (PAP) and, optionally, the Policy Information Point (PIP). A node, depending on its capabilities and the available resources, might include one or more of these functional components 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. The proposed framework is 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 following requirements formed core design rationale for the applied technical solution:

- **REQ_MW13 Access Control Policies for middleware components** - Middleware components shall have policies that identify sets of the following entities for access control functions: the subjects of access control, the objects of access control and the operations between the object and the subject covered by the policy.

- **REQ_MW14 Access Control Functions for middleware components** - A mechanism to perform Access Control, based on defined Access Control Policies, shall be implemented in the Middleware component(s).

5.2.7 SHIELD Middleware Protection Profile (Prototype 31)

In order to address security, privacy and dependability (SPD) open issues, the nSHIELD project aims at addressing SPD in the context of ESs as "built in" functionalities, proposing and perceiving with this strategy the first step towards SPD certification for future ESs.

In this direction, a first step to define a security problem definition and security objectives for embedded systems could be represented by editing a particular document called “Protection Profile” that in this acceptation can be considered as a prototype for the nSHIELD project.

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.

In this case PP is defined by a research group (WP5 partners) to specify IT security needs of a generic Embedded System middleware to be nSHIELD compliant.

5.2.8 Adaptation of Legacy Systems (Prototype 29)

The nSHIELD architecture should be also generic enough in order to allow participation of legacy embedded systems not capable to support the nSHIELD SPD modules. Therefore nSHIELD can be regarded as a network consisting of nSHIELD and Legacy embedded devices. The L-ESDs since they do not understand nSHIELD middleware services they need a gateway nSHIELD device in order to participate in the nSHIELD system.

In order to allow legacy devices to make use of nSHIELD services, specific adapters (ad-hoc software) are provided. The ad-hoc software is OSGi bundles in the nSHIELD side and in L-ESD side. The software in L-ESD side provides discovering of the nSHIELD remote services and the software in nSHIELD side provides advertising nSHIELD services for being remotely discovered.

These ad-hoc routines are further described in detail in D5.2 and D5.3.

5.3 SHIELD Middleware and Overlay prototypes verification and validation

In this section a description of the verification and validation activities is reported, with particular attention to those prototypes whose associated requirements are marked with a T (i.e. a test is needed or verifies that the requirement is met).

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

5.3.1 SHIELD semantic model (Prototype 26)

In this section the validation and verification means of the Semantic Models prototype are detailed.

Table 5-2: Prototype 26 - Semantic Models Verification cases

ID	nSHIELD Requirement	Mean of verification	Description
T.1	**REQ_MW06 Information retrieving**	R, D, I	[R, D] The possibility of retrieving information is verified by review of the architectural documents (WP2) showing the presence of data bases and modules to collect data. [I] These components are also visible in the software code.
T.2	**REQ_MW09 Data management**	R, D	[R, D] As above, it is reflected in the architectural design documents the presence of module performing data management
T.3	**REQ_SH16 Data backup**	R, D	[R, D] This requirement is validate with the presence of backup/redundancy in the definition of SHIELD data repository
T.4	**REQ_SH17 Data storage redundancy**	R, D	
T.5	**REQ_SH18 Data storage integrity**	R, D	[R, D] The solution adopted to store information must include a function that performs integrity check on the stored data. This function is delegated to the environment (i.e. operating system)
T.6	**REQ_SH19 Data storage confidentiality**	R, D	[R, D] The solution adopted to store information includes a function that assures the confidentiality of the stored data (ciphered repository).

5.3.2 SHIELD secure discovery (Prototype 32)

In this section the validation and verification means of the secure discovery prototype are detailed.

