



M4.1.600 Cybersecurity Reference Architecture Model and Building Block Repository

Status: final
Version: 2.0
Date: 28 March 2025

Legal disclaimer

The sole responsibility for the content of this document lies with the authors. It does not necessarily reflect the opinion of the European Union. The European Commission is not responsible for any use

that may be made of the information contained therein. All images are provided by the respective partners (unless otherwise noted) and are approved for reproduction in this publication.



Document information

Project acronym	NAPCORE
Full project title	National Access Point Coordination Organisation for Europe
Grant Agreement No.	MOVE/B4/SUB/2020-123/SI2.852232
Activity no. and title	[include no. and name of Activity]
Author(s)	Massimiliano Masi, Autostrade Per L'Italia SpA, Paal Aaserud (Vegvesen)
Co-author(s)	Andre Rehberg, Armis; Martin van Ekelburg, NDW ; Jan Vlcinsky Tam Tam Research; Josef Kaltwasser Albrecht Consult; Jean-Philippe Mechin, CEREMA; Paoletti Fabrizio, Autostrade Per L'Italia SpA; Sorin Dumitrescu, Elsol; Bard de Vries, NDW, Paal Aaserud NPRA
Related to Milestone no.	[include number]
External Milestone	[yes or no]

Document history

Version	Date	created/ modified by	Comments
1.0	28.03.2025	Massimiliano Masi Autostrade Per L'Italia	Final version of ABB and SBB

Action requested

- ☐ To be revised by partners involved in the preparation of the document
- ☐ For review/ approval by the Core Alignment Team
- ☐ For approval by the NAPCORE Steering Committee

Abstract

The actors, data, and the technical aspects involved in a DATEX II exchange have similarities in other sectors. A Datex II project typically involves the exchange of road conditions and traffic-related information as well as information about traffic management decision to be agreed and cooperatively acted among a number of stakeholders which may have different cybersecurity posture who commit to solve interoperability problems using a standardized information sharing. Such stakeholders are typically Motorway Operators, Smart Cities, Service Providers and OEM. The Cybersecurity working group identified similarities in additional three sectors: healthcare, electronic delivery, and International Data Space Association (IDSA). This document is composed as follows: the subsequent sections will analyse the Reference Architecture for security for the three sectors mentioned above, and in the last section, we will perform a discussion on how DATEX II can learn from the experiences, and we conclude.

The text highlighted in **yellow** would specify a similarity to DATEX II.



Abbreviations

Abbreviation	Meaning
EC	European Commission
CAT	Core Alignment Team
NAP	National Access Point
NAPCORE	National Access Point Coordination Organisation for Europe
SC, SCOM	Steering Committee
SCS	Steering Committee Support
SWG	Sub-working Group
WG	Working Group
WP	Working Programme

Table of contents

No table of contents entries found.

List of figures

Figure 1 A graphical representation of Archimate CORE layers and its structures.....	9
Figure 2 The architectural metamodel.....	13
Figure 3 Building block landscape	14
Figure 4 Example Business Process View	15
Figure 5 Example of Application View	16
Figure 6 Example of Technology View.....	17
Figure 7 The IDSA RAM.....	18
Figure 8 NODEABB model.....	21
Figure 9 AUTHZABB Archimate Model	26
Figure 10 Archimate model for Data Integrity	31
Figure 11 The W3C Definition of PROV	36
Figure 12 DPABB Archimate Model.....	37
Figure 13 Archimate model for the DEABB	41
Figure 14 Non-Repudiation ABB: archimate model	48
Figure 15 The TLP protocol.....	58
Figure 16 Picture from J. Somorowski et al, All your clouds belong to us – Security Analysis of Cloud Management Interfaces, In Proceeding of the ACM Cloud Computing Security Workshop (CCSW), 2011.....	87



Introduction

Cybersecurity is nowadays usually referenced as a *Cross-Cutting Concern*, explicitly to describe that cybersecurity is a *process* supported by specific technical countermeasures that span over different abstraction layers, from the technical to the business. Moreover, the concept known as *defence in depth*, prescribes to architect countermeasures to provide security even if one of the countermeasures fail. However, choosing and defining such processes and the related technical countermeasures, is an error-prone activity that is often left to the talent of the Cybersecurity practitioners, which is per se a scarce resource¹. This fact hinders the creation of systems which are secure by design since the way how cybersecurity countermeasures are implemented are not rigorous nor repeatable. Notably, countermeasures are put in place to mitigate risks, as also highlighted by the recent EU 2022/2555, where a risk management practice has to be established by the owner of the services, being driven by e.g., Business Impact Analyses (BIA), or Data Protection Impact Assessments (DPIA). In particular, EU 2022/2555 defines the concept of systems with “high criticality”, for which cybersecurity countermeasures have to be in place to mitigate risks. The NAP is a critical infrastructure, according with Annex 1, section 2, (d). For this reason, this document prescribes building blocks that are defined to mitigate specific risks, and do not prescribe a checklist of requirements. However, a Reference Architecture is presented, to guide new implementations, or to create eventual gap analyses with existing implementation. In fact, the aim of all the countermeasures described in this document is to attain interoperability in cybersecurity implementations.

Being a Cross-Cutting Concern, cybersecurity countermeasures are defined at various levels by international standards. In this document, the following will be used.

- ISO 27001:2022 and ISO 27002:2022, defining the governance of the Cybersecurity (ISO 27001) with technical description on how to implement the requirements (ISO 27002) identified.
- MITRE ATT&CK: a repository of techniques and tactics historically used in real world observations, on events pursued by threat actors.

Datex II, the model-driven methodology behind the solutions developed within the scope of NAPCORE, allows the creation of systems for the sharing of information about mobility. At high level, the methodology is divided in two steps: firstly, a Platform Independent Model (PIM) is created. This representation covers:

- Definition of Business Scenarios
- Use Cases
- Requirements
- Context Diagram and independent features

PIM is defined in ISO 19468 General clauses 1-5. Part of the PIM is the definition of the Exchange Patterns and the Functional Exchange Profile (EP and FEP), which further detail with collaboration, sequence, and state diagrams (defined in ISO 19468 clauses 6). After the definition of the business scenarios and the use cases, and which pattern to use to exchange data, the second part of the methodology defines the *technology* to use. Its output maps the rules of the FEP to technology means, specifications of messages, and samples. This is the Platform Specific Model (PSM).

Providing a rigorous approach to the Datex II methodology is defined by means of *Enterprise Architecture, EA*. The concept of EAs, defined in ISO 42010:2022 as the set of *fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles*

¹ See, e.g., <https://www.enisa.europa.eu/news/enisa-news/higher-education-in-europe-understanding-the-cybersecurity-skills-gap-in-the-eu> (Last Accessed, May 2, 2023)

of its design and evolution, facilitate stakeholders in increasing their domain knowledge² and enable managers to take informed business decisions³. Enterprise Architecture describe the system at different levels, either independent or dependent from the technology. This is achieved using *building blocks*, wrapping security countermeasures, and used to compose Datex II projects in both PIM and PSM. Using the Lego Brick analogy, such building blocks are used in different ways to construct (and secure) complex projects.

In this work we will propose an EA approach to help the Datex II engineer to build projects which are secure by design. The work is structured as follows:

- In Section Enterprise Architecture, Viewpoints and Views we will introduce the concept of Enterprise Architecture in detail
- In Section The Methodology we will depict the methodology that will elicit the views to build a secure project
- In Sections Architecture Repository – ABB and Architecture Repository – IBB we will provide the metamodel and the corresponding building blocks

Enterprise Architecture, Viewpoints and Views

Enterprise Architecture (EA) is the organising logic for business process and IT infrastructure, reflecting the integration and standardisation requirements of the firm's or project's operating model⁴. An EA is a description and a representation of a system in terms of frameworks, models, and blueprints which are diagrams, schema, or other visualization of the architecture at the conceptual, logical, and physical level. An architectural approach to system analysis and design defines a rigorous and repeatable methodology to define which are the system components and the rules for their interaction, to understand how the system works and what are the requirements for obtaining a desirable behaviour. Some models produced by EA are prescriptive, in the sense that the design activities define how the system shall be; other models are descriptive in the sense that they describe how the system is. Usually, the activities of system design rely on a *Reference Architecture (RA)*, which is a conceptual model providing the template to represent an *Enterprise Architecture* in a particular domain.

Central to the Datex II, are the architectural constructs of *viewpoint* and *view*, which is borrowed from ISO 42010:2022. A view is defined as a part of an Architecture Description that addresses a set of related concerns and it is tailored for specific stakeholders⁵. A view is specified by means of a viewpoint, which prescribes the concepts, models, analysis techniques and visualizations that are provided by the view. A typical example of views is a computer program's code. The very same source code is analysed with different purposes by a functional analyst who will be concerned about the performances, and by a security researcher who is interested in unbounded loops, use-after-free, and passwords stored in immutable objects. Those are two different views created by their respective viewpoints.

² Article cited: Franke, U., Cohen, M. & Sigholm, J. *What can we learn from enterprise architecture models? An experiment comparing models and documents for capability development*. Softw Syst Model 17, 695–711 (2018).

³ Article cited: Iacob, M.E., Meertens, L.O., Jonkers, H. et al. *From enterprise architecture to business models and back*. Softw Syst Model 13, 1059–1083 (2014)

⁴ Article cited: Jeanne W. Ross, *Enterprise Architecture, Driving Business Benefits From It*, MIT Center for Information System Research, 2006, available at https://cistr.mit.edu/publication/MIT_CISRwp359_EntArchSlctdRsrchBriefs (Last Access May 2, 2023)

⁵ See, e.g., <https://www.opengroup.org/togaf>, (Last Access May 2, 2023)



An *Architecture Framework* establishes a common practice for using architecture descriptions within a particular domain of application or stakeholder communities. Examples of AFs are the following.

- Smart Grid Architectural Model (SGAM)⁶ whose community is Smart Grid security engineers.
- Reference Architectural Model for Industry 4.0 (RAMI 4.0)⁷ (cite) whose community is in manufacturing.
- IDSA Reference Architecture Model (RAM)⁸ whose community is composed by Data Scientists.
- Integrating the Healthcare Enterprise (IHE)⁹ whose community is healthcare professionals and vendors of Healthcare IT.

These models/frameworks share the concept of *Architecture Layers*¹⁰. Layers enable stakeholders to express different concerns regarding the system to be designed, usually from a technical level up to a business level.

A *Reference Architecture* provides a generic template for solution architectures, usually targeted at dealing with a particular domain. Addressing Cybersecurity straight in the Reference Architecture is a form of *Security by Design* since the concern is dealt with straight from the initial definition of the technical solution of the system. Therefore, Cybersecurity should be a compelling concern for Reference Architecture, to allow the architects to develop a Solution Architecture from a template which can be considered secure. A reference architecture is usually composed of *building blocks* which are both architectural assets wrapping specific functionalities, collect business data, application and technology requirements used at design level (*Architectural Building Block, ABB*), and assets defining the implementation and technology requirements for interoperability (*Interoperability Building Block, IBB*). Details of building blocks are defined in the next Section.

The Security Architecture Framework

In this section the Security Architecture framework is introduced. According to ISO/IEC 42010, an “architecture framework establishes a common practice for creating, interpreting, analyzing, and using architecture descriptions within a particular domain of application or stakeholder community”. In this section, the community is bound to the users of NAPCORE, and DATEX II in general, focused on Cybersecurity. The description will be performed in using the Archimate formalism.

- Section “Archimate” will briefly introduce the formalism used (that will be further detailed when discussing about other Frameworks in Section *Mapping with other architectures* for DSA).
- Section “Building Blocks” defines the concept of Building Blocks, their syntax and semantics
- Section “Architecture Principles” describes the principles in which the Security Architecture Framework is built upon
- Section “The Metamodel” will describe the language and the architectural principles.
- Section “Architecture views” is used to describe the point of view of the architecture for the different stakeholders (developers, system integrators, etc.).

⁶ See, e.g., <https://www.cenelec.eu/areas-of-work/cen-cenelec-topics/smart-grids-and-meters/smart-grids/> (Last Access May 2, 2023)

⁷ See, e.g., DIN SPEC 91345:2016-4 Reference Architecture Model for Industry 4.0, <https://www.beuth.de/en/technical-rule/din-spec-91345/250940128> (Last Access May 2, 2023)

⁸ See, e.g., <https://internationaldataspaces.org/publications/ids-ram/> (Last Access May 2, 2023)

⁹ See, e.g., <http://www.ihe.net> (Last Access May 2, 2023)

¹⁰ See, e.g., <https://www.opengroup.org/archimate-forum/archimate-overview> (Last Access May 2, 2023)



Archimate

Archimate is an industry standard from Open Group, and it is a graphical notation for modelling architectural concepts. The core framework is composed by three layers, as follows.

- **Business.** It models the organisation structure business functions and processes, services, and products to external parties;
- **Application.** It documents application landscape, interfaces, and dependencies that exist between applications;
- **Technology.** It describes the hardware, software, and network infrastructure;

Each layer is divided into *structures*, further specializing the components, allowing to express architectural concepts in logical sentences.

- **Active**, anything that is capable to perform a behaviour. For instance, an element which is in
 - **Business/Active**, is a typical business role;
 - **Application/Active**, is an application component;
 - **Technology/Active**, is a hardware or software component;
- **Behaviour**, represents the nature of the thing being performed. For instance, an element which is in
 - **Business/Behaviour**, is a business function or process;
 - **Application/Behaviour**, is an application function;
 - **Technology/Behaviour**, is a technology function;
- **Passive**, represent what the behaviour acts upon (typically data elements). For instance, an element which is in
 - **Business/Passive**, represents business object;
 - **Business/Application**, represents data;
 - **Business/Technology**, represents physical storage of data;

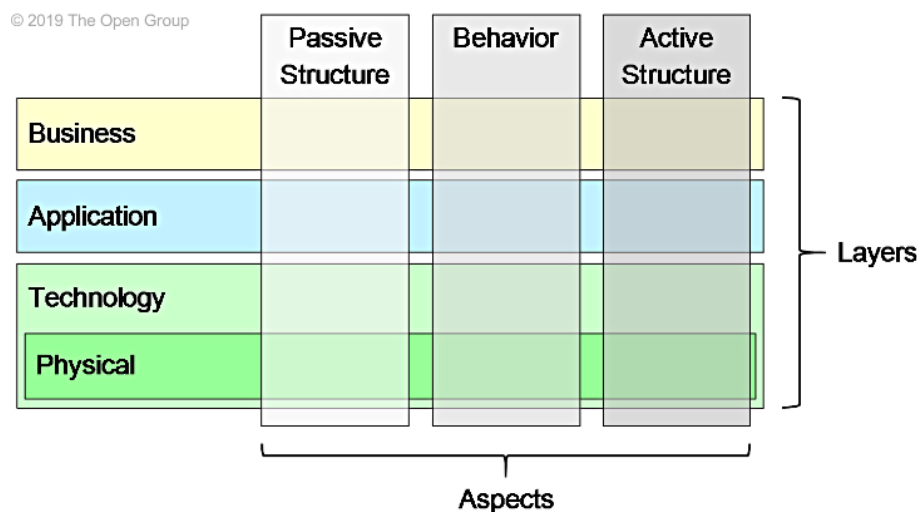


Figure 1 A graphical representation of Archimate CORE layers and its structures

Archimate has been designed to add a visual support to architectural frameworks such as TOGAF. Thus, additional layers have been introduced, such as:

- **Physical**, to model facilities, factories, equipment;

- **Implementation & Migration**, to describe implementation strategies;
- **Strategy**, to describe capabilities and resources;
- **Motivation**, identify stakeholders, and justify the project, or set goals.

Building Blocks

This document adopts the TOGAF concept of building blocks, and more specifically Architecture Building Blocks (ABB) and Interoperability Building Blocks (IBB)¹¹. According with TOGAF 9, a building block is defined as follows.

- A package of functionality defined to meet the business needs across a domain,
- It has a defined boundary and offers services that are generally recognizable by domain experts (Notably, the concept of “defined boundary” guarantees, to some extent, the *technological sustainability* of the architecture. When an architecture is composed by many BB, if the technological evolution of a single building block is limited to its boundary, it will create evaluable impacts in the overall system design)
- It may interoperate with other, inter-dependent building blocks, creating a *composable enterprise architecture*;
- It considers implementation and usage, and evolves to exploit technology and standards;
- It may be assembled from other building blocks.

In this document, we define two kind of building blocks:

- Architecture Building Block, ABB, as an abstract component that captures architecture requirements and directs and guides the development of interoperability building blocks. The subset of all available ABBs which describes a system (e.g., a NAP) from high level perspective is named *reference architectural template, RAT*
- Interoperability Building Block, IBB. It is a concrete element which provides implementation details of a specific technology or a design pattern, or a profile of a standard or norm, which is devoted to implement a specific security countermeasure. The subset all available IBBs defines the requirements that the implementation SHALL follow, and it is named *Interoperability Technological Template (ITT)*

Briefly, the ABBs are used as a method to describe the security countermeasure of a specific system, while the IBB are derived from the high-level RAT implementing specific technology and guiding the implementation and the deployment.

A typical ABB has the following format:

- **Name** (e.g. Accountability ABB), which briefly describes the aim of the ABB
- **Rationale**, what this ABB provides. It is a description of the technology, or architectural pattern, which is provided by the ABB
- **Tag** (one of AVAILABILITY, CONFIDENTIALITY.*, etc). Notably, Datex II *features* are included (as defined in the exchange PIM, and they are: *client identification, supplier identification, integrity, confidentiality, state the intended recipient, client authentication, supplier*

¹¹ IBBs are similar in concept as TOGAF 9 solution building blocks, but they add interoperability aspects such syntactical and semantical requirements to allow an interoperable data exchange. See, <https://pubs.opengroup.org/architecture/togaf91-doc/arch/chap37.html>, last access May 29, 2023



authentication, client authorization, non-repudiation). These tags are used by the methodology (see below) in order to refer to the ABB when deriving the RAT

- **Generic Requirements.** It is the set of the requirements that the ABB imposes. The format of the requirements is as follows

Brief name	Description	Variability Point
AccountabilityABB-1-L	The System SHALL perform operation	
AccountabilityABB-2-T	The system SHOULD implement the service	Use this option only if the system is under control

A requirement is indicated uniquely by the ABB name, a incremental number, and a suffix, where **L** stands for legal requirement (e.g., coming from a national legislation), **T** means technical. The variability point (available only when the prescription level is **SHOULD** or **SHOULD NOT**), further suggests (but not prescribe) guidance

- **Views.** An ABB view may be viewed by different stakeholders, for instance a IT architect, or a traffic manager. They are interested in different aspects. In addition to the architecture views (see below), ABB may have specific interest views to explain with various angles, what the ABB provides
- **Related ABB:** indicates ABBs that are related to the functionality, and they should be considered for implementation
- **Threat mitigation (MITRE):** list of threats mitigated and addressed by this ABB, according with the MITRE Tactics, Techniques, and Procedures from MITRE
- **Controls implemented:** the list of ISO 27001 and similar controls that this ABB satisfy.
- **Used by:** List of RATs which are using this ABB
- **Done Checklist:** a checklist to do in order to be sure that the ABB has been implemented correctly – this can be mutated as a “Definition of Done” in Agile environments
- **IBB:** the list of IBB that realize this ABB
- **Contributors**
- **History**

A typical IBB has the following format:

- **Name:** similarly, to the name of an ABB, it contains both a description and a code
- **Rationale:** description of the technology and rationale why the technology has been chosen
- **Requirements:** the list of requirements implemented by this IBB in the same notation as ABB
- **Similar IBB and Dependencies:** a list of IBB which provide similar functionalities, and a list of IBBs that must be implemented in order to attain the security countermeasure defined in this IBB
- **Known Use:** ITTs which are using this ABB
- **Technical Implementation:** description of the profile of the standard
- **Test Assertions and exploitation suggestions:** a test assertion following the OASIS TAG guidelines, and a set of practical means to test. A test assertion must be in the following format

TA id	<i>Id of the test assertion</i>
Normative Source:	<i>The requirement ID being tested</i>

Target	<i>The target system, the System Under Consideration, SuC</i>
Prerequisite	<i>List of potential prerequisites in order to repeat the test</i>
Prescription Level	<i>Optional/Mandatory</i>
Predicate	<i>The predicate being tested, from the requirement</i>
Variable	<i>Eventual variables that may pose the test at risk of reproducibility</i>

- **Implementation details:** eventual implementation details or sample code
- **Done checklist,** as defined in the ABB section.

Test assertions are used as guidance for evaluating the correct implementation of an IBB. As a result, from the stocktacking initiative performed within the project, it has been observed that other verticals use test assertions to establish events named “connect-a-thons”, or “project-a-thon”, where systems test interoperability against each other and a third party checks for conformance, guided by the tool. The tools and the methodology are the basis of the EU GITB initiative (Global Interoperability of Test Beds).

Test assertions may be also used by developers to self-assess the correct implementation, without the use of tools, as this deliverable contains the needed evidence to reproduce the intended result.

Architecture Principles

The overarching principles applied in the design of the architectural components, (ABB, IBB) are based on such concepts as Openness, Reusability, and Technological neutrality and data portability amongst the others defined in the European Interoperability Framework (EIF)¹².

The building blocks foster **composability**, e.g., they are designed in the way that they can adapt to the various NAP and Datex II deployments. A Cybersecurity architect may use either one of the RATs defined in this document, or either create its own one by selecting the ABBs and deriving the corresponding IBB, either guided or not by the methodology defined in this document.

The design of the building blocks is constrained by the following Design:

- The technology is Development & Deployment where possible. Research and Development is discouraged.
- It shall be based on open standards.
- They shall be production ready and shall not mandate any internal NAP architecture

The Metamodel

The architectural metamodel is depicted in the Figure below.

¹² See, e.g., <https://joinup.ec.europa.eu/collection/nifo-national-interoperability-framework-observatory/european-interoperability-framework-detail>, last Access May 29, 2023



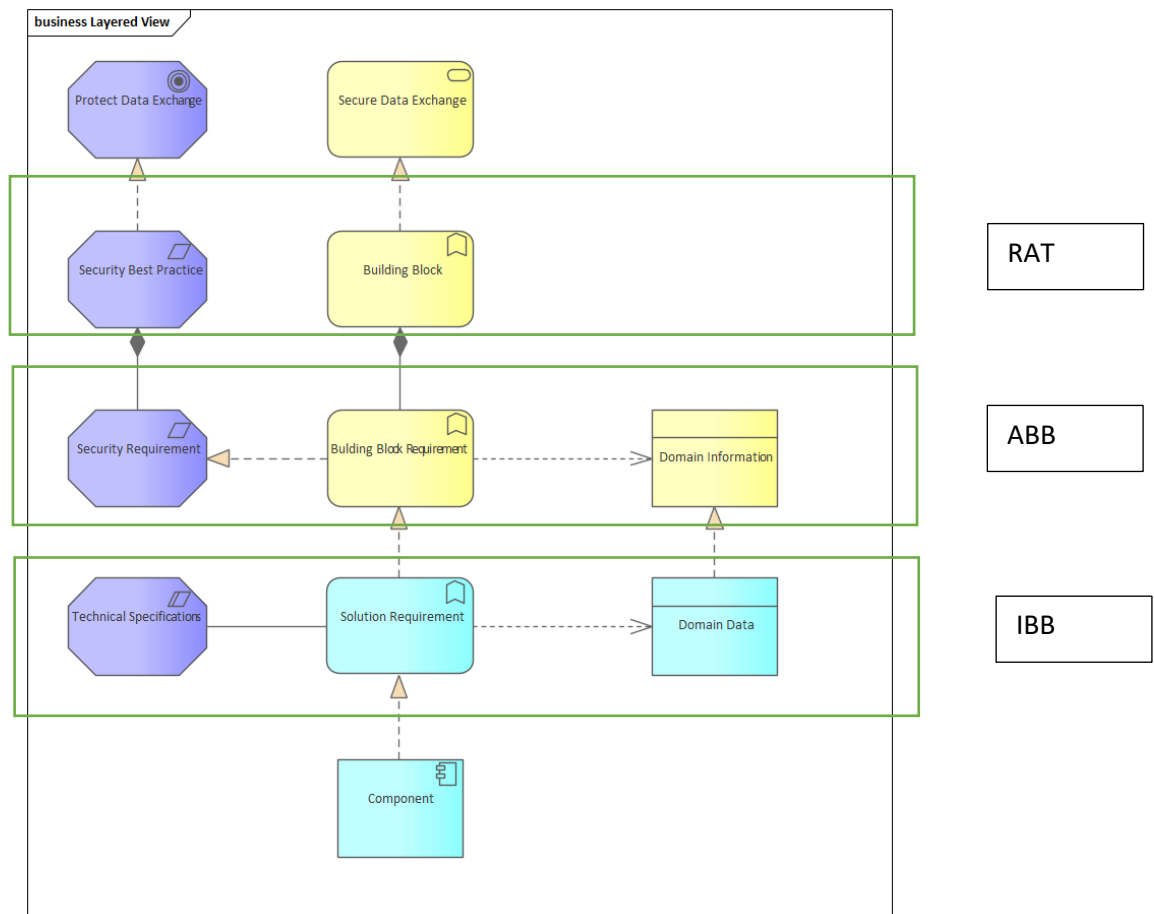


Figure 2 The architectural metamodel

The metamodel, written using Archimate, can be read as follows¹³.

Architectural building blocks provide a set of **requirements**, expressed as functionalities, realizing **security requirements** (expressed as requirement), and access domain information (such as traffic regulations, VMS, expressed as domain object). **Building blocks**, ABBs (expressed as functionalities) are collections of such requirements to realize the **secure data exchange** service. Security requirements are aggregated from **security best practices** (expressed as requirement), that realize the **protected data exchange** (expressed as goal). The building block requirement are realized by **solution requirements** (expressed as application functionality) related to **technical specifications** (expressed as constraints), accessing domain data (expressed as domain object). The realization of a solution requirement is implemented by single software or hardware **components** (expressed as components).

This version of the Reference Architecture will define building blocks for the following concepts.

ABB	
ABB Name	Description
Data Integrity ABB	Defines requirements to guarantee data integrity by means of techniques such as Cryptographic hashes, digital signatures
Data Encryption ABB	Defines requirements to guarantee the data confidentiality by means of techniques such as Encryption, or Access Control
Non-Repudiation ABB	Defines requirements to guarantee various non-repudiation mechanisms (origin, receipt, submission, delivery) for data

¹³ Typographically, **bolded** terms relate to concepts, while underlined terms represent the relations, and “(expressed as” means the specific Archimate elements

Audit Trail ABB	Defines requirements to guarantee rules to define what to audit and how
Authorization ABB	Defines requirements to guarantee transaction authorization (i.e., authorization aspects for data in transit)
Node Authentication ABB	Defines requirements to guarantee the (mutual) authentication of nodes (such as NAP)
Data Provenance ABB	Defines requirements to guarantee the data provenance (e.g., where the data comes from and it is used)
Data Classification ABB	Defines requirements to classify data based on confidentiality, availability, and integrity
Consistent Time ABB	Defines the requirements for establishing synchronised time
IBB	
Digital Signature IBB	Defines and profiles the technical standards for XML and JSON signatures, establishing trust
TLS IBB	Defines the requirements and use of TLS 1.3 with BCP 195
Certificate Pinning IBB	Defines the methodology to pin a certificate, establishing trust
Web Service Security IBB	Defines the profile of Web Service Security
Audit Syntax IBB	Defines the syntax and the semantics for the audit trails
JWT IBB	Defines and profiles the technical standards for transaction authorization based on OAuth 2.0 and JWT
Provenance and Data Sharing IBB	Defines the technical standards for the provenance of data

The landscape of the building blocks is shown in Figure below.

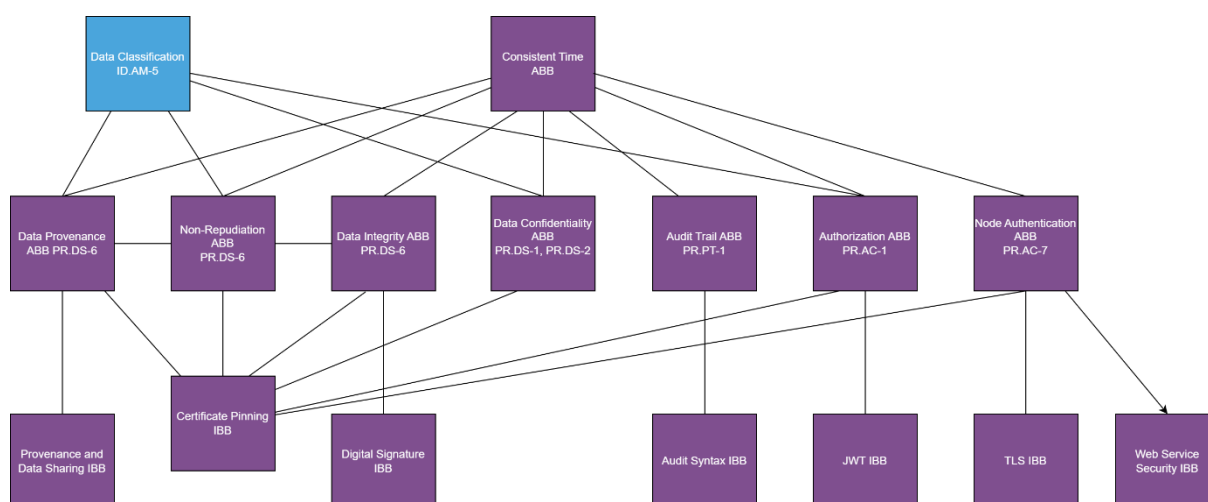


Figure 3 Building block landscape

The colours of the building blocks are related to the NIST framework version 1.1. Purple means protection building blocks, while blue means identification. For each ABB it is represented the NIST cybersecurity framework subcategory that it implements. As example, ID.AM-5 prescribes that “Assets are prioritized based on classification, criticality, resources, and impact on the mission”, thus, Data Classification is a foundational countermeasure to attain the requirement of the control. Notably, this mapping should be considered just an indication to show that they aim to *protect* resources, as each building block will have this detailed information in the “controls implemented” section. Lines represent the dependencies of each building blocks. From the drawing Consistent Time is required by

almost all building blocks, and that Data Classification is suggested to be performed before the decision of applying a specific RAT. Technically, certificate pinning (e.g., the decision whether to trust or not a specific certificate by out of band means) is required by all building blocks that uses asymmetric cryptography.

Architecture views

In Enterprise Architecture, a viewpoint represents a particular view on a building block whose audience is a specific stakeholder. For example, business architects are more likely to be interested in business processes, while system administrators in the deployment of components, while software developers in code samples. Each building block (either architectural or interoperability) has the following viewpoints.

Business Process View

In this view there are representations of:

- business event triggers,
- internal and external actors and roles,
- processes that realise business services.

Stakeholders	Domain application architects	
Concerns	High level description of the functionality	
Layer	Archimate	Business / Application
	RAMI	Business / Information
	IDSA	Business
Aspects	Behaviour, active structure, passive structure	

As example, the following archimate diagram shows the view for the process “Encrypt Data”.

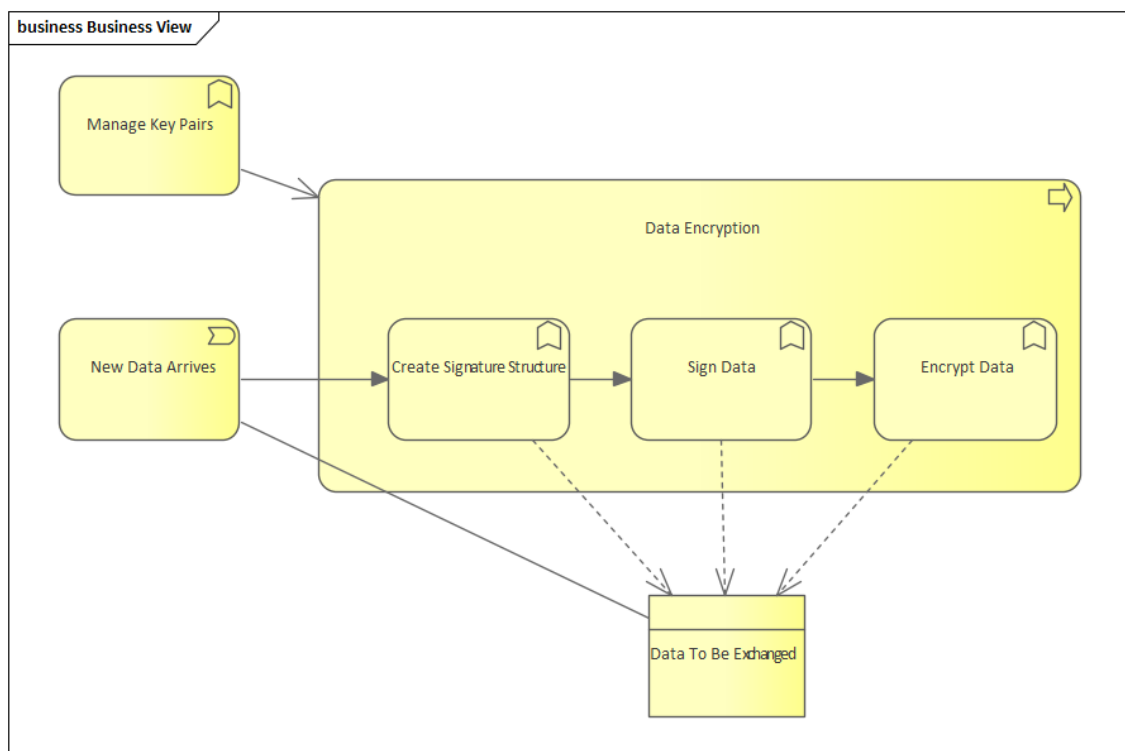


Figure 4 Example Business Process View

The figure is described as follows: when **new data arrives**, it triggers the business process of **data encryption**. Having **Key Pairs** available, the data encryption process is composed of set of activities. Firstly, it **creates a signature structure**, then it **signs the data** arrived, then it is **encrypted**. Every service accesses the object **data to be exchanged**.

Application View

In this view there are representations of:

- Application landscape,
- Interfaces
- Highlight dependencies.

Stakeholders	IT Software architects	
Concerns	Application data exchange	
Layer	Archimate	Application
	RAMI	Information / Communication
	IDSA	Functional
Aspects	Behaviour, active structure, passive structure	

The following picture shows an example of Application view for the Data Encryption.

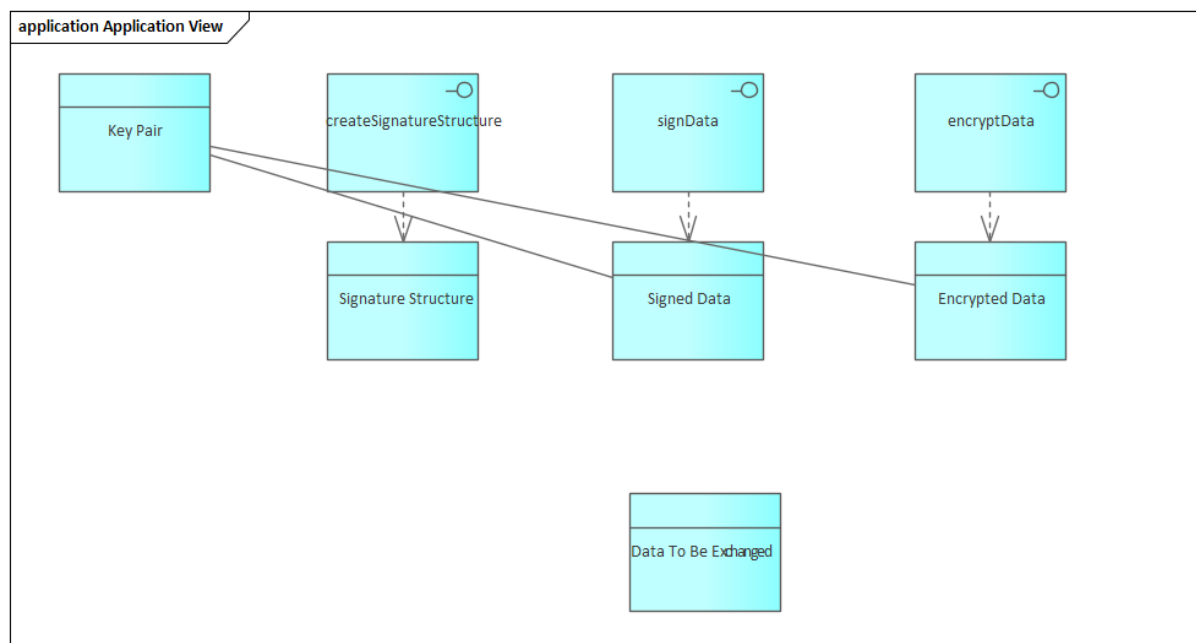


Figure 5 Example of Application View

In the figure below, the business view is enriched with application view components. Each business function is accessing the respective data. For example, the *manage key pairs*, accesses *key pairs*. Similarly, the other business functions access their relative data. In order to create data, interfaces are

defined. In particular, the application shall expose three interfaces, the *createSignatureStructure*, *signData*, and *encryptData*¹⁴.

Technology View

In this view there are representations of:

- Nodes representing client and servers,
- Model Hardware Resources
- System for infrastructure level.

Stakeholders	IT System Administrators	
Concerns	Technological aspects (such as code, or deployments)	
Layer	Archimate	Technology
	RAMI	Integration/Asset
	IDSA	System
Aspects	Behaviour, active structure, passive structure	

The following picture shows an example of Technology view for the Data Encryption.

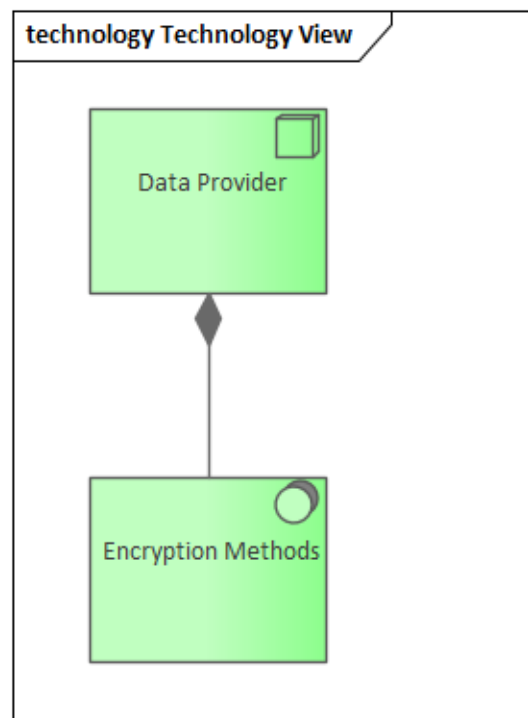


Figure 6 Example of Technology View

In the figure above, the business and application views are enriched with technological components. The interfaces are implemented in a system software, *encryption methods*, owned by a node, a *data provider* (e.g., a NAP).

¹⁴ The meaning of this will be shown in the building block. In few words is to allow the variability points of the building block itself, when data just need to be signed, or just encrypted, or both.

Mapping with other architectures

The International Data Space Association (IDSA)¹⁵ defined a Reference Architecture Model (IDSA-RAM), which is following the same concepts shown in this document. Systems are divided in *Layers*, and *Perspectives*. Cybersecurity is considered, as in this document, a cross-cutting concern, spanning from the technological part up to the Business requirements. This is shown in the next Figure.

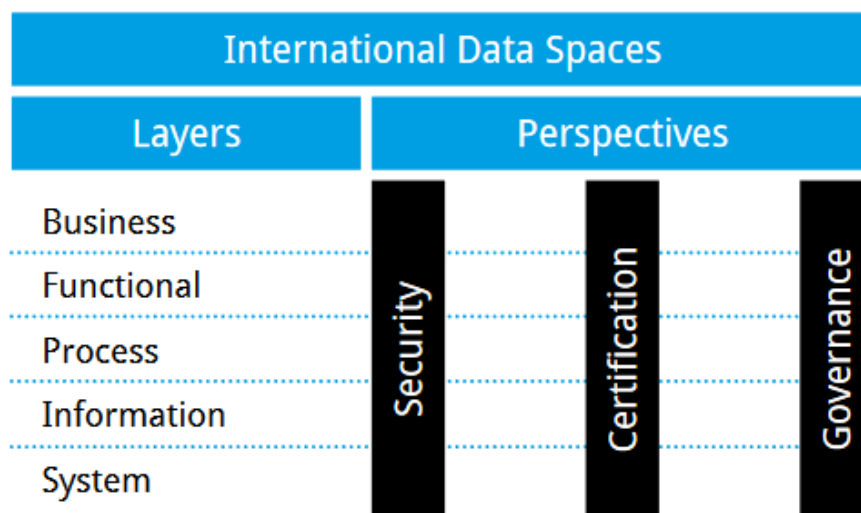


Figure 7 The IDSA RAM

IDSA RAM layers are mapped to Archimate according to the following table. Moreover, each view of the Security Architecture Framework has been described also in terms of IDSA RAM.

IDSA	Archimate
Business: Specifies different roles of IDS, and the main activities for each role	Business: offers products and services to external customers, which are realized in the organization by business processes performed by business actors.
Functional: defines the functionalities of the exchange	Application: supports the business layer with application services which are realized by (software) applications.
Process: specifies the interactions in BPMN notation	
Information: defines a conceptual model which makes the use of linked data	

¹⁵ See, e.g., <https://internationaldataspaces.org>

System: is concerned with logical software components. Integration, configuration, deployments.	Technology: offers infrastructure services (e.g., processing, storage, and communication services) needed to run applications, realized by computer and communication hardware and system software.
--	--

Architecture Repository – ABB

Node Authentication ABB

Name

Node Authentication ABB – NABB

Rationale

Establishing a secure network communication channel is foundational of any data exchange. To be considered secure, a *communication channel* between two services shall enjoy such properties as

- Confidentiality – through TCP packet encryption
- Authentication – through cryptographic means among the services
- Authorization – through the validation of specific access credentials

This ABB provides such **basic security** properties needed by all form of secure systems.

When performing data exchange, applications establish a communication channel using a protocol over a network socket. By design, any user in the network can observe, modify, and forge packet in transit, even more a malicious user. Those are named *passive* and *active* attacks, respectively. A malicious user can impersonate either a sending or a receiving application, effectively hindering the data confidentiality, availability, and integrity of the data being exchange. As example, a NAP is exposing data to be shared with peers with a specific contract. Without establishing a secure communication channel as defined by this ABB, there are no other means to guarantee the authenticity of the sender, nor that the receiver has a data sharing agreement in place.

Notably, this ABB works at *node* level. A node is an application that, in an end-to-end fashion, needs to exchange data with another application.

Nodes are divided in two types:

- **Secure Node:** is a system that provides security and privacy services (user authentication, secure communications, security audit reporting, and security policy enforcement) for *all the software and services running in the system*. A secure node is a system where not only the application is sharing data using secure communication channels, but also the operating system is controlled by the node itself for which, for any access to data, this is protected and audited adequately. Notably, a secure node vendor does not need to invent its own disk drives or write its own operating system. Contractual control over security is considered to be

sufficient. If exceptions to the concept of secure node on the system are known, a short list of exceptions may be provided. It is up to a risk analysis to indicate that those exceptions are not significant, and the full list of exceptions is documented. This permits cloud-based and other system architectures to claim to be a secure node when there are sufficient contractual controls to ensure that security and privacy services cover the entire relevant hardware and software stack. An example of a secure node is a sensor or an actuator, or a Fixed C-ITS station for which a compliance with a common criteria protection profile is required. Also, a server whose only mean to access is through a Privilege Access Manager (PAM) which records all the sessions, eventually blocks commands, and audits all the access to the data, is considered a secure node.

- **Secure Application:** is a system that shares data exclusively with a secure communication channel, but it does not own its entire stack. An example of secure application is a server, either local or in the cloud, where there is no audited access to the data or to the services. Or a mobile app, sharing resources within a mobile phone.

The definitions of secure node or secure application should be used to guide the risk analysis and the cybersecurity tests. For instance, a secure node can maintain a secret (similarly to the concept of *confidential client*, see AUTHZABB) while a secure application should rely on other external services (e.g., a Hardware Security Module).

It is worth noting that to establish a secure node or application, a Public Key Authority is usually contacted and used. Thus, this ABB assumes that the entire governance model of the Certification Authority includes Cybersecurity by design. In fact, it is assumed that the organization shall maintain procedures, policies, services, and IT administration staff trained to maintain the authentication mechanism based on a PKI secured. In particular, the PKI should maintain adequate records and backups of authentication information, maintain a local and online certificate revocation list in order to remove untrusted certificates. Moreover, the organization should have procedures to replace keys and certificates that have been lost or compromised in a timely fashion. The highest impact when dealing with secure nodes is usually the number of certificates installed remotely. Secure nodes such as sensors, or C-ITS, are relatively complex to access and to change, thus an automated solution is strongly recommended, using standards such as ACME (rfc 8555), CMP (rfc4210), SCEP (rfc 8894), EST (rfc 7030). A suggested time for certificates is between 60 days and 1 year, depending on the automation efficiency. Note: shorter the certificate validation is, less relying on certificate revocation.