Table 5-3: Prototype 32 - Secure Discovery Verification cases

ID	nSHIELD Requirement	Mean of verification	Description
T.7	**REQ_MW01 Discovery**	R, D	[R, D] The architectural design document includes a bundle that is in charge of actuating service discovery. No specific technology is requested, so a multi-technology bundle is foreseen.
T.8	**REQ_MW02 Secure Discovery**	R, D, T	[R, D] The solution adopted to perform discovery, includes security features (e.g. authentication or cyphering) in order to assure the security of the mechanism. [T] Test 6.3.2.1 verifies that the discovery is performed in a secure way
T.9	**REQ_MW11 Non-repudiation of origin for secure service discovery, composition and delivery protocols**	R, D, I	[R, D, I] The design of the middleware framework includes specific modules computing signature to mark all the requests and consequently avoid repudiation
T.10	**REQ_MW12 Non-repudiation of receipt for secure service discovery, composition and delivery protocols**	R, D, I	

Table 5-4: Prototype 32 - Verification case #1

Verification Test Nr.: 5.3.2.1	**Written by:** UNIROMA1	**Conducted by:**	**Date:**	**Test Category:** Middleware Security
Software and Hardware Configuration Details	• SLP Secure Directory Agent should be running • SLP Secure Service Agent should be installed on the OSGi			
Test Name:	***Validation of the Secure Service Discovery: secure service registration***			
Purpose:	Verify that the service discovery registration is performed in a secure way			
Modules/Interfaces/Code Tested:	SLP Secure Directory Agent (SLP Daemon Bundle) SLP Secure Service Agent (Pluggable Discovery Module SLP)			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Install the SLP Secure Directory Agent on the OSGi (SLP Daemon Bundle)	Installation should be ok, with no errors.		
2	Install the Service Discovery APIs on the OSGi (Pluggable Discovery Module API)	Installation should be ok, with no errors.		
3	Install the SLP Secure Service Agent on the OSGi (Pluggable Discovery Module SLP)	Installation should be ok, with no errors.		
4	Run the SLP Secure Directory Agent	The SLP Secure Directory Agent should start, with its GUI appearing as a proper window		
5	Run the SLP Secure Service Agent	In case, debug message stating the SLP Secure Service Agent was started		
6	Start registration of services through the SLP Secure Service Agent	The SLP Secure Service Agent should start sending SLP messages to the SLP Secure Directory Agent. The list of the registered services should appear in few moments on the SLP Secure Directory Agent		

Table 5-5: Prototype 32 - Verification case #2

Verification Test Nr.: 5.3.2.2		**Written by:** UNIROMA1	**Conducted by:**	**Date:**	**Test Category:** Middleware Security
Software and Hardware Configuration Details		• SLP Secure Directory Agent should be running • SLP Secure User Agent should be installed on the OSGi			
Test Name:		***Validation of the Secure Service Discovery: secure service discovery***			
Purpose:		Verify that the service discovery is performed in a secure way			
Modules/Interfaces/Code Tested:		SLP Secure Directory Agent (SLP Daemon Bundle) SLP Secure User Agent (Pluggable Discovery Module SLP)			
Step	**Action**	**Expected Result**		**Pass/Fail**	**Remarks**
1	Install the SLP Secure Directory Agent on the OSGi (SLP Daemon Bundle)	Installation should be ok, with no errors.			
2	Install the Service Discovery APIs on the OSGi (Pluggable Discovery Module API)	Installation should be ok, with no errors.			
3	Install the SLP Secure User Agent on the OSGi (Pluggable Discovery Module SLP)	Installation should be ok, with no errors.			
4	Run the SLP Secure Directory Agent	The SLP Secure Directory Agent should start, with its GUI appearing as a proper window			
5	Run the SLP Secure User Agent	In case, debug message stating the SLP Secure User Agent was started			
6	Start discovery of services through the SLP Secure User Agent	The SLP Secure User Agent should start sending SLP messages to the SLP Secure Directory Agent. The list of the registered services should be returned by the SLP Secure Directory Agent to the SLP Secure User Agent and appear in few moments on this last one			

5.3.3 SHIELD Security Agent (Prototype 33)

In this section the validation and verification means of the Security Agent prototype are detailed.