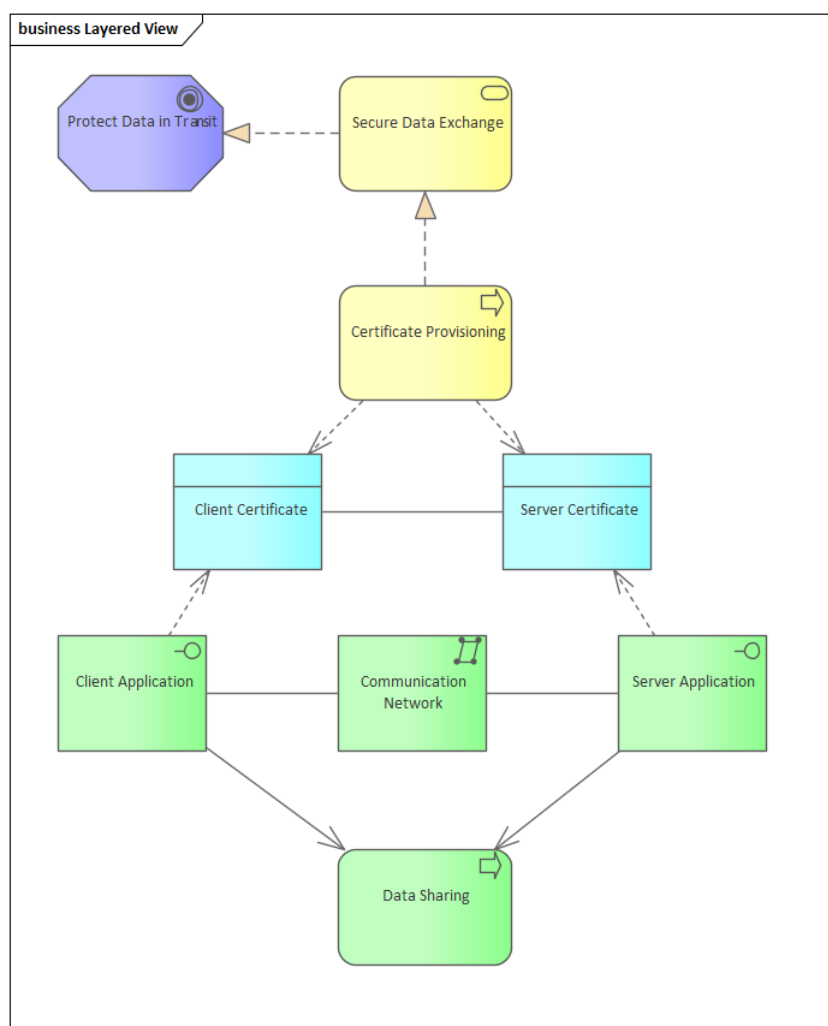


Figure 8 NODEABB model

The archimate model for the ABB is defined in the Figure above. The motivation for this ABB is to *Protect Data in Transit*, which is realised by the *Secure Data Exchange*, in turn realised by establishing a business process on *certificate provisioning* which accesses both *client* and *server* certificates. Those are also accessed by *client* and *server* applications (either secure node or secure applications) to communicate through a *communication network* for *data sharing*.

Tag

Client Authentication, Supplier Authentication, Client Identification, Supplier Identification, Confidentiality in Transit

Generic Requirements.

Brief name	Description	Variability Point
NODEABB-1-T	Either the Secure Node or the Secure Application SHALL share data only through a Secure Communication Channel for all network connections	

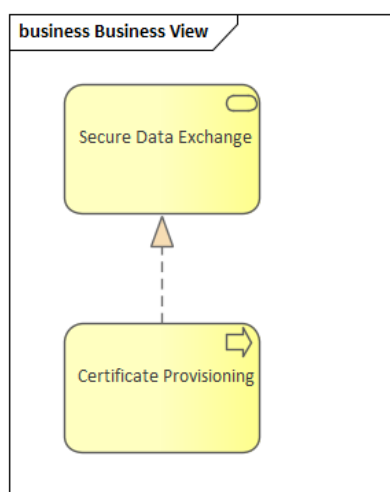
NODEABB-2-T	Provide sufficient authentication methods, based on risk assessment, to ensure that only authorized users have access to the system	Use of a Secure Remote Access – Use of Certificate Pinning – Use Multifactor Authentication
-------------	---	---

Views

Business Process View.

The Business Process View is composed of two business processes: a secure data exchange service and a certificate provisioning function. This means that when deploying this ABB, the security architect SHALL define at least two processes.

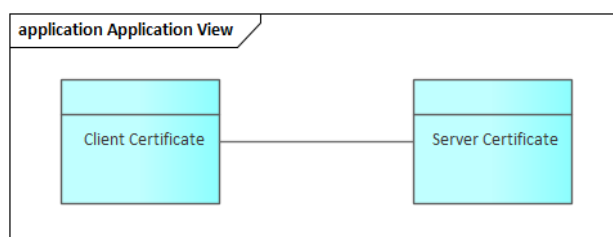
- Certificate provisioning: in this case, a PKI is contacted or established, with criteria based on the IT security policy of the company. Usually, a commercial CA is contacted. In case where machine-to-machine authentication is needed (e.g., in NODEABB-2-T), an internal CA may be provided



- Secure data exchange: in this case, a procedure or a policy is shared with the stakeholders interested in the data exchange. For example, it is shared the entire certificate chain, or the certificate practice statement (or at least its public part), among the stakeholders. A potential "trust list" may be created, where all root and sub-CA of all the stakeholders are shared through a central point of contact. Potentially, the single certificate may be exchanged out of band in case of certificate pinning.

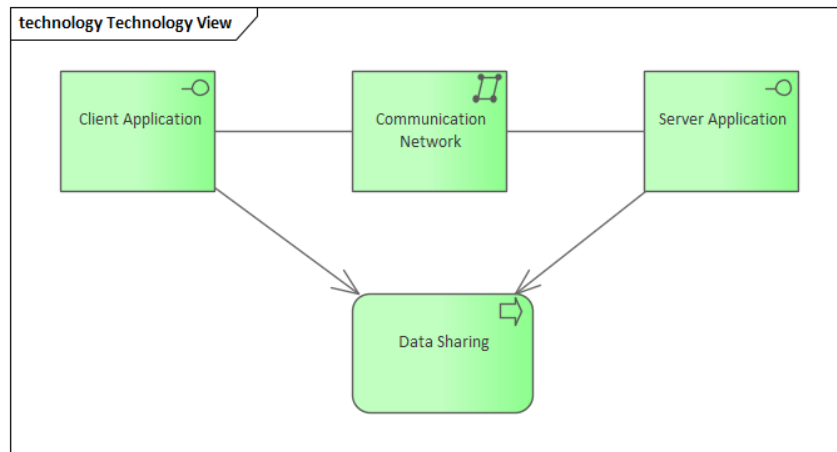
Application View.

The Application View is composed of two Certificates / Key Pairs which shall be made public / protected. In this case, for such components which can be considered critic (e.g., a NAP), these data objects may be stored in a Hardware Security Module certified Common Criteria EAL 5+.



Technology View.

The technology view is composed of two applications, a server (e.g., a Tomcat server running on a Linux host) or a client (e.g., a Java or a Python application) requesting data through an TCP/IP communication network (e.g., a corporate LAN or even internet). Notably, both the client and the server can be secure node or application. In fact, it is suggested to protect the physical and logical access to the machines through the use of a Secure Remote Access (Privilege Access Manager) solution, in order to audit, log, and protect the information from the unwanted sysadmin.



Related ABB

[Consistent Time ABB] to provide synchronized clocks, needed by all cryptographic algorithms.

[Audit Trail ABB] to provide interoperability and secure reporting of all the security-relevant activities

[Authorization ABB] To define authorisation means over an authenticated channel.

Threat mitigation (MITRE)

Data Source: Network Traffic, ID DS0029, protection over *data transmitted across a network (ex: Web DNS, Mail, File) that is either summarized (ex: Netflow) and or captured as raw data in an analysable format (e.g., PCAP)*

Mitigations: Encrypt Network Traffic, M808

Techniques Addressed by Mitigation: T0839 Module Firmware, T0842 Network Sniffing, T0857 System Firmware, T0860 Wireless Compromise, T0887 Wireless Sniffing

Controls implemented.

- NIST CSF v1.1 ID.AM-3, *Organisational communication and data flows are mapped*, ISO/IEC 27001:2013 A.13.2.1, A.13.2.2, IEC 62443-2-1:2009 4.2.3.4
- NIST CSF v1.1 PR.DS-2, *Data-in-transit is protected*, ISO/IEC 27001:2013 A.8.2.3, A.13.1.1, A.13.2.1, A.13.2.3, A.14.1.2, A.14.1.3, IEC 62443-3-3:2013 SR 3.1, SR 3.8, SR 4.1, SR 4.2

Used by

N/A at the moment. Will be further described in future versions of this deliverable.

Done Checklist

The building block is successfully implemented if

- 1) all network communications sharing data are mapped and secured using the requirements defined
- 2) a risk analysis is performed evaluating the variability points identified, thus evaluating which IBB to use (certificate pinning, authorization, audit trail).

IBB Realization

[Certificate Pinning IBB] to provide means to enforce out-of-band authentication of remote certificates exchanged during the establishment of a secure channel

[TLS IBB] The implementation of TLS, used to establish a secure channel

[Web Service Security IBB] For using WS-Security instead of TLS

Contributors

Massimiliano Masi <mmasi@autostrade.it>

History

First release, June 5, 2023

Authorization ABB

Name

Authorization ABB – AUTHZBB

Rationale

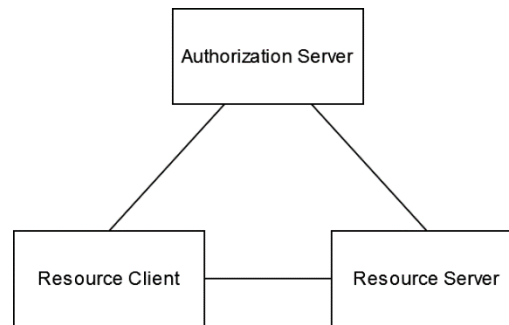
This profile enables network authorization when using RESTful transactions.

Being authorized means that the user, or the machine (also known as “the principal”) has legitimate access to the resource. The authorization includes identifying the user and the application that is making the request to the HTTP RESTful server, so that server can make further access control decisions. Example of applications are when a user accesses a web application, or when a RESTful client is accessing a resource through a set of APIs, or via a mobile phone.

This ABB makes use of OAuth 2.0 [rfc6749] as authorization protocol. Other mechanisms (e.g. HTTP basic authentication) SHALL NOT be used. Notably Github, Facebook, and Google APIs use it. OAuth relies on authentication scenarios called *flows*, and this ABB constraints the use of three specific flows, named “Authorization code”, “client credentials”, and “Proof Key for Code Exchange” (PKCE), according to the current best practice [draft-ietf-oauth-security-topics-21, exp 31 March 2023].

The exchange is as follows.





A resource client (the application) interacts with the Authorization Server to get an access token. The application then uses the access token to access the resource on the Server. In turn, the server is able to make reliable and accurate authorization decisions, and it is expecting a common method for receiving access tokens. Clients and application developers expect a common method for obtaining and using access tokens.

OAuth 2.0 clients are defined in two types: *public* and *confidential* (see, as a reference the Node Authentication ABB). A public client is a client who is not capable to hold a secret (e.g., a key material), and it is typically a web browser, or a Smartphone app. A *confidential* client is capable to hold a secret instead. When architecting the system, this definition will guide the authorization flow to use.

OAuth 2.0 is agnostic to the type of tokens to be exchanged. A token is typically included in a wrapper that can be either *access*, *id*, or *refresh*, used to accessing a resource, identifying a principal, or used to obtain additional tokens when they expire, respectively. Implementations use Opaque tokens (e.g., crypto material who are proprietary by the server). This ABB SHOULD use Json web Tokens (JWT), as defined in JWTBB.

Authorization flows identified by this BB are the following:

- Authorization code grant [rfc6749, Sec 4.1]: it is used to obtain both access and refresh tokens and it is optimized for confidential clients. It is based on HTTP redirect, and it may require user to enter username and password in a form. The implementation requirements are defined in section 4.1.1 of rfc6749. The definition of Scope will be further defined by additional BBs
- Client credential grant [rfc6749, Sec 4.4]: The client can request an access token using only credentials, when the client is requesting access to the protected resources under its control, or those of another resource owner that have been previously arranged with the authorization server. This is optimized for confidential clients. The implementation is defined in Sec 4.4.1 and subsequent.
- PKCE [rfc7636]: this flow is optimized for public clients whose risk of authorization code interception attack is relevant.
- Native Apps [rfc8252]: this flow is optimized for authorization requests initiated by native apps (e.g., swift, kotlin, java), or made through external user-agents.

The OAuth 2.0 authorization framework requires client identification, which may be based on a *client_id* parameter. Depending on the flow, the use of the *client_id* may be required. For example, the Authorization code grant type requires the use of *client_id* for client identification, while the client credential grant does not.

The framework also requires client authentication for confidential and credential clients. It recommends the use of asymmetric methods for client authentication [Oauth 2.1, Sec 9.1], as specified in the Node Authentication ABB.

When defining and choosing the authorization flow, the OAuth 2.0 Threat Model [rfc6819] SHALL be considered.

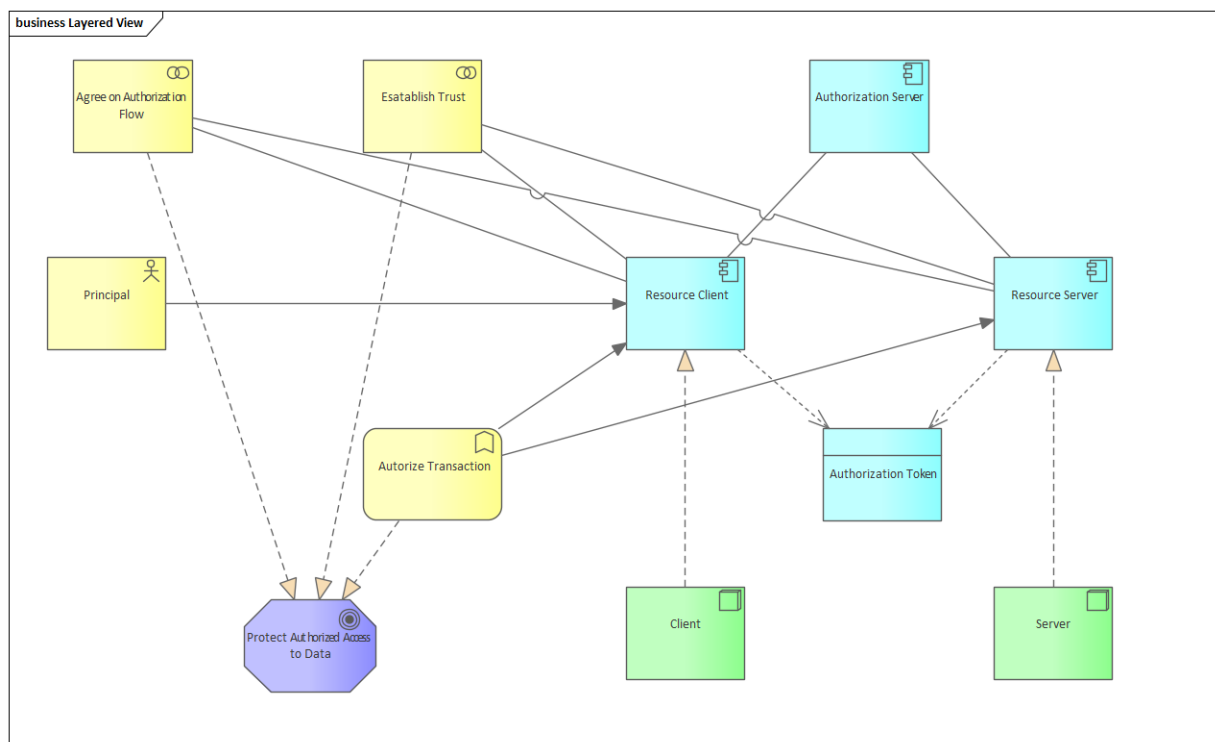


Figure 9 AUTHZABB Archimate Model

The figure above shows the archimate model of the Authorization ABB. In order to provide **authorized access to data**, three business processes shall be in place: parties have **to agree on the authorization flow to use**, **establish trust**, and **authorize transaction**. Those processes are realized by three applications: the **resource client** which can be either public or confidential, the **resource server**, which guarantees the access control based on the claims contained in the **authentication token** provided along with the transaction, and the **authorization server**. The resource client and servers are implemented by the **client** and **server**, respectively. Notably, the authorization server may be outsourced (see the Technology View).

Differently from the Node Authentication ABB, who can provide *basic* authorization capabilities by e.g., denying data access to untrusted nodes, this ABB works at a higher level, as the authorization is given to the specific human or service (the **principal**) owning the resource client.

Tag

Client Authorization, Supplier Authorization

Generic Requirements.

Brief name	Description	Variability Point
AUTHZABB-1-T	The Resource Client SHALL choose an authorization flow based on the type of client and the threat model	
AUTHZABB-2-T	The Resource Client SHALL obtain and incorporate token in a RESTful transaction using the Authorization: Header over a TLS protected channel as provisioned in the Node Authentication ABB	Use of mutual authenticated TLS channel should be governed by a risk analysis
AUTHZABB-3-T	The Resource Server SHALL obtain the token from the transaction, and verify its integrity	
AUTHZABB-4-T	The Resource Server SHALL perform a token introspection	

Views

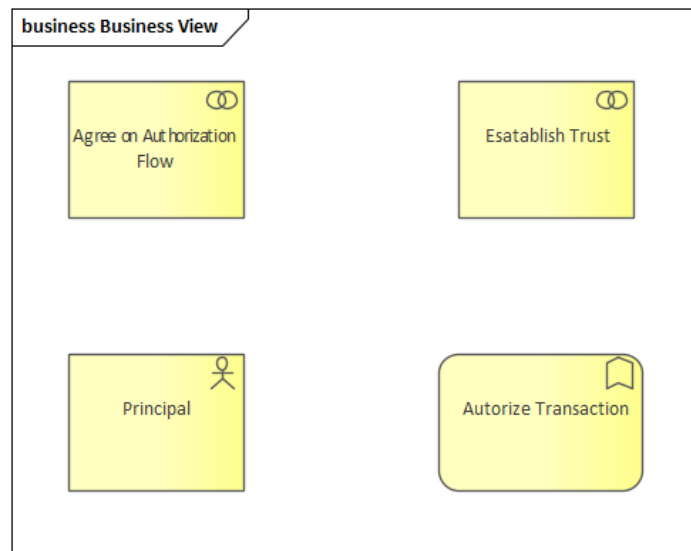
Business Process View.

The Business Process View is composed of several business elements.

- Stakeholders shall agree on the authorization flow to be used. Further guidance can be found in the rfc and in the rationale. Usually the choice depends upon the type of client, either confidential or protected, and if the data exchange is machine-to-machine, or principal-to-principal.
- Stakeholders shall establish trust. The way how it is done, is out of scope of this document. Trust establishment is made usually by means of contractual agreement underlying technical requirements (such as the implementation of the architecture defined in this deliverable), ISO 27001 certifications. An example of multi-stakeholder trust establishment is given by the Central Point of Contact for C-ITS (CPOC) where the EU commission defined a Security Policy document which defines the basis of trust relationships among road authorities, car manufacturer, and users¹⁶.

¹⁶ See, e.g., <https://cpoc.jrc.ec.europa.eu>

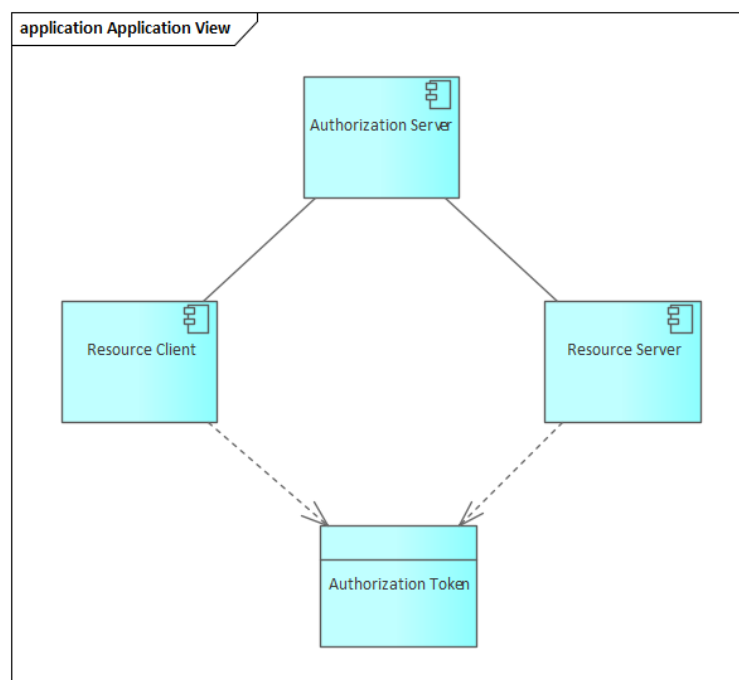




- Stakeholder shall agree on authorize the transaction. There shall be a list of agreed attributes to be exchanged defined as a list of syntactic and semantic values.
- Principals should be provisioned, and their identity should be part of the trust establishment.

Application View.

The Application View is composed of the two components, the client and the server. At deployment level, particular care shall be made to the client in case it is a confidential one (e.g., using a Hardware Security Module, HSM to handle keys, or having a tool which is rotating certificates, as suggested in the NODEABB). Similarly, a Resource Server should be adequately protected, e.g., in an ISO 27001 audited data center where procedures for both logical and physical protection are in place.

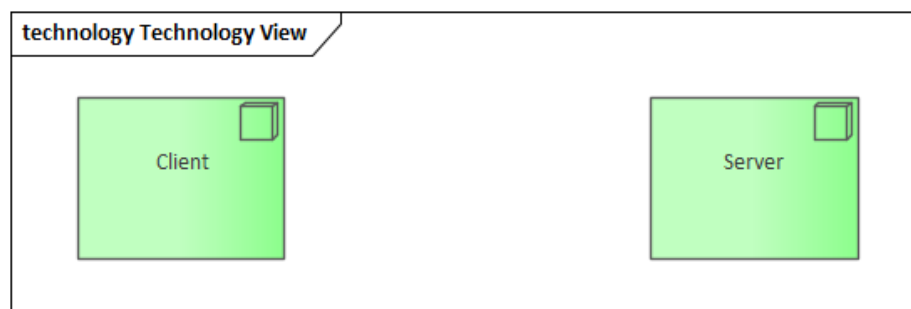


The authorization server is the owner of the identities. This application component may be either interal or external to the data exchange. In some cases, it may be provided by public services, or handled internally by tools such as Keycloak, Shibboleth, or Active Directory.

Note: in case of IDSA RAM components, the client and the server may be logically coupled with the DSA Connector, while the Authorization Server can be logically coupled with the Identity Manager.

Technology View.

Technologically, this ABB is composed of two applications, the client and the server. Notably, the authorization server is deliberately left out of scope, as it may be external to the data exchange.



Both client and servers may be Linux or Windows-based services, exposing the applications.

Related ABB

[Consistent Time ABB] to provide synchronized clocks, needed by all cryptographic algorithms.

[Audit Trail ABB] to provide interoperability and secure reporting of all the security-relevant activities.

[Node Authentication ABB] to provide a secure channel when contacting the authorization or the resource server. Notably that non-TLS connections to the server and to the authorization server MUST NOT be used. The mutual version should be evaluated by a risk analysis as suggested by the Node Authentication ABB.

Threat mitigation (MITRE)

Data Source: Network Traffic, ID DS0029, protection over additional data sources of *Network Traffic Creation*, *Network Traffic Content*, and *Network Traffic Flow*. User Account, ID DS0002, and User Interface, DS0042. More specifically, Web Credentials, ID DS0006, in particular creation and usage. Notably, if no mTLS connection is used, an Adversary-in-the-middle can exploit credential usage.

Mitigations: Encrypt Network Traffic, M808, Encrypt Sensitive Information M1041, Limit Access To Resource Over Network M1035,

Controls implemented.

- NIST CSF v1.1 PR.AC-4, ISO/IEC 27001:2013 A.6.1.2, A.9.1.2, A.9.2.3, A.9.4.1, A.9.4.4, A.9.4.5

Used by

N/A

Done Checklist

The building block is successfully implemented if

- It has agreed a mechanism to exchange tokens
- It has agreed a set of attributes to evaluate the authorization decision upon
- It has been evaluated the classification of the client (public/confidential)
- An authorization server is available, capable to handle client_ids

IBB Realization

[Certificate Pinning IBB] to provide means to enforce out-of-band authentication of remote certificates exchanged during the establishment of a secure channel

[TLS IBB] The implementation of TLS, used to establish a secure channel

[Authorization IBB] provides a minimum set of attributes for a token to be used to authorize, based on JWT

Contributors

Massimiliano Masi <mmasi@autostrade.it>

History

First release, June 5, 2023

Data Integrity ABB

Name

Data Integrity ABB - DIABB

Rationale

Data Integrity is defined by the NIST Glossary as *guarding against improper information modification or destruction and includes ensuring information non-repudiation and authenticity*.¹⁷ Data integrity is foundational to ensure data quality, and trust establishment. It is usually performed by means of One-Way Hash Functions, such as SHA, and in untrusted channels where cryptographic operations are possible (e.g., in IT) with digital signatures. When PKI is used (see NODEABB), integrity protection is also providing data authentication, ensuring a minimum way to provide weak non-repudiation (see NONREPABB). Data integrity is provided both,

- at rest where integrity check may be performed out-of-band or far away in time, or,

¹⁷ <https://csrc.nist.gov/glossary/term/integrity>



- in transit, when data is shared between stakeholders.

Cybersecurity techniques to protect those two different states exists and they cover also long-term signature validation. However, in this ABB the following techniques will be used for both data at rest and in transit:

- ADeS, Advanced Electronic Signatures, including long-term variants
- Hash functions and Canonicalization algorithms

The reason is due to the fact that data is shared and stored in high level concepts, and thus the need to use ad-Hoc HMACs is not foreseen.

The EU regulation 910/2014, the “eIDAS”, sets the regulatory framework for digital signatures, one of the countermeasures to guarantee data integrity. The eIDAS technical specifications¹⁸ make use of *ADeS as a way to implement the information, as well as containers to hold file objects and their associated signatures (the ASiC).

When using long-term signatures, it is advisable to start taking into account the Post-Quantum Computing aspects. In fact, architects shall consider eventual re-signature of data (at rest) algorithms such as Dilithium or other NIST-approved standards. Risk Analysis and bandwidth considerations should guide the evaluations.

Digital signature can be either Detached, Enveloping, Enveloped....

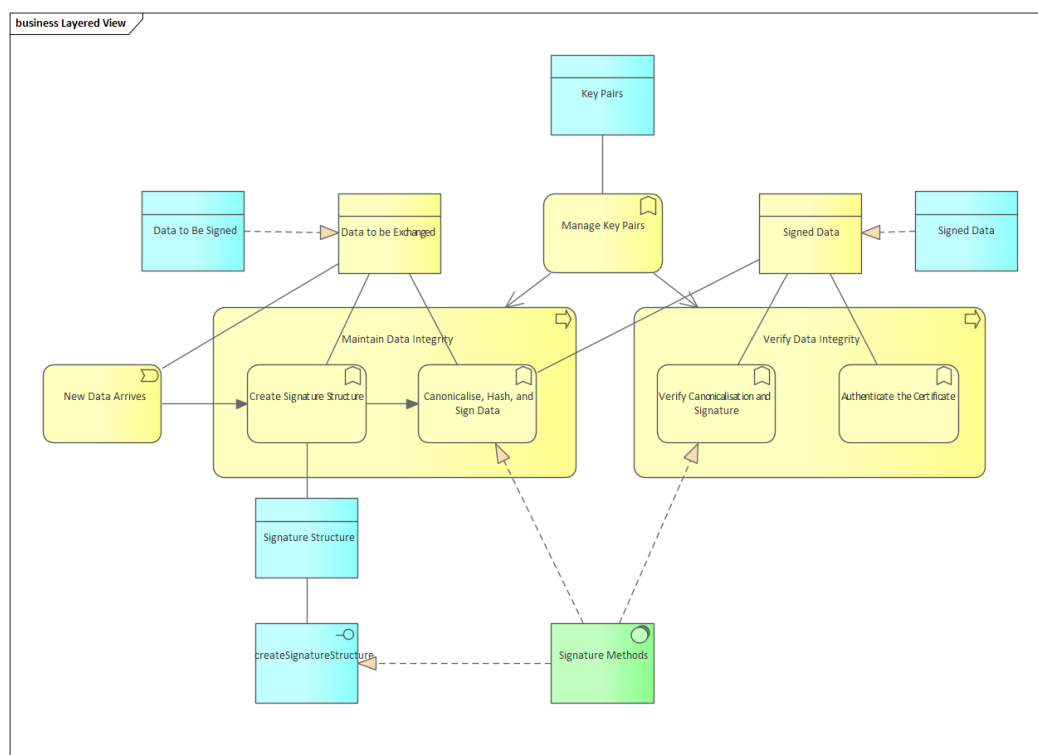


Figure 10 Archimate model for Data Integrity

The data signature is provided by two main business processes:

- Maintain Data Integrity, mandatory, which is devoted to providing integrity, by means of signature,

¹⁸ <https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Standards+and+specifications>

- Verify Data Integrity, optional, which is devoted to verifying the integrity in due time

Tag

Data Integrity

Generic Requirements.

Brief name	Description	Variability Point
DIABB-1-T	Data Integrity SHOULD be provided by means of Digital Signatures	Hash functions MAY be used by sensors in trusted (e.g., air gapped) networks
DIABB-2-T	Obsolete Hash Functions SHALL NOT be used	
DIABB-3-T	Authentication of the data SHOULD be performed when receiving, by means of PKI verification	Risk analysis may be performed to avoid certificate verification
DIABB-4-T	Asymmetric digital signature mechanisms SHOULD be used	Integrity verification by symmetric means MAY be implemented only when a key distribution channel is available
DIABB-5-T	Long-term signatures MAY be used	When the integrity and authentication verification go beyond the validity of the CA, ADeS SHALL be used.
DIABB-6-T	Containers SHOULD be used when dealing with multiple data objects	If multiple data objects are exchanged, the use of containers should be considered.

Views

Business Process View.

The business process view is composed of two processes (maintain data integrity and verify data integrity).

- The *maintain data integrity* process is established upon data to be exchanged¹⁹. Firstly, the process creates a metadata structure, the “signature structure” that contains meta information about how the signature is done. In fact, signature algorithms, canonical form of data, hashing functions shall be defined prior to the implementation of the process. After the

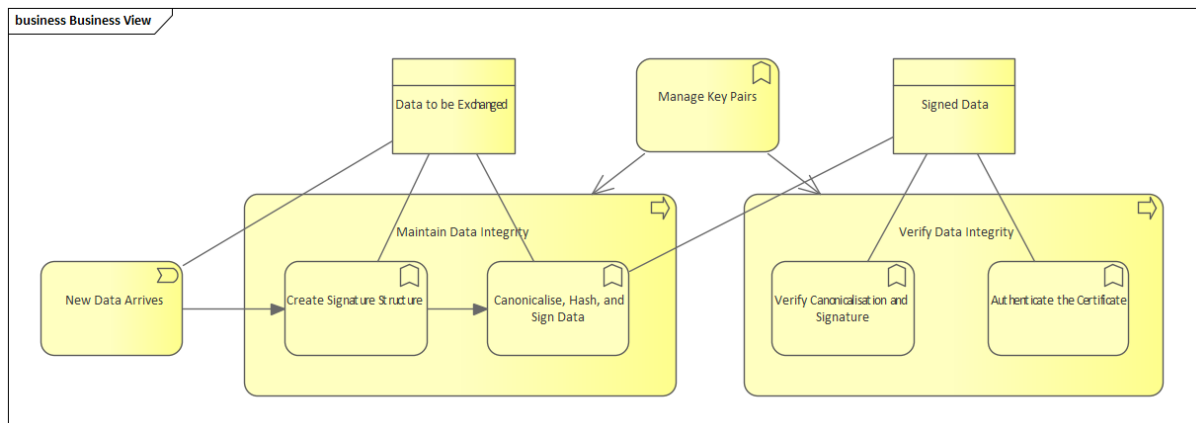
¹⁹ Notably, the notion of data to be exchanged does not constrain this ABB to the concept of data in transit. This ABB also covers data at rest.



creation of the metadata, the signature value is computed: data is canonicalized, hashed, and then signed, creating *Signed Data*.

- The output of signed data is then verified with the *Verify Data Integrity* process. Also, this process is composed of two sub-functions: verify the signature and authenticate the certificate used to sign (to provide integrity and authentication properties).

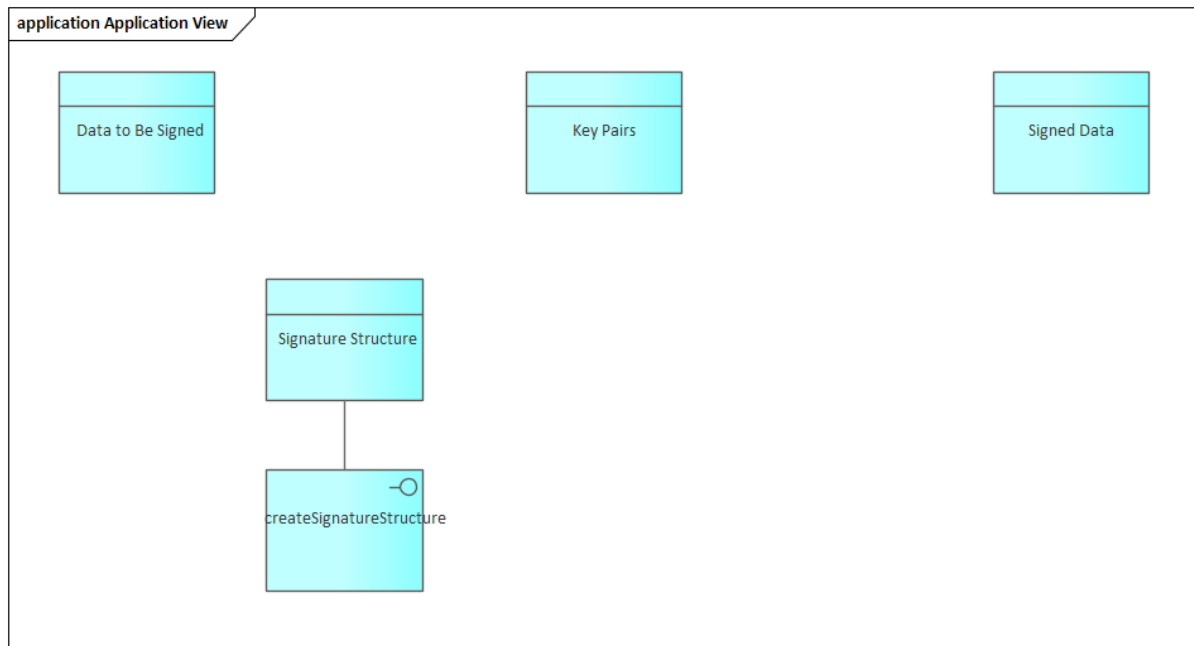
In order to implement the system, trust shall be established (see Views section of Authorization ABB). Once trust is established, a set of key pairs (in form of X.509 certificates) are shared amongst the two processes. Thus, a **certificate policy** on certificate **and trust management** should be put in place, which covers issuing and online revocation of certificates (see, e.g., TLS IBB).



Application View

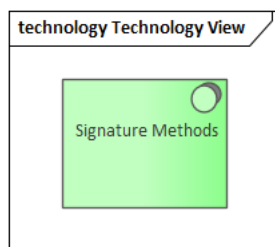
The application view is composed of four data objects and one interface, as follows.

- **Data to Be Signed**, a data structure holding the data in its canonical form, ready to be signed. It is of utmost importance that this data is protected by the application, by, e.g., not storing for long time in the hard disk, or to be handled in memory just for the signing instant.
- **Key Pairs**, a data structure holding the signing keys. Application should secure the keys by storing them in safe place (e.g., HSM, Secure Element, or with access control mechanism, in a file system), and with less exposure in memory as possible (e.g., by zeroing the data structure when used, and not to be stored in immutable objects)
- **Signature Structure**, a data structure which contains the metadata about the signature itself. This information is public, but the specific structure must be defined upfront (e.g., as in the XMLSDG IBB).
- **Signed Data**, a data structure which contains both the signed data and its metadata. The information is publicly available; however the syntax of the structure shall be defined upfront (e.g., JSON, XML, aDES).



Technology View

The technology view is implemented by a several methods, which should be hardened or, at least, reviewed using a SAST approach. The XMLDSG IBB provides a sample implementation.



Related ABB

[Consistent Time ABB] to provide synchronized clocks, needed by all cryptographic algorithms.

[Audit Trail ABB] to provide interoperability and secure reporting of all the security-relevant activities.

[Non-Repudiation ABB] to provide different non-repudiation scheme

Threat Mitigation

M1060: out-of-band communication channel

Controls Implemented

Data Integrity is the pillar of the ISO 27001:2022 CIA Triad, Confidentiality, *Integrity*, and availability. If the organization runs an ISMS, the information shall be documented. This ABB implements the following controls.

- ISO 27001:2022 A.1.8.24, NIST CSF PR.DS-01, PR.DS-02, PR.DS-10

IBB realization

[Digital Signature IBB] Defining XML, ADeS (XAdES, CAdES, PAdES), signatures

Done Checklist

The building block is successfully implemented if

- It has been agreed a format for data signature
- It has been defined a policy to trust cryptographic material

Contributors

Massimiliano Masi mmasi@autostrade.it

History

First release, March 14, 2023

Data Provenance ABB

Name

Data Provenance ABB – DPABB

Rationale

Provenance is the foundation of data quality. In a decentralized or distributed setting where multiple sources share data with several endpoints, creating a chain of trust and quality among the stakeholder is crucial. Provenance has two different viewpoints: from the *source*, it answers the question “where the data goes”, while from the *destination*, answers the question “where the data comes from”. Moreover, Data Provenance tracks also all the permutations of data over its lifecycle. The objective of Data Provenance is similar to Non-Repudiation ABB: it collects evidence about a data exchange, in a interoperable way. However, Non-Repudiation ABB is not efficient when in a distributed scenario, as the non-repudiation tokens may grow exponentially (two – or more token per single point-to-point exchange) while Data Provenance is designed to hold the information in a single, central, document.

According with W3C²⁰, provenance is defined as information about entities, activities, and people involved in producing data, which can be used to form assessments about its quality, reliability and trustworthiness.

Intuitively, Provenance is defined as follows.

²⁰ See: <https://www.w3.org/TR/prov-overview/> Last Accessed Nov 28, 2023.



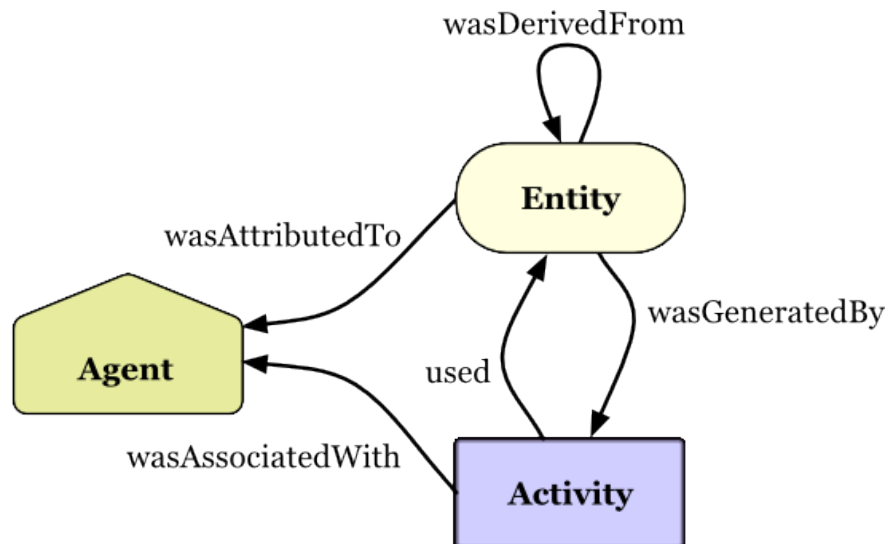


Figure 11 The W3C Definition of PROV

Entities are data that can be derived by other entities as well. For instance, an XML representing traffic conditions may be defined upon information about a previous condition. This is expressed by using the *wasDerivedFrom* action. *Entities* are generated by *Activities* (e.g., publications, transfer, creation, modification) via the *wasGeneratedBy*. *Activities* may use other entities to be performed. Both *Entities* and *Activities* are attributed to and associated with *Agents*, respectively. *Agents* are either humans (in the form of accounts) or generic principals, such as Software components (e.g., the NAP).

Such concepts are applied to every phase of the data lifecycle in a document, the *Provenance Document* which is detached from data. The Provenance Document can be either in XML or in Turtle representation. The set of all provenance documents constitutes the *provenance ontology*.

Subscribers to this ABB, should stipulate a data permutation process to the central provenance service, by sending standardized notifications to update the provenance knowledge base about the specific data item.

Provenance tracking has been implemented using two major patterns: using a *proxy* that intercepts messages and extrapolate provenance data to create and update provenance records, and *client-server*, where a client sends, voluntarily, provenance updates. The proxy pattern has been chosen by healthcare industry, with the IHE mXDE and QEDm profiles²¹: data sent by a source to a consumer is intercepted by a proxy. Since the messages are all standard, it is feasible for the proxy to extrapolate provenance evidence.

In NAPCORE, differently from Datex II, data can be in any format. Thus, the IBB will utilize the client-server approach. However, a new IBB using the proxy pattern for selected Datex II events will be provided.

²¹ See: <https://profiles.ihe.net/ITI/mXDE/index.html> Last Accessed Dec 1, 2023.