Table 5-6: Prototype 33 - Secure Agent Verification cases

ID	nSHIELD Requirement	Mean of verification	Description
T.11	**REQ_MW05 Orchestration and choreography**	R, D	[R, D] The requirement is verified because the orchestration/choreography bundle is included in the architectural design of the SHIELD middleware
T.12	**REQ_MW03 Composition**	R, D	[R, D] The requirement is verified because the composition bundle is included in the architectural design of the SHIELD middleware
T.13	**REQ_MW04 Secure/Trusted Composition**	R, D	[R, D] The requirement is verified because the composition is equipped with mechanisms or algorithms that enable a secure/trusted composition
T.14	**REQ_MW08 Enforcement**	R, D	[R, D] The requirement is verified because the enforcement bundle is included in the architectural design of the SHIELD middleware

5.3.4 SHIELD Control Algorithms (Prototype 20)

In this section the validation and verification means of the Control Algorithms prototype are detailed.

Table 5-7: Prototype 20 - Control Algorithms Verification cases

ID	nSHIELD Requirement	Mean of verification	Description
T.11	**REQ_MW15 Configurations definition**	R, D	[R, D] The requirement is validated because the security Agent contains a module that, by using the available data sources, is able to extrapolate a configuration (domain data bases + abstraction data base)
T.12	**REQ_MW16 Configurations quantification**	R, D	[R, D] The metric approach is translated into a set of deterministic/logical steps to quantify a configuration. This is done by the Security Agent
T.13	**REQ_MW17 Configurations selection**	R, D, T	[R, D] In the architectural definition of the SHIELD middleware the decision making engine has been inserted to assure the elaboration of a solution of the composability problem. [T] 6.3.4.1 verifies that at least one configuration is selected

Table 5-8: Prototype 20 - Verification case #1

Verification Test Nr.: 5.3.4.1		**Written by:** UNIROMA1	**Conducted by:**	**Date:**	**Test Category:** Middleware Security
Software and Hardware Configuration Details		• PC running CPNtools program			
Test Name:		***Validation of the Secure Service Discovery***			
Purpose:		Verify that the service discovery is performed in a secure way			
Modules/Interfaces/Code Tested:		OSGI Service Discovery Bundle			
Step	**Action**		**Expected Result**	**Pass/Fail**	**Remarks**
1	Open CPNtools and load the appropriate file				
2	Open each SPD functionality module and add the appropriate tokens (representing the feasible implementations of each functionality) in each place *feasible*		Place feasible initial marking represents the list of possible implementation of each functionality.		
3	Add a token representing the desired SPD value to place *desired* in the main page, and run the simulation.		If there is a least one solution (for each functionality, there is at least one available implementation that respects the SPD desired value) the system reaches the optimal configuration, hence each place *FUNC impl* contains a token that satisfy the desired SPD value. The configuration is determined by the set of implemented functionalities and the optimal one is obtained enabling, for each functionality, the implementation with the smallest SPD value that satisfies the desired value.		
4	Open a functionality sub-module (for example Authentication) and add appropriate tokens (representing new available implementations) in the place *new_fnc*.		The new implementation is added to place *feasible*. Then, if enabling the new implementation the system achieves a better configuration, the system reaches this new configuration; else no configuration change is performed.		
5	Open a functionality sub-module (for example identification) and add a true boolean token in the place *off*. This means that the implemented functionality will become unavailable.		The system actives a new implementation to achieve the desired level of SPD, obviously if at least one is available.		

6	Add a new token representing a new desired SPD value to place *desired* in the main page.	The system achieves the new optimal configuration.		
7	Open a relation sub-module (for example Coupling) and add appropriate tokens (representing the implementations that needs specific are in relation with each other) in each place of specific functionality.	Functionality places in each relation sub-page have a marking representing the list of functionality implementation that is in relation each other. The system checks if the implemented functionalities have a coupling constraint to satisfy. In particular, the control action addresses any coupling constraint driving the system by means tokens only if the implementations required by the coupling are available and satisfies the required SPD level. On the other hand, if this condition is not verified then the coupling block “turns off” the implementations that required the coupling. Thus the system actives a new implementation to achieve the desired level of SPD, obviously if at least one is available.		