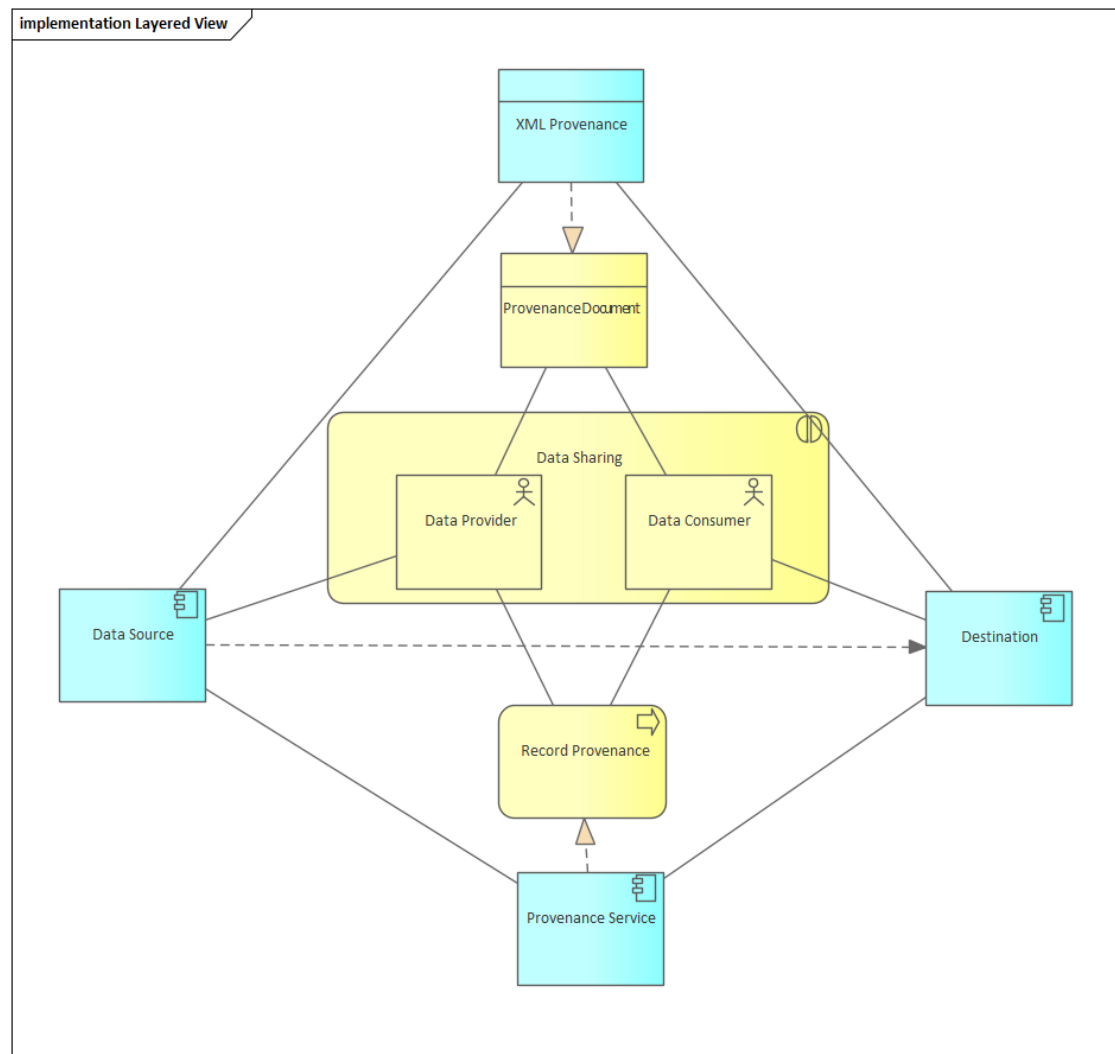


Figure 12 DPABB Archimate Model

The figure above shows the archimate model of the Data Provenance ABB. In order to fulfil the business process to **record provenance**, the two actors, **data provider** and **data consumer** shall be in place. They shall agree on the content of the **Provenance Document** meaning providing a vocabulary for the entities, for the actors, and the agents that will concur with the relevant transactions to record provenance. Data Provider and Consumer are realized by two generic components, one or more **Data Source(s)** and **Destination(s)**. Notably, source and destination can be recursively interchanged. The Provenance Document is realised in this specification in **XML Provenance**. This document is updated and made available by a **Provenance Service**. In fact, every time there is a need to track provenance, the source updates the service with the relevant information. Viceversa, every time there is a need to validate the provenance, the Provenance Service is contacted to retrieve the XML provenance document.

A crucial role is the definition of the provenance **template**: the set of agents, events, and actions, that are a case for a provenance record to be updated, and the relevant transactions. Moreover, it is compelling to define *what to record*: the data exchange, the data manipulation, the data creation, etc. The data for which provenance is requested could span from a single XML up to a dataset. Each data shall be uniquely identifiable, to attain its provenance tracking.

This specification will provide a IBB describing a template for the NAP to exchange data. Others may follow.

Tag

Data Provenance

Generic Requirements

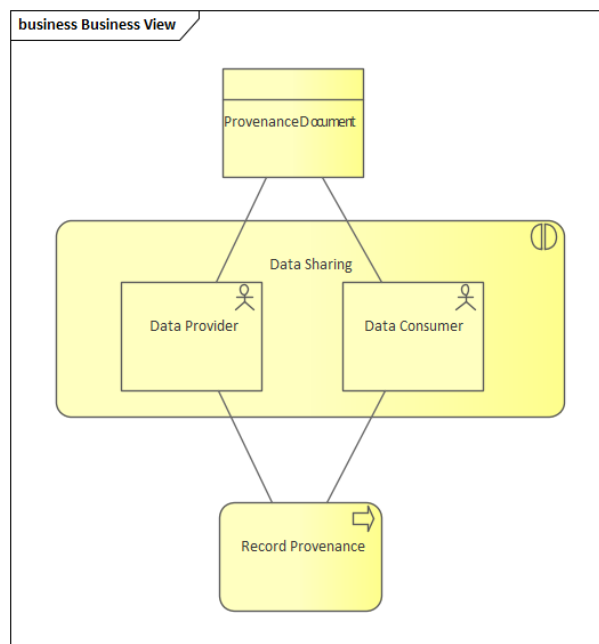
Brief name	Description	Variability Point
DPABB-1-T	All data sources and destination SHALL agree on one or more provenance templates	
DPABB-2-T	The Data Source SHALL update the Provenance Service whenever a relevant data modification, or exchange, happens	
DPABB-3-T	Data SHALL be uniquely identifiable	

Views

Business Process View.

The business process view is composed by two actors, the data provider and the data consumer, who share a provenance document, while sharing data. In particular, we have

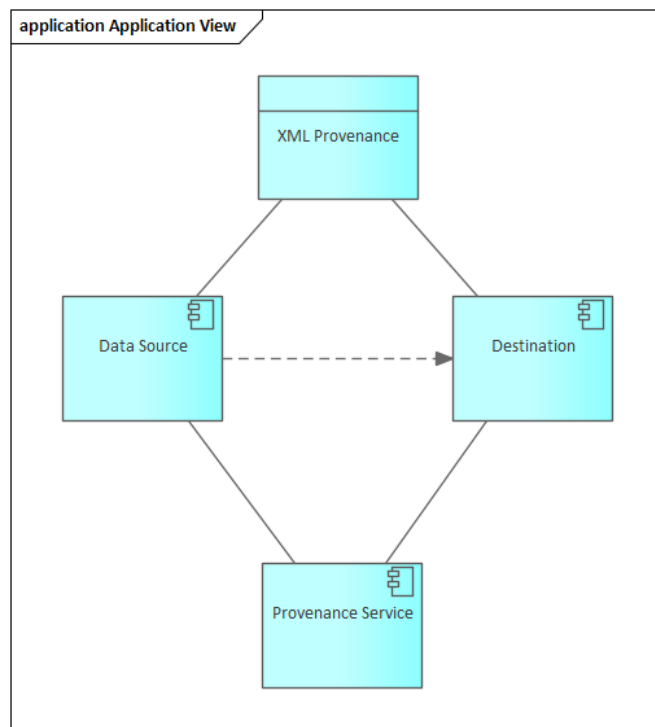
- Stakeholders shall agree on one or more provenance template, i.e., the format of the Provenance Document. Stakeholders need to document what is the process of Recording Provenance (making data identifiable, what re the manipulation that should be recorded, where to update the information, and when the information can be retrieved)
- The Provenance Document shall be trusted, thus agreements between the stakeholders and the authority holding the provenance document shall be in place.

**Application View**

The application view is composed by the two components, the data source and the destination. They share a provenance template in a form of XML Provenance shared by the Provenance Service. The connection between the application components and the provenance service shall be performed using



NODEABB. The provenance service shall expose an URL which will be accessible by all the components, unauthenticated.



Related ABB

[Consistent Time ABB] To provide synchronized clocks, needed by all the algorithms.

[Node Authentication ABB] to provide a secure channel when contacting the provenance service.

[Data Integrity ABB] to provide a signature over the data to be exchanged and the provenance document itself.

Threat mitigation (MITRE)

Data Source: File, ID DS0022

Controls implemented

- ISO 27001:2022 A.5.14, A.5.28, A.5.32, A.5.33, NIST CSF DE.DP-2. As per ISO 27002:2013, section 13.2.2, "procedures for ensuring traceability and non-repudiation".

Used by

N/A

Done Checklist

The building block is successfully implemented if:

- It has been agreed a provenance template
- It has been agreed which are the data manipulation/transfer activities to record
- It has been established a trust with the central Provenance Service

IBB Realization

[Provenance Data Sharing IBB] The provenance template for the data sharing



Contributors

Massimiliano Masi mmasi@autostrade.it

History

First release, November 30, 2023

Data Encryption ABB

To Name

Data Encryption ABB - DEABB

Rationale

Data Encryption is a mechanism to provide confidentiality, e.g., granting access to data only to authorized principals who owns a key. This ABB supports confidentiality by providing a means to encrypt any kind of data in a transport independent way. Its approach enables access to data to be targeted to specific recipients, realising end-to-end encryption, i.e., not allowing intermediaries to access the entire data set, or parts of it (in case of XML-based payloads).

This ABB allows for multiple methods of identity and key management, including X.509 certificates, symmetric keys, or passwords.

To realise these objectives, the ABB makes use of standard technologies such as

- Cryptographic Message Syntax (CMS, rfc 5652)
- XML encryption (XMLENC)
- JSON Web Encryption (JWE, rfc 7516)

Encryption can be potentially grouped with digital signature (as suggested and provisioned by the DIABB). The order when performing signature and encryption is important:

- Sign-before-encrypt: in this way, end to end confidentiality is guaranteed, over authenticated data enjoying integrity
- Encrypt-before-sign: in this way, the data consumer can authenticate the principal who performed the encryption, but not the data itself
- Sign-Encrypt-Sign: in this way, the data consumer can verify both cases

This building block do not prescribe any encryption and signature scenario, as the decision is performed on a use-case-basis.

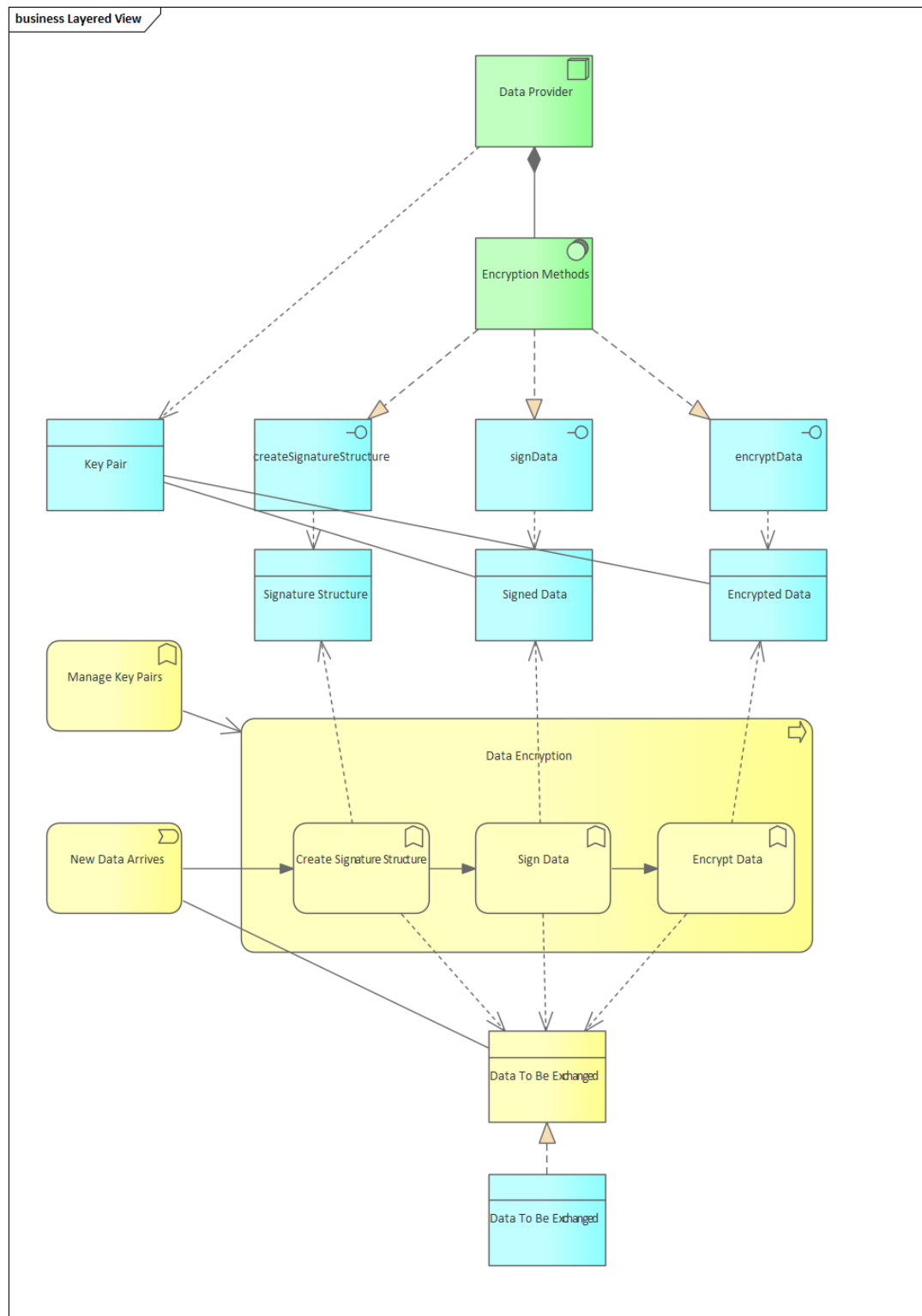


Figure 13 Archimate model for the DEABB

The ABB is shown in the figure above. The data encryption process is performed everytime *new data arrives*, on the object *data to be exchanged*, realized as application component with the same name. A process to manage the keys exists, which are used both for eventual sign (as *create signature structure* and *sign data*), and the mandatory *encrypt data*. For each business function, a data object

and its corresponding interface is extracted. The reason why they are specified is due to the fact that signature is optional, and it is part of *Encryption Methods*.

Tag

Data Confidentiality

Generic Requirements.

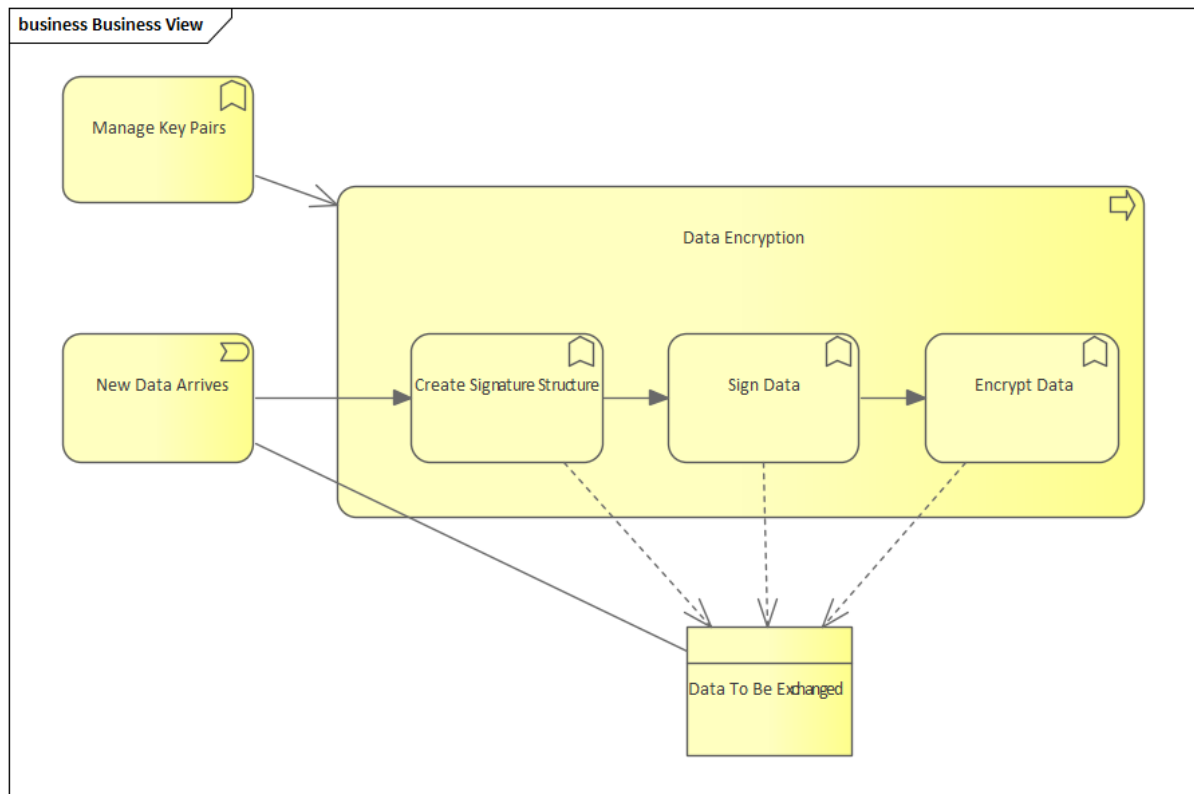
Brief name	Description	Variability Point
DEABB-1-T	Deprecated encryption algorithms MUST NOT be used	
DEABB-2-T	An encryption and signature scenario SHOULD be agreed ex-ante	Depending on what has to be authenticated, either the data, the encryption, or both
DEABB-3-T	A method to securely exchange keys SHALL be in place	

Views

Business Process View

The business process view is composed of a main process, who encrypts data, when a new data arrives (the triggering event). The function Manage Key Pairs is essentially a definition of a method to have the key material to prepare the Data To Be Exchanged. In particular, we have the following.

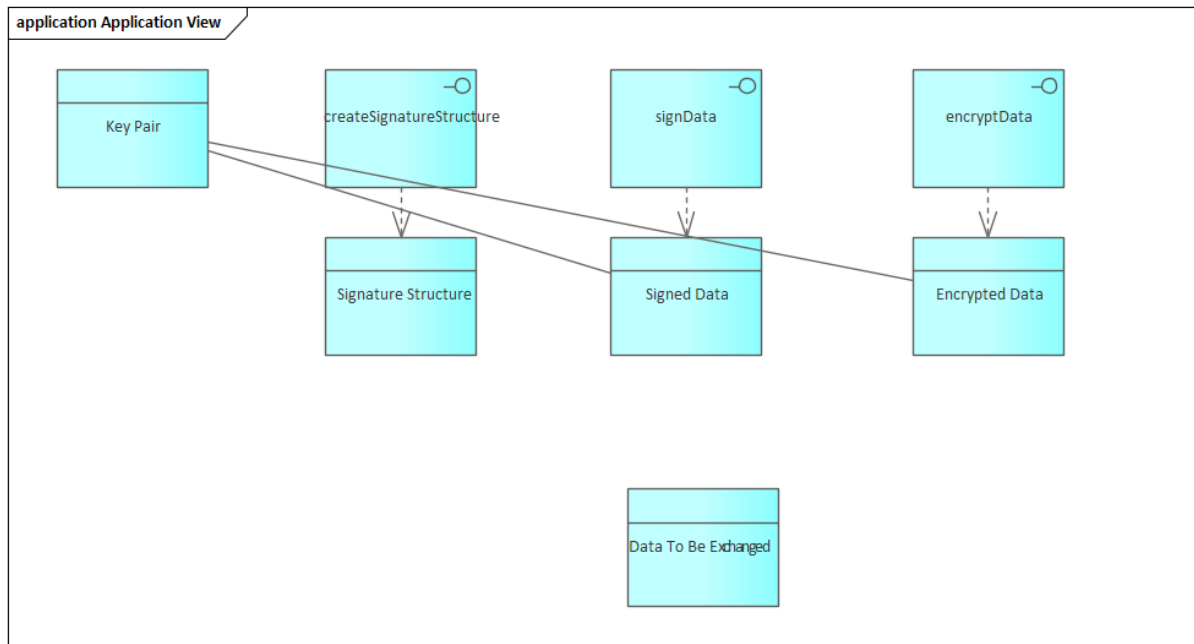
- Manage Key Pairs: is a business function that handles the key material for the (eventual) signature and encryption. Symmetric keys, or private keys are examples of such information. In case of encryption, usually the used key is public. However, when providing signature services, the key material shall be protected and documented similarly as defined in DIABB.
- New Data Arrives: this is a business event that triggers the main process.
- Data To Be Exchanged: this is a business object representing the entire data that is to be exchanged, encrypted and (eventually) signed.
- Within the Data Encryption business process three sub-process exists:
 - Create Signature Structure, as defined in DIABB
 - Sign Data, as defined in DIABB
 - Encrypt Data, it is the process of applying the chosen encryption algorithm with the given key material. In order to perform the encryption/decryption process successfully, the algorithms shall be agreed upfront.



Application View

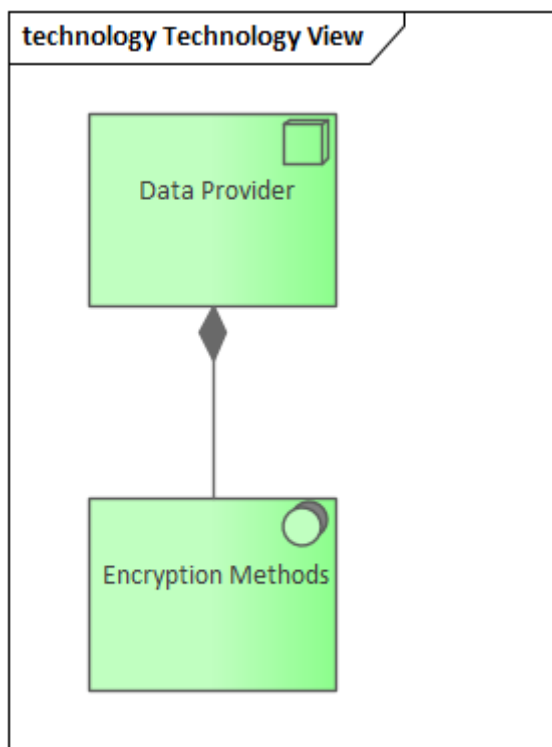
The application view is composed of four data objects KeyPair, SignatureStructure, Signed Data, and Encrypted Data. The first three objects are similar to the respective defined in DIABB. Similarly, the interfaces createSignatureStructure and signData are defined in DIABB. In particular, the data object Encrypted Data and its related interface encryptData, are the application who apply the encryption algorithm. Particular care shall be taken when implementing the algorithm, e.g., proprietary, non-standard algorithms shall be avoided, and available encryption protocols should be obtained from EUCRYPTO or the NIST references²².

²² See, e.g., <https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines>



Technology View

The technology view is composed of two elements: the Data Provider and the Encryption methods. No particular requirements are defined for those components.



Related ABB

[Consistent Time ABB] to provide synchronized clocks, needed by all cryptographic algorithms.

[Data Integrity ABB] to provide information about the signature structure

Threat Mitigation (MITRE)

Countermeasure: M1041, Encrypt Sensitive Information. Threats are multiple, e.g., Adversary in the Middle (T1557), Content Injection (T1659), etc.

Controls Implemented

Data Confidentiality is a pillar of the ISO 27001:2022 definition of CIA: *confidentiality*, integrity, and availability. In particular, this ABB realizes the following prescriptions:

ISO 27001:2022 A.1.8.24, NIST CSF 2.0 PR.DS-01, PR.DS-02, PR.DS-10.

Used By

N/A

IBB realization

N/A. IBBs for data encryption will be made available in next versions of this deliverable.

Done Checklist

This building block is implemented when:

- A encryption algorithm and its metadata has been agreed
- A syntax for the container of the encrypted data has been agreed
- A policy for the management of encryption and signing keys has been defined by both parties

Contributors

Massimiliano Masi <mmasi@autostrade.it>

History

First release, March 14, 2024

Non-Repudiation ABB

Name

Non-Repudiation ABB - NRABB

Rationale

Non-repudiation services are mandated to generate, collect, maintain, make available and validate evidence concerning a claimed event or action in order to resolve disputes about the occurrence or non-occurrence of the event or action. Non-repudiation mechanisms provide protocols for the exchange of non-repudiation tokens specific for non-repudiation service. Non-repudiation tokens shall be stored as non-repudiation information that may be used subsequently in case of disputes by services named Trusted Third Parties (TTP).

The following level exists, when dealing with non-repudiation.



- Application logging: unreliable data sent over unreliable message exchange (e.g., developer debug information or info over UDP syslog – there is no interoperability)
- Audit Trail: reliable data sent over unreliable message exchange (e.g., standardized log messages sent over UDP syslog)
- Non-Repudiation: reliable data over reliable message exchange (e.g., standardized log messages sent over reliable protocol such as JMS)

Non-repudiation is often associated to the exchange of documents, in which case the relevant properties are: Definition from the book: *Secure Multiparty Non Repudiation Protocols and Applications*, J. Onieva, J. Lopez, J. Zhou, Advances in Information Security, Springer, 2008

- Non-Repudiation of Origin (NRO)
- Non-Repudiation of Submission (NRS)
- Non-Repudiation of Delivery (NRD)
- Non-Repudiation of Receipt (NRR)

Non-repudiation may obviously be associated to a much wider set of events than electronic delivery, e.g., "access to a document", "modification of an element". In general, non-repudiation is strictly related to the content of the audit log.

Non-repudiation properties are enforced by tokens, named *evidence*. The following tokens are listed:

- Non-Repudiation of Origin (NRO) protects against the originator's false denial of having originated the message. EVIDENCE OF ORIGIN (EOO) is generated by the originator or a TTP on its behalf and **will be held by the recipient**.
- Non-Repudiation of Receipt (NRR) is intended to protect against the recipient's false denial of having received the message. The recipient, or a TTP on its behalf generates EVIDENCE OF RECEIPT (EOR), and **the originator will hold it**.
- Non-Repudiation of Submission (NRS) it is intended to provide evidence that the originator submitted the message for delivery. The delivery agent generates EVIDENCE OF SUBMISSION (EOS) and **the originator will hold it**.
- Non-Repudiation of Delivery (NRD) is intended to provide evidence that the message has been delivered to the recipient. The delivery agent generates EVIDENCE OF DELIVERY (EOD) and the originator will hold it. Similarly, we should be aware that evidence provided by this service couldn't be used to make further deductions about the delivery status without some sort of assumptions in the communication channel.

It is worth noticing that in case of the presence of a delivery agent, this can act as a TTP, and it can create EOO and EOR on behalf of the originator and recipient, respectively.

It is worth noticing that NRO+NRR is in the *direct model* (e.g., in SOA), while NRS, and NRD requires a delivery agent in an *indirect model*. For instance, in SOA models like ESBs, such properties as NRS and NRD may be required. Delivery agents can be classified into

- Transparent, where the delivery agent is performing actions (like semantic mappings, SMTP)
- Non-transparent, like, in Enterprise Service Bus (ESB)-to-ESB communications.

Evidence is used to solve disputes. They are usually implemented by means of *Digital Signatures* (with Qualified Certificates) or *Secure Envelopes*. A TTP can be e.g., a CA, a *Time Stamp Authority* (TSA), a *notary service* (providing evidence on behalf of an actor) or an *Adjudicator* (drives the dispute resolution process). When the TTP is offline (e.g., services of the TTP are used for dispute resolution) the class of non-repudiation protocols are named *Optimistic*. It is worth noticing that protocols exist where the TTP **may** misbehave c.f. M. K. Franklin, *Fair Exchange with semi-trusted third party*, CCS 97 due to an external compromise.

For this ABB, common requirements are as follows:

Fairness: each party holds the expected items at the end of the exchange process. Asokan c.f. Asokan, N. 1998. Fairness in electronic commerce. Ph. D. thesis, University of Waterloo. defined the following:

- **Strong** fairness, when the exchange is completed, sender A can prove to an arbitrator that recipient B has received (or still can receive) the item, *without any further need of cooperation from either B or any trusted third party*. In other words, there is no need for any further proof. For instance, using a protocol only based on digital signatures of messages is *not fair*. In case of disputes, one party may refuse to provide its signed nonrepudiation token to a judge
- **Weak** fairness, when exchange completed, A can prove that B has received (or still can receive) the item, or otherwise an affidavit can be presented to demonstrate that B misbehaved, or a network failure occurred. We need to have a judge
- **Eventually Strong:** fairness is ensured within the system but with the provision that additional assumptions about the participating parties are made.

Efficiency (or Effectiveness): if no error occurs, the TTP should not intervene

Timeliness If a transaction has been terminated (or delayed) parties will know the status of the transaction eventually.

The choice of which requirement to use, is left to the system architect, based on a risk analysis, and a Business Impact Analysis.

Several industry standards exist to implement Non-Repudiation. ETSI REM approach and the ISO 13888. The ISO 13888 (SC27) is a specification divided in the following chapters:

- 13888-1 General Definitions
- 13888-2 Non-Repudiation mechanisms using symmetric techniques
- 13888-3 Non-Repudiation mechanisms using asymmetric techniques
- 13888-4 Non-Repudiation policies
- 27037 Guidelines for identification, collection, acquisition, and preservation of digital evidence
- 27042 Guidelines for the analysis and interpretation of digital evidence (also involving chain of custodies)

The ABB is modeled as follows.

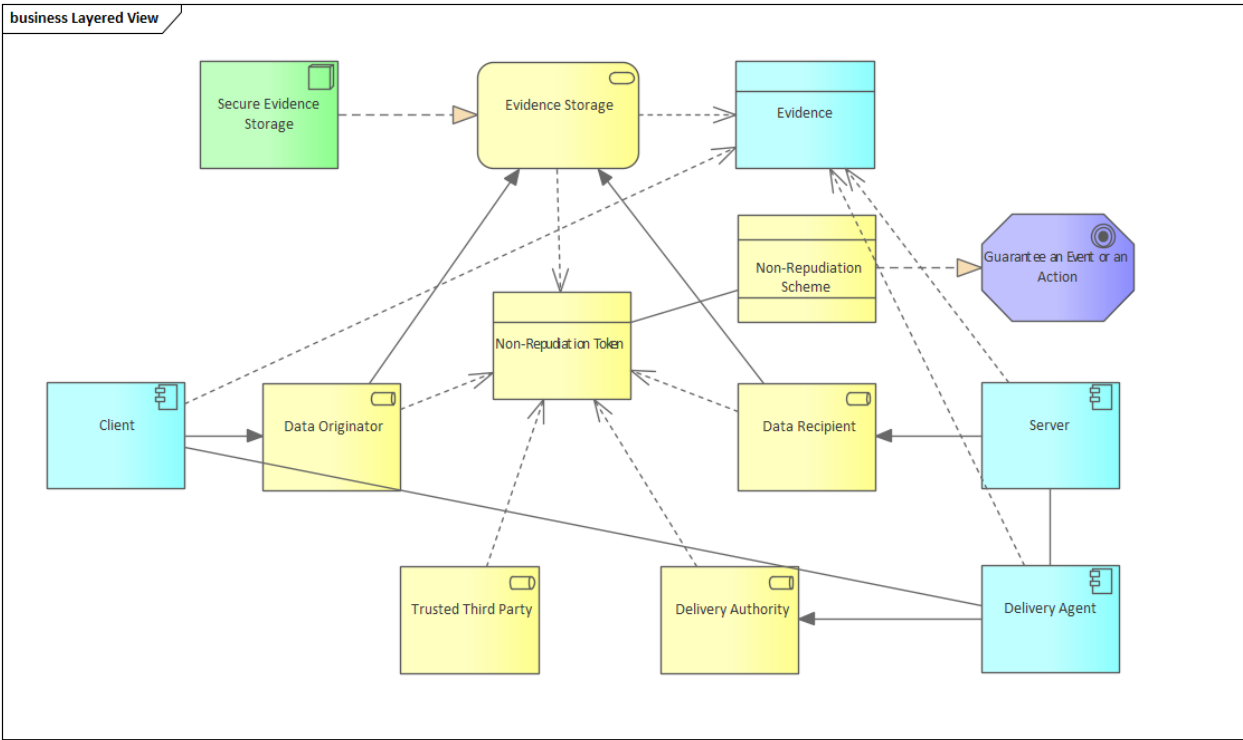


Figure 14 Non-Repudiation ABB: archimate model

A *Data Originator* needs to exchange data with a recipient. Once defined a *Non-Repudiation Scheme*, which is involving the decisions on which fairness property to use, and which token and format. Once decided the *Non-Repudiation Token*, this is exchanged according to the scheme, stored in the *Evidence Storage*, in the form of *Evidence*. Notably the evidence storage should be *tamper-proof*. Eventually, depending on the scheme chosen, a *Trusted Third Party* is appointed. If the delivery mode is not direct (e.g., it is not using a channel established using the NODEABB), a *delivery authority* and its corresponding *agent* is involved.

A non-repudiation token SHALL be defined as follows:

$$NRT = text1 || z1 || Sa(z1)$$

Where $Sa(z1) = sign\{Pol || fNR || A || B || TTP || T_{\{issue\}} || T_{\{action\}} || Q || SHA(m)\}$. In other words, it shall provide an eventual Policy, a function indicating which NR token is used (origin, submission, etc), the sender, A, the receiver, B, the eventual trusted third party, the time of issuing the token, and the time of the action (either originating, submission, or delivery). Q are eventual other data, and the SHA is the application of SHA256 to the entire message (e.g., the signature of the message, see DIABB).

Tag
Non-Repudiation

Generic Requirements.

Brief name	Description	Variability Point
------------	-------------	-------------------

NRABB-1-T	A Non-Repudiation Evidence SHOULD be emitted as the subsequent operation of the action	Risk analysis should define if non-repudiation is needed, or an audit trail is sufficient
NRABB-2-T	Evidence MAY be signed	Hashing may be sufficient, in case the evidence storage is within the same trust actor
NRABB-3-T	The evidence storage SHALL guarantee the persistence of the data	
NRABB-4-T	The evidence storage SHOULD be tamper-proof	
NRABB-5-T	The non-repudiation token SHOULD contain the information defined in this ABB	

Views

Business Process View

The Non-Repudiation is composed by several role and business objects. The roles are the following:

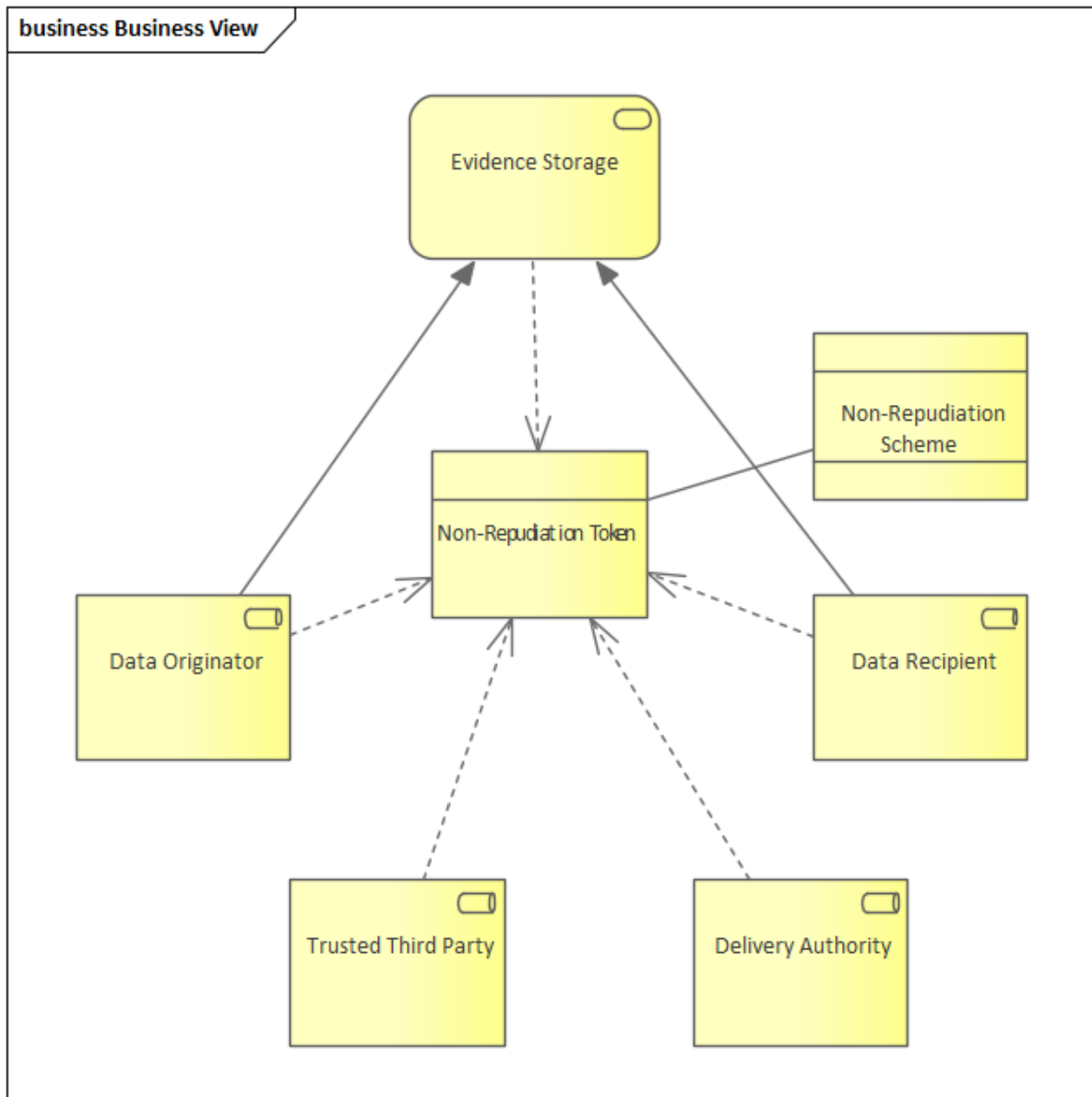
- Data Originator: who is creating or modifying (e.g., manipulating) data
- Data Recipient: who is receiving data and in the need to evaluate non-repudiation
- Delivery Authority: the software agent who is delivering the data, from the originator(s) to the recipient(s). In fact, recipients can be multiple, as a potential infinite list of recipients can manipulate the data
- Evidence Storage: the entity who is responsible to store the evidence and it is known to all the stakeholders.
- Trusted Third Party: the entity who is trusted by all the stakeholders and it is responsible for preparing non-repudiation metadata

The view also covers a contract (the Non-Repudiation Scheme) and a data object, the Token.

It is required that recipient and originator agree on the same trusted third party by defining a common policy in which the trust relationships are expressed. In case of the presence of delivery authorities, a specific trust policy shall be defined. In fact, potential modification to data can be done also by the delivery authorities, acting as data recipients. Moreover, a policy governing the evidence storage is required: how data in transit is protected, how data arrives to the evidence storage (IPC, or via networked services), and which level of guarantee is required. This latter point is important: then the token arrives to the recipient, what is the intended behavior, in case the storage is unavailable? Should the transaction be halted, since it is not possible to deliver to the storage, or the transaction proceeds, risking to not have non-repudiation evidence afterwards? This aspect should be defined.

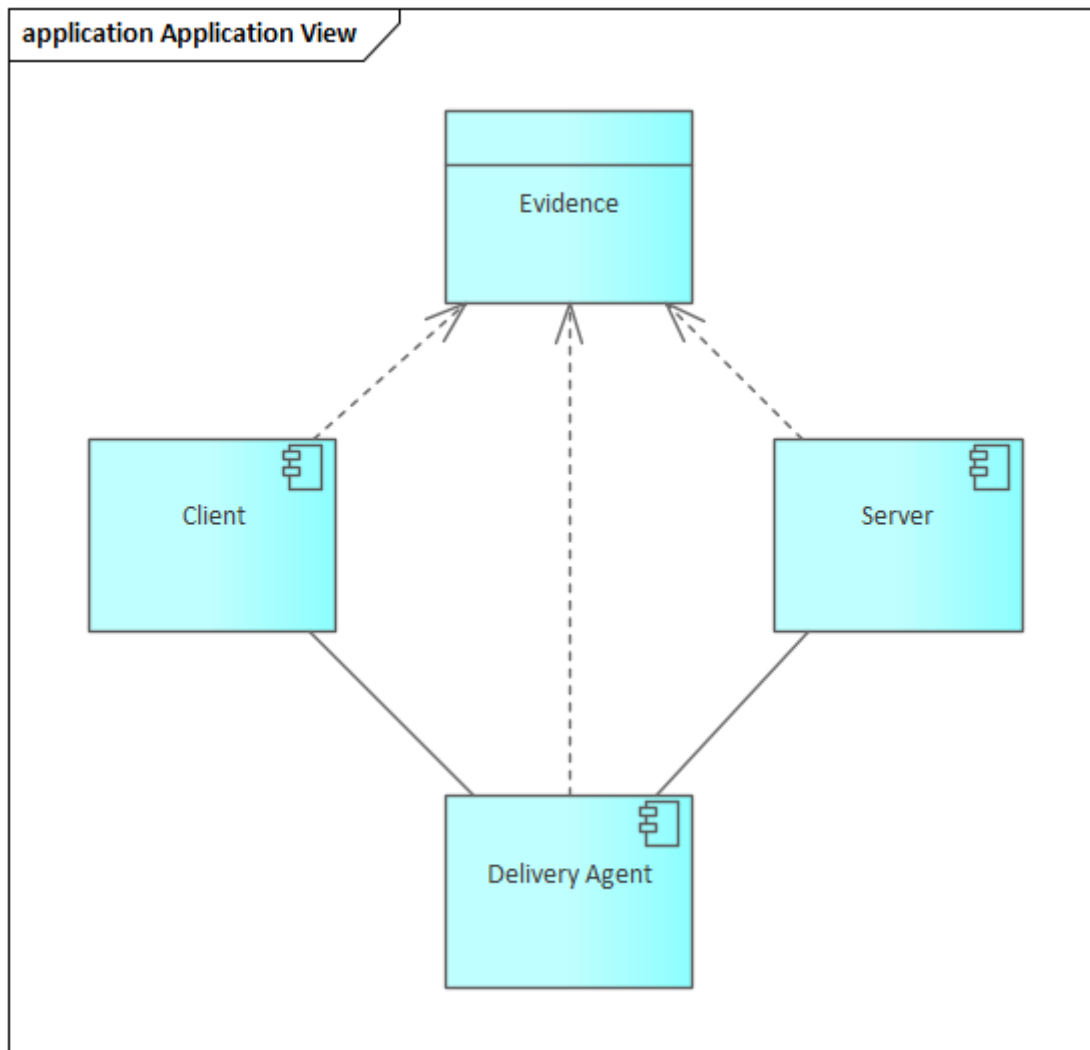
It is worth noting that Evidence Storage is a meta-actor: evidence can travel along the same message (e.g., as in CEF eDelivery), or in response to a message request, depending on the Message Delivery Pattern.

Evidence Storage may also be an immutable object, in case the evidence is requiring long-term storage. Use of distributed ledgers, or tamper-proof systems may be considered.



Application View

The application view of Non-Repudiation defines the three actors, Client, Server, and (eventual) delivery agent, persisting the Evidence data object. Potentially, both client and server should contain key material to sign the Evidence. In this case, the provisions in key material protection of the DIABB applies.



Technology and Motivation View

The technology and motivation view do not have security considerations.

Related ABB

[Consistent Time ABB] to provide synchronized clocks, needed by all cryptographic algorithms.

Threat Mitigation

Using non-repudiation log events which are of utmost importance are safely recorded. Using this building block the following design flaws are implemented:

- CWE-223: Omission of Security-relevant Information
- CWE-778: Insufficient Logging

Controls Implemented

ISO 27001:2022 A.8.15

IBB realization

N/A. IBB realization will be provided in future versions of this document.

Done Checklist

This building block is implemented when the policies defined in the business process view are published and agreed among all the stakeholders.

Contributors

Massimiliano Masi <mmasi@autostrade.it>

History

March 14, 2023, First Release

Audit Trail ABB

Name

Audit Trail ABB - ATABB

Rationale

Each application has specific and known states which are relevant for cybersecurity. For instance, administrator's login, failure in verifying a certificate, are states which could trigger further investigation for potential cybersecurity incidents. Usually, to perform such analysis, a Security Operation Centre (a SOC) is contacted. SOC's use tools named Security Information and Event Management, SIEM, a collector of log events coming from disparate sources. The way how SIEM works is correlating log entries coming from applications. Each log entry is parsed by the SIEM when it is instructed. Thus, a potential parsing problem or an unknown log entry, may hinder the incident recognition. Moreover, if the data from the log source is tampered with, or it is not reliable, it is impossible to perform forensic analysis. However, the application flow should continue even if the logging transport is unavailable.

To avoid these shortcomings, audit trails are defined. By using unreliable mechanisms, audit trails provide a defined syntax and semantics for the logged data. Having such a syntax allows the creation of parsers already provided by the vendors, while semantics enable forensic analysis. As example, an audit trail for administrator's login may contain the username of the admin, and crypto-verifiable information guaranteeing the proof of evidence.