5.3.5 SHIELD Middleware Intrusion Detection System (Prototype 22)

Validation and Verification for the SHIELD Middleware Intrusion Detection System consists of the following activities:

1 Verification Procedure

- Code Review (A)
  1) Against relevant requirements (REQ_SH02)
- Testing
  1) Check against malformed input manually (REQ_MW07)

2 Validation Procedure

- Code Review (REQ_SH02)
  1) Taint analysis,
  2) Interfaces validation
- Automated testing scenario to validate the intrusion detection functionality
  1) Automated testing tools developed (fulfilling REQ_SH33)
     - Basic functionality test (SendReceiveTest)
     - load generation (CriticalLoadTest)
     - Information filtering tests (BlackListTest and WhiteListTest)
  2) Validation of functionality against requirements (REQ_SH02, REQ_MW07, MW17)

Relevant test cases were developed as test units part of the development, and submitted in source code form in [5]. Test results verified that related requirements (REQ_SH02 Information transmission integrity, REQ_MW7 Information filtering for intrusion detection, REQ_MW17 Configurations selection) were satisfied.

5.3.6 SHIELD Policy-based Access Control (PBAC) & Policy Based Management (PBM) (Prototype 19)

In this section the validation and verification means of the Policy-based Access Control and Policy-base Management prototypes are detailed. The functional requirements covered by the framework are verified as follows:

Table 5-9: Prototype 19 - Policy Based Management Framework Verification cases

ID	nSHIELD Requirement	Mean of verification	Description
T.14	**REQ_SH06 Authorisation**	D,T	Verification that the only authorized entities can have access to nSHIELD nodes' resources and services.
T.15	**REQ_MW01 Discovery**	D,T	Verification that DPWS discovery functionality operates properly on all entities.
T.16	**REQ_MW02 Secure Discovery**	D	Secure discovery is optional and facilitated by DPWS but only pertinent for power nodes due to the imposed performance overhead.
T.17	**REQ_MW06 Information Retrieving**	D, T	Verification that information retrieving and request functionality operates as designed.
T.18	**REQ_MW08 Enforcement**	D	PBAC framework deals, by design, with the enforcement of security policies. Validation of the whole framework will verify that the requirement is covered.

T.19	**REQ_MW09 Data Management**	A, D	Prototype analysis to verify that DPWS device metadata is defined and exchanged as intended.
T.20	**REQ_MW11 Non-repudiation of origin for secure service discovery, composition and delivery protocols**	D	Verify design data-origin authentication is integrated into the design.
T.21	**REQ_MW12 Non-repudiation of receipt for secure service discovery, composition and delivery protocols**	D	Verify that appropriate mechanisms are integrated into the design.
T.22	**REQ_MW13 Access Control Policies for middleware components**	R, D, I	REQ_MW13 is core design requirement and formed the rationale for the development of the PBAC framework. Review documentation and design choices to verify that the requirement is covered. Inspect elements to verify that the required policies are present on pertinent entities (e.g. Policy Administration Point).
T.23	**REQ_MW14 Access Control Functions for middleware components**	R, D, T	REQ_MW14 is core design requirement and formed the rationale for the development of the PBAC framework. Review documentation and design choices to verify that the requirement is covered. Validate Framework to certify core requirements are covered.