The following events SHALL be logged²³:

- Physical Access
- Access Control (login/logout/failed login)
- Backup and Restore
- Configuration Change
- Potential Reconnaissance

²³ https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html



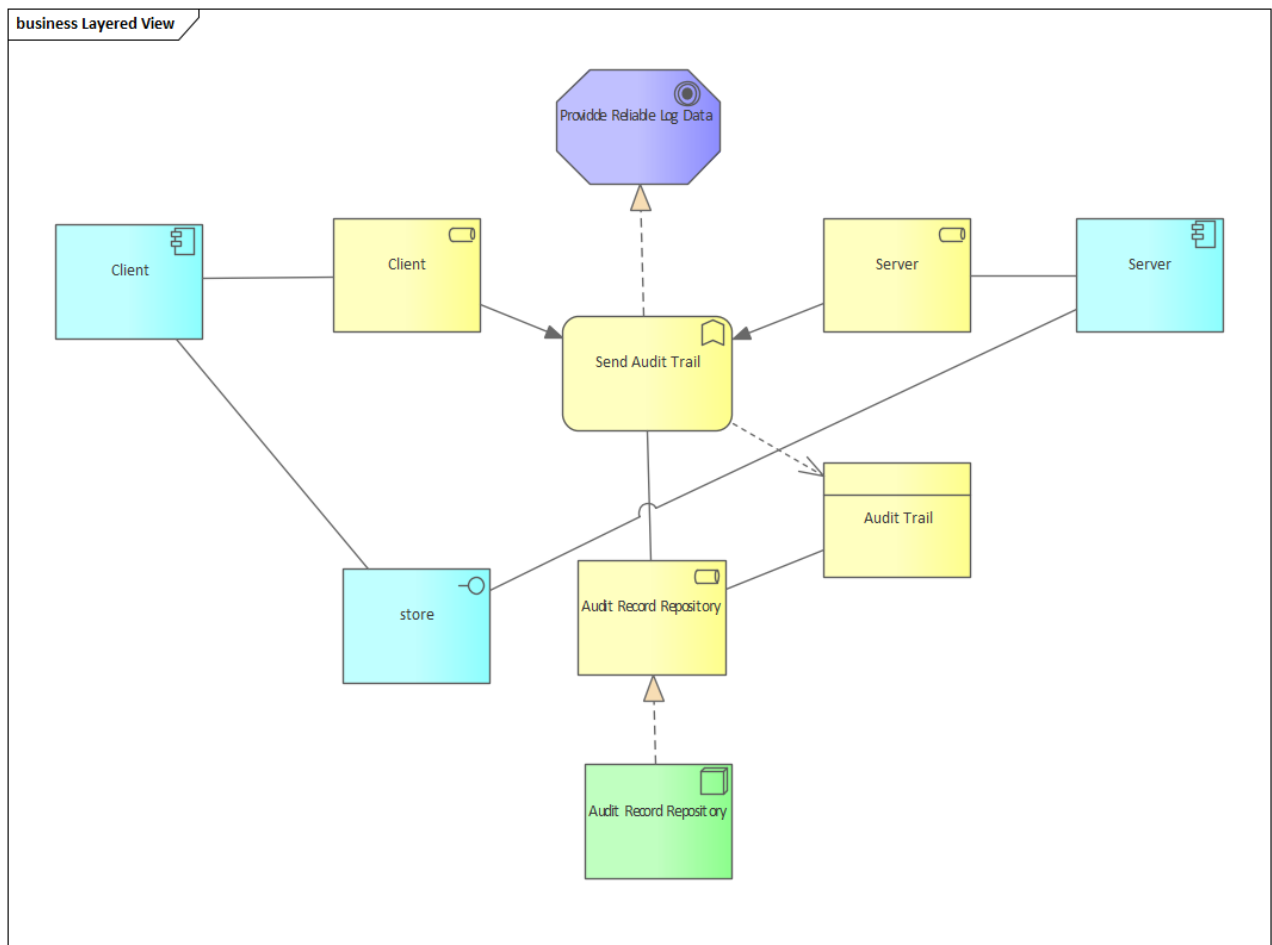
- Log system events
- TLS Error
- Rekeying
- Session Management Failures

Each log shall contain the following attributes

- When, date and time in UTC of the log and of the event, in format ISO 8601
- Where, application identification, host IP, dns name, eventual Subject Alternate Name of the X.509 certificate if NODEABB is used. For both source and destination
- Who, the principal performing the action, the identity, and where to prove the existence of the authentication (with link to the identity provider, if available)
- What, the type of event, the outcome (SUCCESS, FAILURE, MAJOR FAILURE) on which resource, with which severity (EMERGENCY, ALERT, INFO)

It is mandatory that the following data SHALL be excluded (even from log events):

- Application source code
- Session identification values
- Access Tokens
- Personal data and personal information
- Passwords
- Connection Strings
- Encryption Keys and Master Secrets
- Bank Account or Payment Card
- Classified Data



Tag
Logging

Generic Requirements.

Brief name	Description	Variability Point
ATABB-1-T	A logging system SHALL NOT log sensitive data	
ATABB-2-T	Audit Trails SHALL NOT be signed	
ATABB-3-T	A system SHOULD provide correlation rules	

Views

Business Process View

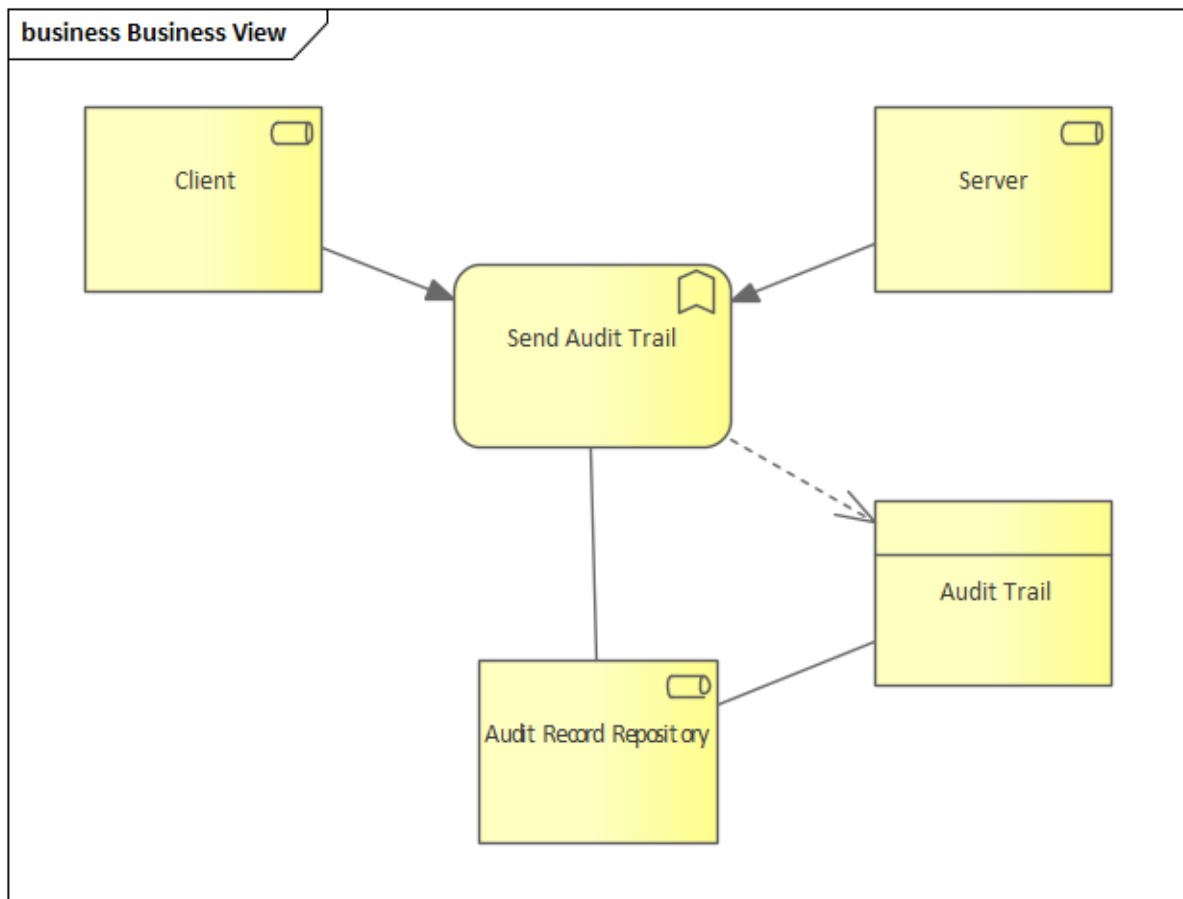
The business process view is composed of a business function that is sending audit trail, shared by the client, who is in charge of sending the audit trail after sending the transaction, and the server, who is



in charge of sending the audit trail after processing the transaction. Audit trails are stored in an Audit Record Repository.

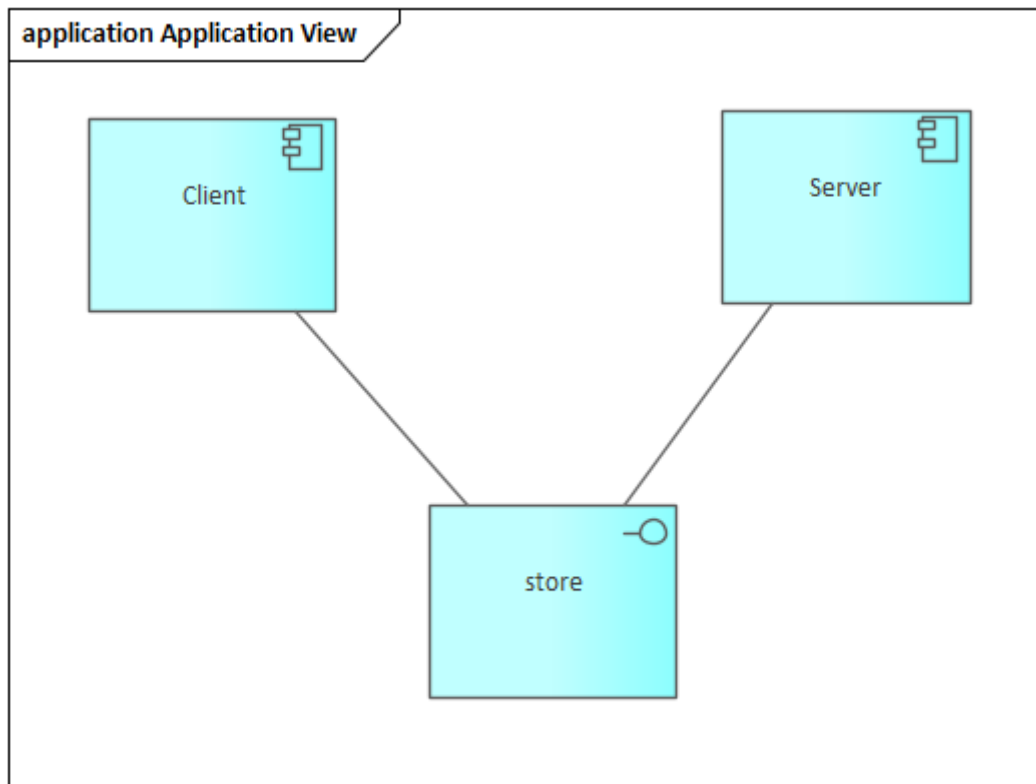
In order to implement successfully this building block, a policy of log retention shall be in place. To attain interoperability, the format of the audit trail shall be correctly defined and agreed among the client and the sender, so the same information (semantic-wise and trust-wise) is available for eventual forensic use.

The audit record repository should be regulated by access policies: access to the repository shall be audited as well.



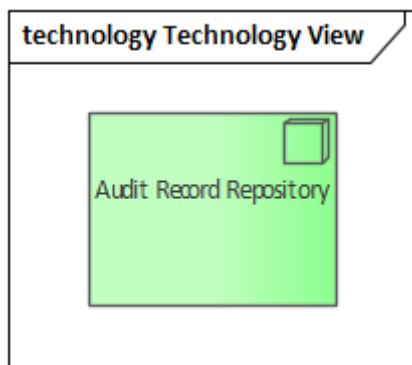
Application View

Similarly, the application view is composed by a server and a client who are sharing a store. Notably this store is not required to be a single point of contact, e.g., shared uniquely among the Clients and Servers. As the audit trails are defined to be interoperable, using the same syntax, semantics, and data source, it can be obviously distributed in different security domains, as long as the governing policies are shared accordingly.



Technology View

The Audit Record Repository shall be protected in best effort by unintentional access. The usage of privileged access solution is strongly suggested, to record the activity of system administrators, according with the EU 2016/679.



Related ABB

[Consistent Time ABB] to provide synchronized clocks, needed by all cryptographic algorithms and for the timestamp in the audits.

Threat Mitigation

Mitre ATT&CK, T1070: Indicator Removal : Automatically forward events to a log server or data repository to prevent conditions in which the adversary can locate and manipulate data on the local system. When possible, minimize time delay on event reporting to avoid prolonged storage on the local system.

Controls Implemented

ISO 27001:2022, A.8.15 Logging

IBB realization

[Audit Syntax IBB] Defining the syntax of the audit trail.

Done Checklist

This building block is correctly implemented if all the policies defined in the views are implemented and made available. In particular:

- It has been defined a log format
- It has been agreed when to trigger the audit trail
- It has been agreed where the logs are stored
- It has been agreed who can access the logs.

Contributors

Massimiliano Masi <mmasi@autostrade.it>

History

March 14, 2024, First Release.

Data Classification ABB

Name

Data Classification ABB

Rationale

Data classification is a practice that gives to a data, information, or a generic hardware or software asset, a label, indicating its sensitivity, and it is part of its lifecycle. In fact, a particular asset can be labelled initially, then reclassified, removed, etc. Different labels may end up in different policies for the access, treatment and safe cancellation of the asset. It is an instrument to define the degree of protection to be implemented. There are several techniques to perform data classification.

Ideally, all data shall be classified in clusters, to assist access control engines to ensure that sensitive information is shared with the appropriate audiences. Examples of data classification is the Traffic Light Protocol (TLP)²⁴. TLP is a four-dimensional approach to data classification. It starts with the highest level, the Red, indicating that labelled data shall be restricted to a group of users. Then, the Amber+Strict indicates that the data can be disclosed to an organization, which is stricter than Amber, that includes also the clients. The last two, Green and White indicates that the information can be delivered to the community or to the public. The protocol is depicted in Figure 14 The TLP protocol.

Similar approaches can have similar labels with similar semantics such as *confidential*, *restricted*, *internal*, and *public*.

²⁴ See, <https://www.cisa.gov/news-events/news/traffic-light-protocol-tlp-definitions-and-usage>



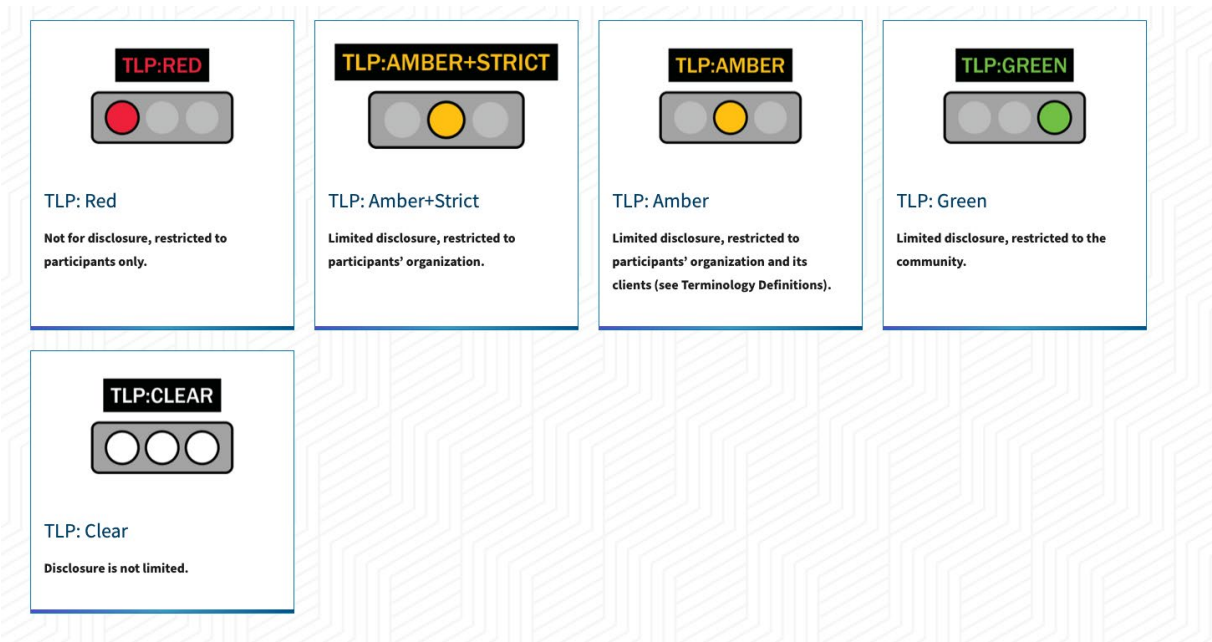


Figure 15 The TLP protocol

Classification of information may also be used to provide quality of services, in the context of critical infrastructures where availability is of utmost importance. Certain labels may be treated by specific cybersecurity aspects pushing availability of the information despite of its confidentiality and integrity.

In particular the Commission Delegated Regulation (EU) No 886/2013, Art. 3, defines a list of Road Safety-Related events or conditions (SRT). Although the data is public in nature, the label may suggest the said specific treatment by e.g., prioritizing the communication channel, or vulnerability assessment, to guarantee that the SRTI is delivered to the stakeholders.

Notably, different schemes can be used. Whenever there is the need to interoperate between stakeholders, the TLP protocol SHOULD be used to achieve interoperability.

This building block provides a five-dimensional approach to data classification, enriching the TLP with SRTI dimension. As example, the data “obstacles on the road” can be labelled “TLP:RED:SRTI” when the information firstly arrives from either a sensor or a notification by the user, i.e., it is not yet verified. Subsequently, when the information is confirmed, can be relabelled “TLP:GREEN:SRTI”, indicating that the information shall be routed only through highly trusted devices.

Another example is a data containing contractual information about a data set exchanged through the NAP. This information is classified TLP:RED unless the contract becomes publicly available.

Tag
Data Classification

Generic Requirements.

Brief name	Description	Variability Point
DCABB-1-T	Data SHALL be classified using a labelling scheme.	

DCABB-2-T	The TLP Protocol SHOULD be used to classify data	If the organization already have a classification scheme, this can be used.
DCABB-3-T	The concept of SRTI SHOULD be used when labelling the information	

Views

This is an organizational building block, thus no architectural approach is provided.

Related ABB

N/A

Threat Mitigation

Mitre ATT&CK ID M1057, Data Loss Prevention, which covers threats such as Data Exfiltration.

Controls Implemented

ISO 27001:2022 A.5.13: Labelling of Information

IBB realization

n/a

Done Checklist

This building block is implemented if the classification scheme is defined and all data is classified accordingly.

Contributors

Massimiliano Masi mmasi@autostrade.it

History

March 14, 2024, First Release

Consistent Time ABB

Name

CTABB

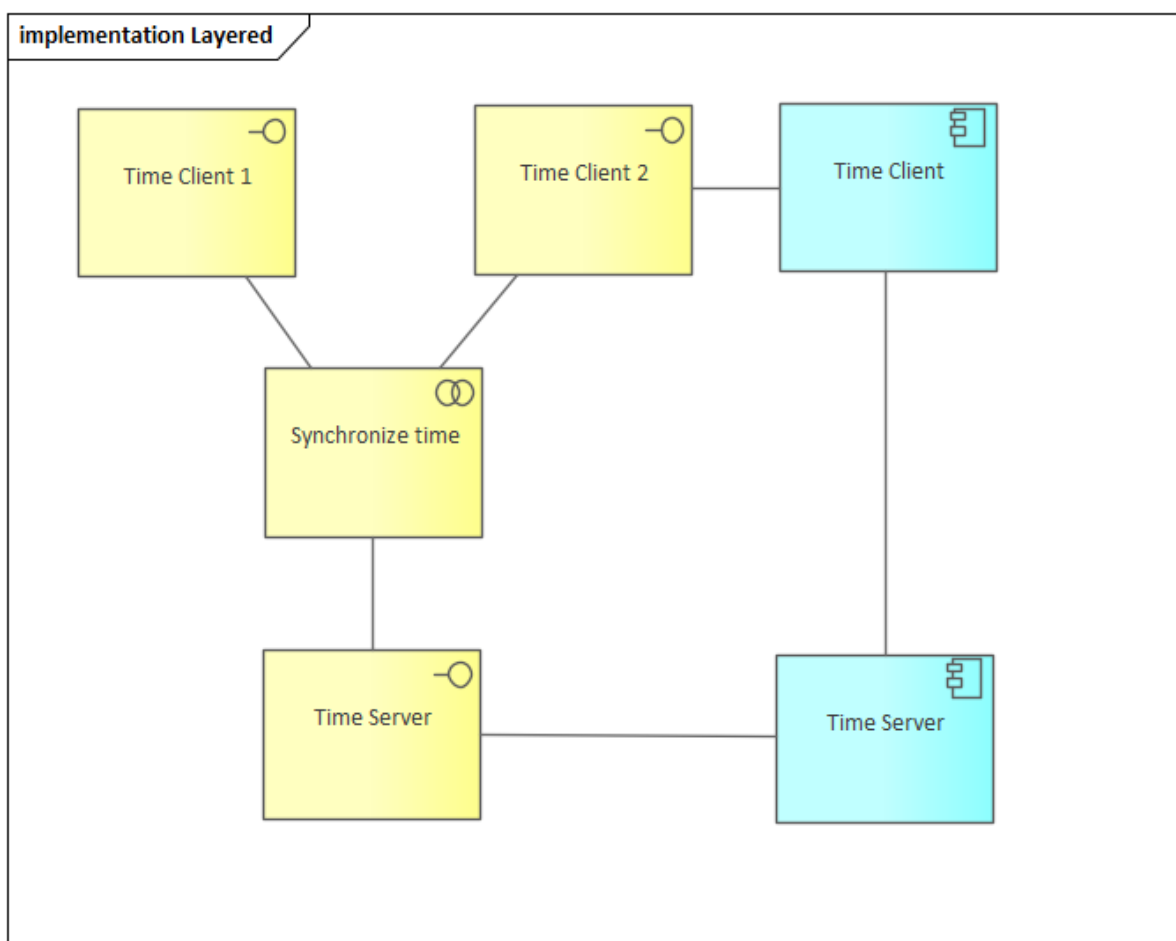
Rationale

The consistent time ABB provides a means to ensure that the system clocks and time stamps of the many computers in a system exchange are well synchronized. This ABB specifies a synchronization with a median error less than 1 second.

In most of the cybersecurity protocols, the clock synchronization is essential to enjoy the freshness of the messages. For instance, sessions, authentication or authorization tokens are bound to time limit, and if this is not enforced by the stakeholders, an attacker can gain access to the resources. This building block defines the use of the Network Time Protocol (NTP) defined in rfc 5905 (NTPv4).