Table 5-10: Prototype 19 - Verification case #1

Verification Test Nr.: 5.3.6.T14		**Written by:** TUC	**Conducted by:**	**Date:**	**Test Category:** Middleware Security
Software and Hardware Configuration Details		• 1 nano node setup as a DPWS device with PEP functionality • 1 micro node setup as a DPWS device with PEP functionality • 1 PC running DPWS client software			
Test Name:		***Authorisation Verification***			
Purpose:		Verify that access to node resource is limited to authorized entities only.			
Modules/Interfaces/Code Tested:		Nano node DPWS PEP module Micro node DPWS PEP module			
Step	**Action**	**Expected Result**		**Pass/Fail**	**Remarks**
1	Setup a nano and a micro node as DPWS devices featuring Policy Enforcement Points (PEPs). Setup PC as DPWS client.				
2	Switch on all nodes and start applications.				
3	Use unauthorised DPWS client application on PC.	Client application discovers nano and micro nodes.			
4	Attempt to access micro node resources via client	DPWS Client fails to present required credentials and, thus, cannot access micro node services.			
5	Attempt to access nano node resources via client.	DPWS Client fails to present required credentials and, thus, cannot access nano node services.			
6	Switch off all nodes.				

Table 5-11: Prototype 19 - Verification case #2

Verification Test Nr.: 5.3.6.T15	**Written by:** TUC	**Conducted by:**	**Date:**	**Test Category:** Middleware Communications
Software and Hardware Configuration Details	• 1 nano node setup as a DPWS device with PEP functionality • 1 micro node setup as a DPWS device with PEP functionality • 1 power node running PDP bundle on Knopflerfish OSGi platform • 1 power node running PAP/PIP bundle on Knopflerfish OSGi platform • 1 PC running DPWS client software			
Test Name:	***Discovery Verification***			
Purpose:	Verify that DPWS discovery functionality operates properly on all types of PBAC entities.			
Modules/Interfaces/Code Tested:	Nano node DPWS discovery functionality Micro node DPWS discovery functionality Power node PDP discovery functionality Power node PAP/PIP discovery functionality			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Setup a nano and a micro node as DPWS devices featuring Policy Enforcement Points (PEPs). Setup a power node with Knopflerfish OSGi platform running the PDP bundle. Setup a power node with Knopflerfish OSGi platform running the PAP/PIP bundle. Setup PC as DPWS client.			
2	Switch on all nodes and start applications.	PDP discovers nano DPWS devices featuring PEPs. PDP discovers micro DPWS devices featuring PEPs. PDP discovers power DPWS devices PAP/PIP.		
3	Use DPWS client application on PC.	Client discovers nano and micro nodes, their hosted services and respective operations.		
4	Switch off all nodes.			

Table 5-12: Prototype 19 - Verification case #3

Verification Test Nr.: 5.3.6.T17	**Written by:** TUC	**Conducted by:**	**Date:**	**Test Category:** Middleware Functionality & Communications
Software and Hardware Configuration Details	• 1 nano node setup as a DPWS device with PEP functionality • 1 micro node setup as a DPWS device with PEP functionality • 1 power node running PDP bundle on Knopflerfish OSGi platform • 1 power node running PAP/PIP bundle on Knopflerfish OSGi platform • 1 PC running DPWS client software			
Test Name:	***Information Retrieving Verification***			
Purpose:	Verify that middleware is able to retrieve information and requests from nSHIELD components.			
Modules/Interfaces/Code Tested:	Nano node DPWS device Micro node DPWS device Power node PDP OSGi bundle Power node PAP/PIP OSGi bundle			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Setup a nano and a micro node as DPWS devices featuring Policy Enforcement Points (PEPs). Setup a power node with Knopflerfish OSGi platform running the PDP bundle. Setup a power node with Knopflerfish OSGi platform running the PAP/PIP bundle. Setup PC as DPWS client with an "Allow" policy for said client in the PAP.			
2	Switch on all nodes and start applications.			
3	Use DPWS client application on PC to access nano node resources.	Access attempt is intercepted by the node's PEP. PEP requests decision from PDP. PDP retrieves policy information from PAP/PIP and communicates "Allow" decision to PEP. Client accesses nano node service and retrieves a value (e.g. a current temperature value).		
4	Use DPWS client application on PC to access micro node resources.	Access attempt is intercepted by the node's PEP. PEP requests decision from PDP. PDP retrieves policy information from PAP/PIP and communicates "Allow" decision to PEP. Client accesses micro node service and retrieves a value (e.g. a current temperature value).		
5	Switch off all nodes.			

Then the whole framework is validated to certify that its core functionalities are covered as well. The validation procedure is detailed below:

Table 5-13: Prototype 19 - Validation test #1

Validation Test Nr.: 5.3.6.T23	**Written by:** TUC	**Conducted by:**	**Date:**	**Test Category:** Middleware Validation
Software and Hardware Configuration Details	• 1 nano node setup as a DPWS device with PEP functionality • 1 micro node setup as a DPWS device with PEP functionality • 1 power node running PDP bundle on Knopflerfish OSGi platform • 1 power node running PAP/PIP bundle on Knopflerfish OSGi platform • 1 PC running DPWS client software and sniffer			
Test Name:	***Policy-based Access Control Validation***			
Purpose:	Verify that access to nodes' resources is controlled by XACML policies.			
Modules/Interfaces/Code Tested:	Nano node DPWS device code & PEP module Micro node DPWS device code & PEP module Power node PDP OSGi bundle Power node PAP/PIP OSGi bundle			

Step	**Action**	**Expected Result**	**Pass/Fail**	**Remarks**
1	Setup a nano and a micro node as DPWS devices featuring Policy Enforcement Points (PEPs). Setup a power node with Knopflerfish OSGi platform running the PDP bundle. Setup a power node with Knopflerfish OSGi platform running the PAP/PIP bundle. Setup PC as DPWS client.			
2	Switch on all nodes and start applications.	PDP discovers DPWS devices featuring PEPs. Communication is established between PEPs and PDP. Communication is established between PDP and PEP/PIP.		
3	Use DPWS client application on PC.	Client application discovers nano and micro nodes, their hosted services and respective operations.		
4	Attempt to access micro node resources via client	Access attempt is intercepted by the node's PEP, which communicates with PDP to request decision. PDP consults with PAP/PIP in a secure manner (as verified by network sniffer) and issues "Deny" decision to PEP. Client's access to node's resources is refused.		
5	Introduce a new XACML policy on the PAP/PIP node, explicitly allowing access of client to test node.	Repeating Step 4 now returns an "Allow" response from the PDP and the client successfully invokes the service operation, accessing the test node's resources.		
6	Repeat the test, replacing micro node with nano node.	XACML policies are effectively enforced on nano nodes as well.		
7	Switch off all nodes.			

5.3.7 SHIELD Middleware Protection Profile (Prototype 31)

A Protection Profile is a document type well defined in a particular standard (Common Criteria) and so it has appropriate Validation and Verification processes. They can be considered in the "Analysis" macro area identified in section 2 of this document and consist of the following activities:

- Verification Procedure: it is the process that a user community, TOE developers or large corporations execute when define the Protection Profile according to the rules defined in the Annex B of Common Criteria documentation Part 1 – Specification of Protection Profiles [2]. The mandatory content for a PP is outlined in the following diagram

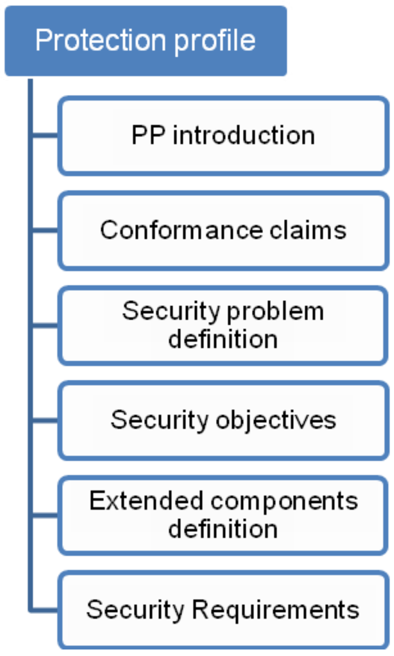

Figure 5-2: Protection Profile contents

The separate sections of a PP and the contents of those sections are briefly summarised below.

- a PP *introduction* containing a narrative description of the TOE type;
- a conformance claim, showing whether the PP claims conformance to any PPs and/or packages, and if so, to which PPs and/or packages;
- a security problem definition, showing threats, OSPs and assumptions;
- security objectives, showing how the solution to the security problem is divided between security objectives for the TOE and security objectives for the operational environment of the TOE;
- Extended components definition, where new components (i.e. those not included in CC Part 2 [3] or CC Part 3[4]) may be defined. These new components are needed to define extended functional and extended assurance requirements;
- Security requirements, where a translation of the security objectives for the TOE into a standardised language is provided. This standardized language is in the form of SFRs. Additionally this section defines the SARs;

- Validation Procedure: it is a process identified with "PP Evaluation". Evaluating a PP is required to demonstrate that the PP is sound and internally consistent, and, if the PP is based on one or more other PPs or on packages, that the PP is a correct instantiation of these PPs and packages. These properties are necessary for the PP to be suitable for use as the basis for writing an ST or another PP. The actions that an Evaluator executes during a PP evaluation process are defined in the common criteria Class APE contained in section 10 of Common Criteria documentation Part 3 [4]. The components defined for the APE class are summarized in the following table.

Table 5-14: Prototype 31 - PP evaluation components

Assurance Class	Assurance family	Assurance component
Protection Profile Evaluation	APE_CCL	1
	APE-ECD	1
	APE_INT	1
	APE_OBJ	2
	APE_REQ	2
	APE_SPD	1

5.3.8 Adaptation of Legacy Systems (Prototype 29)

Validation and Verification for the SHIELD Adaptation of Legacy Systems consists of the following activities:

- Verification:
  1. Code review
  2. Check against relevant requirements (REQ_MW06,REQ_SH01,REQ_MW18)
  3. Check for network connectivity
- Validation: To validate the implementation of adaptation of Legacy Systems into the nSHIELD framework we created a very simple service Nservice (Echo Service) that runs in server side and registers itself to R-OSGi.On the other hand the client side runs a LeNoReSer (Legacy Node Service) that connects remotely to the nSHIELD server and gets the Remote nSHIELD service (Echo Service).

  As a result of the above a local proxy for the remote service is created. The service proxy is registered with the local service registry and can also be retrieved like a normal OSGi service.

  In our test the Echo service is running on a machine and the ad hoc software bundle registers it in R-OSGi.When the ad hoc LeNoReSer runs on another machine on the network gets the Echo service from the remote machine and displays a message.

  So we validate that the scenario of using nSHIELD services by Legacy Systems is applicable.

6 Conclusions

In this deliverable we have demonstrated that nSHIELD prototypes are independent development efforts able to verify and validate that the network requirements defined in WP2 have been met. These prototypes work as a proof of concept that a generic nSHIELD compliant system behaves accordingly to specified rules providing a specific SPD functionality.

In WP7 these prototypes will be considered as the input sources to the nSHIELD scenario demonstrators and whose final behaviour will be validated.

7 References

[1] nSHIELD, D2.2: Preliminary Systems Requirements and Specifications

[2] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model; CCMB-2012-09-001 - Version 3.1 Revision 4, September 2012

[3] Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components; CCMB-2012-09-002 - Version 3.1 Revision 4, September 2012

[4] Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components; CCMB-2012-09-003 - Version 3.1 Revision 4, September 2012

[5] nSHIELD, D5.2: Preliminary SPD middleware and overlay technologies prototype

[6] nSHIELD, D5.3: Preliminary SPD middleware & Overlay technologies prototype Report