In NTP, trust is flowing from the primary servers through secondary servers up to the leaf entities. The attacker model defined in the rfc assumes that the attacker can perform active attacks on the network.





The system is composed by two devices, a Time Client and a Time Server. Business-wise, time clients (e.g., the NAPs) agree with one (or eventual more) time servers to synchronize time. Note that the time server is a logical component, all synchronized with root nodes, so there is no restriction on using a single time server.

Tag

Clock Synchronization

Generic Requirements

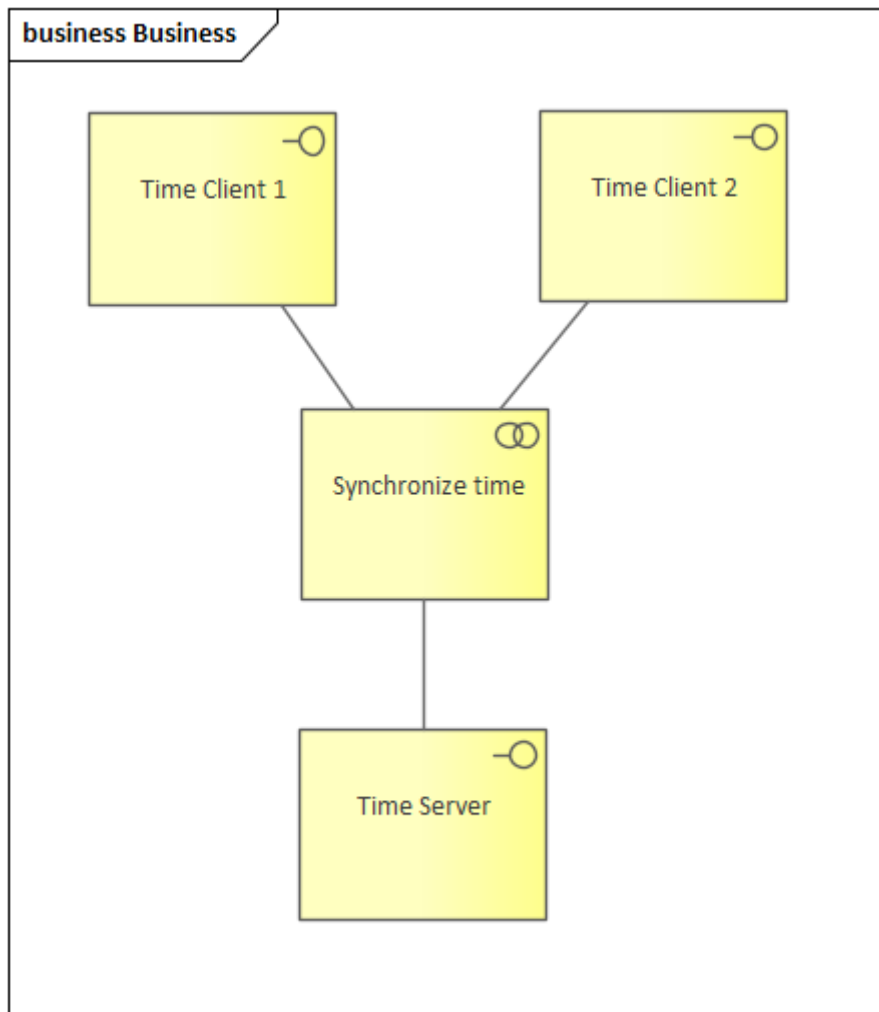
Brief name	Description	Variability Point
CTABB-1-T	Each system SHALL be synchronized by using a time server using the NTP protocol	

Views

Business Process View

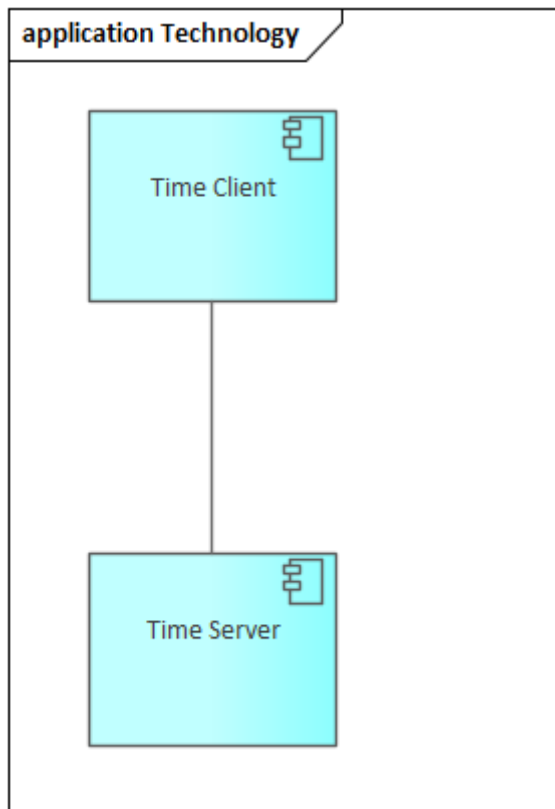
The Business Process View is composed by time clients and time servers that agree on time synchronization. This building block prescribes the use of NTPv4 as business interaction.





Application View

The application view is composed of two generic applications, logically grouped with any service running in the system. Usually, the NTP is configured by the underlying operating system or firmware. Devices with no NTP support SHALL not be used, and the implementation on an application level is discouraged. Frequent checks to eventual charging batteries on IoT sensors not connected to the energy network should be done, to avoid drifts.



Related ABB

All the ABBs defined in these specifications SHALL make use of this Consistent Time.

Threat Mitigation

The Consistent Time is a foundational building block. The use of Clock synchronization mitigates all the threats related to the need of cryptography.

Controls Implemented

ISO 27002:2022 A.8.17 Clock Synchronization.

IBB realization

n/a, as the technical specifications are defined in this ABB

Done Checklist

The building block is correctly implemented if the systems are synchronized with a time server synchronized with a root server.

Contributors

Massimiliano Masi mmasi@autostrade.it

History

March 14, 2024, First Release

Architecture Repository – IBB

TLS IBB



Name

TLS IBB - TLSIBB

Rationale

The TLS IBB specifies foundational components that are part of an overall privacy and security system. These are:

- Node Authentication
- Access Control
- Secure Communications

Successful implementation of TLS IBB also depends on the existence and support of:

- System Security Services
- Privacy and Security Governance.

TLS IBB enables communications participants to:

- Confirm that the server is indeed the authorized server system (default).
- Confirm that the client is indeed an authorized client (in case of mutual recognition of both certificates, mTLS) .

This enables the use of system or machine-level access controls that limit access to only authorized and authenticated machines. The local governance policies will determine whether machine level access control rules are used.

TLS IBB requires user **AccessControl**. User Access Control determines whether the user has access to information or system services. TLS IBB also requires that some form of user authentication be performed. It allows the system and deployment to choose an appropriate method, but all users shall be identified and authenticated. It uses these identities in the event audit log to identify users. It requires that access control use these identities (and other information) to determine what data and services are available to that user. Other system security services may also use the user identities.

Secure Communications are provided using TLS. TLS provides mutual authentication, reliable message transport and private communication through data encryption and integrity protections. Different forms of encryption can be negotiated to protect the data in transit. TLS IBB does not permit the negotiation of no encryption to accommodate sites that prefer to use a different form of protection.

TLS IBB does not restrict implementations and deployments to only use the TLS IBB specified methods. For interoperability reasons, TLS must be implemented and available to be configured. The RFC 9325 “Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)” covers many configuration options. Deployments often follow these recommendations and make them part of their security policies. A deployment’s security analysis may lead to different choices. Therefore, it is important that implementations allow configuring different protocol versions, algorithms, etc., that SHALL not weaken the definition of this IBB and related standards. Following the definition of the rfc 9325, TLS versions prior of 1.2 SHALL NOT be negotiated, or evaluated on a risk-basis. In particular, the reader is pointed to the evaluation of known attacks of weak TLS implementations as pointed in rfc 7457.



Certificate validation is an essential part of the authentication of a TLS channel. Although essential checks are implemented by libraries, the reader should be familiar with the common provisions²⁵.

Requirements

Table 1

Brief name	Description	Variability Point
TLSIBB-1-T	The secure node SHALL provide secure access to data only through TLS	
TLSIBB-2-T	The secure application SHALL provide secure interfaces to access data only through TLS.	Other access MAY exists (e.g., system administrators through either local or remote connection)
TLSIBB-3-T	The validity of all the certificates SHOULD be checked against a Certificate Revocation List (CRL) or via Online Certificate Status Protocol (OCSP) as indicated in the certificate	

Similar IBB and Dependencies

Similar IBB:

[WS-IBB]. This building block provides an alternative to TLS IBB for Web-Service communications, where the intermediary may change parts of a message in its way.

Dependencies:

[Certificate Pinning IBB] to provide means to enforce out-of-band authentication of remote certificates exchanged during the establishment of a secure channel

Certificate Pinning IBB SHOULD be evaluated when designing the architecture. In fact, based on the pinning, many decisions can be taken. As example:

- *TLS non mutually authenticated, and no certificate pin made by the client.* In this case an attacker can impersonate the server by obtaining a certificate from the CA. To mitigate this risk, when owning a certificate authority, a registration process procedure SHOULD be available. Instead, when using public certificate authorities (even eIDAS compliant), a certificate check over a certificate revocation list SHOULD be performed.
- *TLS mutually authenticated, and no certificate pin made by either the server or the client.* In this case, an attacker can impersonate either the server or the client using the same technique as in the previous scenario. Although mutual authentication (known also as mTLS) improves significantly the authentication and confidentiality of the communication channel, a policy

²⁵ See, e.g., https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html

governing certificate provisioning SHOULD be available and monitored. An example of such policy can be found in the Certificate Policy of the C-ITS Point of Contact²⁶

- *TLS mutually authenticated with certificate pinning at both sides.* In this case the scenario in which the attacker can impersonate one of the protocol peers is increasing in complexity. On the other hand, using pinning also increases the administrative burden on managing certificates, thus a certificate renovation policy potentially assisted with automated tools SHOULD be available. Notably, in many Datex II scenarios, the endpoints are executed in a relatively fixed infrastructure, therefore this option is RECOMMENDED for static machine-to-machine communications.

Additional reasonings will be shown in the next sections.

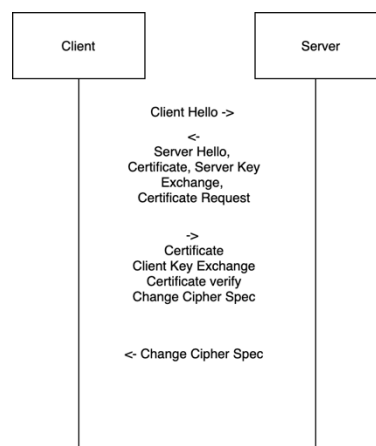
Known Use

N/A. Known uses will be available in a next version of this deliverable.

Technical Implementation:

There are several versions of TLS. This building block use the versions 1.2 for legacy applications, and 1.3 as defined in rfc8446. Implementing TLS is complex²⁷, and a broken TLS implementation may lead to attacks with huge impact²⁸. TLS implementations are heavily dependent on off-the-shelf libraries. Ad-Hoc TLS implementations are discouraged, and they should be evaluated on a risk-basis.

TLS is a protocol aimed at protecting an exchange of a symmetric key to be used to provide confidentiality and integrity to a communication channel over TCP/IP sockets. Such exchange is governed by an initial part, called *handshake*, which use a key management scheme to protect the key derivation and exchange. The protocol (in its versions prior to 1.3) is shown in the picture below.



²⁶ See, e.g., <https://cpoc.jrc.ec.europa.eu/>

²⁷ See, e.g., <https://www.darkreading.com/risk/researcher-poor-ssl-implementations-leave-many-sites-at-risk> (however we should find a better reference).

²⁸ See, e.g., SSL Stripping, SSL Hijacking.

As a prerequisite, a client and a server own a key pair each. Such keypair is a private key, named K_C^- , and a public key, named K_C^+ . The private key is a secret and it is protected from read from other users. Java is protecting the keys by encrypting it in a file named *keystore*. Public keys are signed by a certification authority, who is wrapping a public key in a data structure named X.509, and adding attributes. Each actor in the TLS may have its own CA. In this example, we assume that CA_C is the certification authority serving the client and CA_S is the certification authority serving the server. Public keys and certification authorities' certificates are public by definition. We assume that the client has access to the CA of the server and viceversa. In Java this means that the client is able to access a file, named *truststore* which contains at least CA_S .

Note: key attributes will not be specified in this document, however for TLS protocol, at least TLS Web Server, and TLS Web Client ExtendedKeyUsage attributes SHALL be present.

A client device starts the conversation in a TCP/IP socket by sending a first message, named **client hello**. The server receives the message and returns a **server hello** containing its certificate. The client, upon receiving the certificate, it starts its validation (e.g., checking if it is valid timewise, or if it has been revoked either using a Certificate Revocation List whose URL is defined inside a CA certificate's Distribution Point, or using Online Certificate Status Protocol, OCSP, whose URL is defined in the CA certificate). If the client is able to find, in its truststore, the CA who issued the certificate, then the client is *authenticating* the server: the client is possessing a cryptographic means to be sure that the server is the one supposed to be, but not viceversa: the server does not have any proof that the client is a known one. In order to verify the authenticity of the client, the server issues a **client certificate request** message. If the client answers, the server performs the same checks against the client's CA. If the certificate is valid, then the two endpoints are *mutually* authenticated²⁹. Then, using the key material provided exchanged, a key is shared, and a test message is sent. If both parties agree on the end of the handshake, the connection is authenticated and encrypted.

The algorithm used in the handshake are defined by a *ciphersuite*, which is negotiated during the handshake. A ciphersuite is similar to this string: TLS_RSA_WITH_AES_128_CBC_SHA256. Underscores separate the arguments. In this example, the arguments are as follows:

- TLS: defines the protocol
- RSA: describes the formats of the key pair
- AES, 128: indicates that the key that will be created is AES at 128 bit
- CBC: indicates Cipher Block Chaining
- SHA256 is the hashing mechanism

The use of TLS is bound to best practices. The use of the recommendations in rfc9325 MUST be used. In particular, for TLS 1.2 implementations, the following ciphersuite are RECOMMENDED:

²⁹ This is known as mTLS, or 2-way TLS.

- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

Note: RSA should be considered obsolete, and it will be removed soon. New implementation SHALL use EC scheme.

Notably, in the case of using TLS 1.3, there are additional restrictions. TLS 1.3 reduces the list of supported cipher suites to only the following:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

The first cipher suite (TLS_AES_128_GCM_SHA256) SHALL be supported (according to the dictates of the rfc 8446, Section 9). The protocol supports three basic key exchange modes, as follows.

- (EC)DHE (Diffie-Hellman over finite fields or elliptic curves)
- Pre-Shared Key Only
- Pre-Shared Key with (EC)DHE

The following text is obtained from Downing et al³⁰.

From a cryptographic perspective, major design changes in TLS 1.3 compared to version 1.2 include: (1) encrypting some handshake messages with an intermediate session key, to provide confidentiality of handshake data such as the client certificate; (2) signing the entire handshake transcript for authentication; (3) including hashes of handshake messages in a variety of key calculations; (4) using different keys to encrypt handshake messages and application data; (5) deprecating a variety of cryptographic algorithms (including RSA key transport, finite-field Diffie–Hellman key exchange, SHA-1, RC4, CBC mode, MAC-then-encrypt); (6) using modern authenticated encryption with associated data (AEAD) schemes for protecting application data; and (7) providing handshakes with fewer message flows to reduce latency.

There are two primary modes of the TLS 1.3 handshake protocol. One is the full, one round-trip time (1-RTT) handshake, which uses public-key certificates for server and (optionally) client authentication, and (elliptic curve) Diffie–Hellman ephemeral ((EC)DHE) key exchange, inspired by Krawczyk’s ‘SIGn-and-MAC’ (SIGMA) design [72]. Several session keys are established for a variety of purposes in this mode: to encrypt part of the handshake, to enable export of keying material to other applications, for session resumption, and of course to encrypt application data. This mode gets its name from the fact that application data can be sent from the client to the server with the handshake’s completion after a full round trip, meaning there is one round-trip time (1-RTT) until the first application message can be sent (not counting non-TLS networking operations such as DNS lookups or the TCP 3-way handshake).

³⁰ Downing, B., Fischlin, M., Günther, F. et al. A Cryptographic Analysis of the TLS 1.3 Handshake Protocol. J Cryptol 34, 37 (2021). <https://doi.org/10.1007/s00145-021-09384-1>



The other primary mode of the TLS 1.3 handshake protocol is the resumption or pre-shared key (PSK) mode, in which authentication is based on a symmetric pre-shared key, with optional (EC)DHE key exchange for forward secrecy; this generalizes the abbreviated session resumption handshake from earlier versions of TLS. The PSK mode can optionally be augmented with a zero round-trip time (0-RTT) key establishment, allowing the client to send—along with its first TLS flow—application data encrypted under a key derived from the PSK.

Although there are many examples of successful implementations of TLS 1.3, some early results³¹ indicates eventual risks on 0-RTT and PSK modes. This IBB does not prescribe any further constraint on the use of TLS 1.3. However, if PSKs are used, an audited procedure for the exchange of the pre-share key SHALL be in place.

Test Assertions and exploitation suggestions

TLSIBB-TA-1

Table 2

TA id	TLSIBB-TA-1
Normative Source:	TLSIBB-1-T, "the secure node SHALL provide secure access to data only through TLS"
Target	The SuC in its entirety
Prerequisite	The SuC is running in its final configuration (this is a requirement as it is a secure node), and all its ports and services are mapped and known
Prescription Level	Mandatory
Predicate	FOR EACH service, the connection is performed through the use of secure communication using the ciphersuite defined in this IBB. The tester should provide, for each service, the evidence according to the RAT chosen.
Variable	

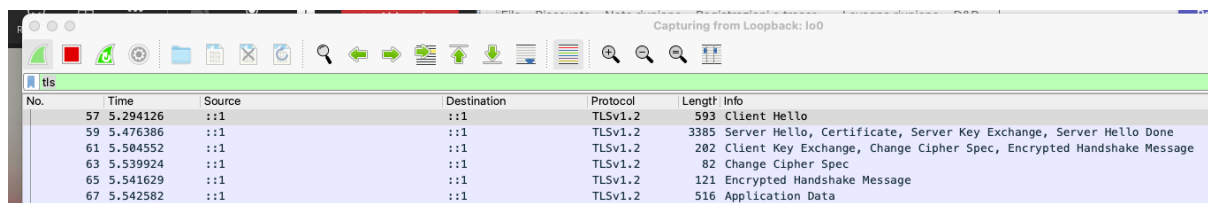
This TA has the aim to control that the Secure Node has controlled access to data. For Roadside or physically exposed equipment, a physical inspection of the system, and an eventual hardware security test to check for such debug port as JTAG or UART should be done.

In the lab testing environment, the following debug output should be shown, for each interface.

When running the system, having mutual authentication turned off, the following output, in wireshark³².

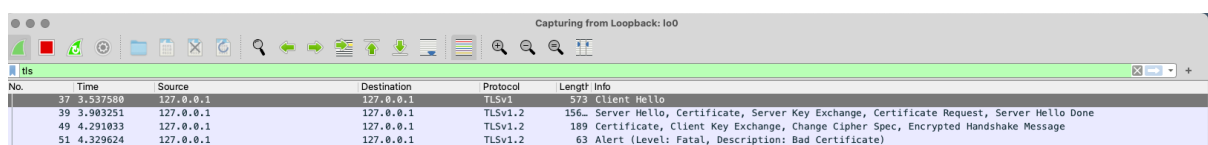
³¹ See M. Fischlin and F. Günther, "Replay Attacks on Zero Round-Trip Time: The Case of the TLS 1.3 Handshake Candidates," *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, Paris, France, 2017, pp. 60-75, doi: 10.1109/EuroSP.2017.18.

³² Notably, all the examples are made using TLS 1.2. For TLS 1.3, similar PCAPs are expected, *mutatis mutandis*.



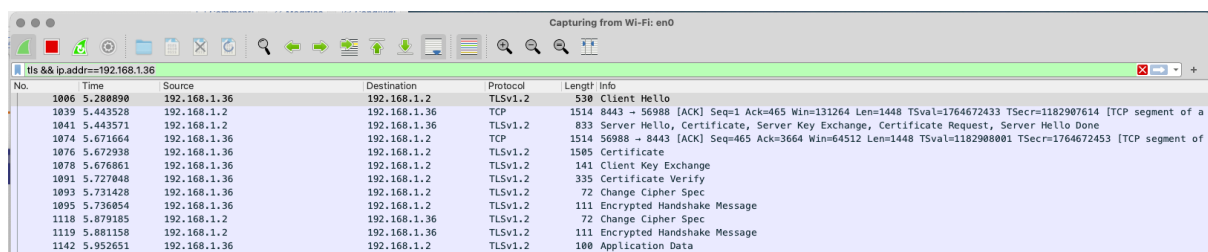
No.	Time	Source	Destination	Protocol	Length	Info
57	5.294126	:::1	:::1	TLSv1.2	593	Client Hello
59	5.476386	:::1	:::1	TLSv1.2	3385	Server Hello, Certificate, Server Key Exchange, Server Hello Done
61	5.504552	:::1	:::1	TLSv1.2	202	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
63	5.539924	:::1	:::1	TLSv1.2	82	Change Cipher Spec
65	5.541629	:::1	:::1	TLSv1.2	121	Encrypted Handshake Message
67	5.542582	:::1	:::1	TLSv1.2	516	Application Data

We see the client sending the first message, the `Client Hello`. The server answers with the values `Server Hello`, `Certificate` (which sends the server certificate), and `Server Key Exchange`, ending the message as `Server Hello Done`. The client then verifies the certificate in its truststore according with the BCP195 (check the expiration, checks if the certificate has a corresponding trust anchor, etc). After that, the keys are exchanged, and the `Application Data` can be sent. When turning on client authentication, when using the web browser, the errors is after the end of `Client Hello` with a `TLS Alert`.



No.	Time	Source	Destination	Protocol	Length	Info
37	3.537550	127.0.0.1	127.0.0.1	TLSv1	573	Client Hello
39	3.903251	127.0.0.1	127.0.0.1	TLSv1.2	156	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
49	4.291833	127.0.0.1	127.0.0.1	TLSv1.2	189	Certificate, Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
51	4.329624	127.0.0.1	127.0.0.1	TLSv1.2	63	Alert (Level: Fatal, Description: Bad Certificate)

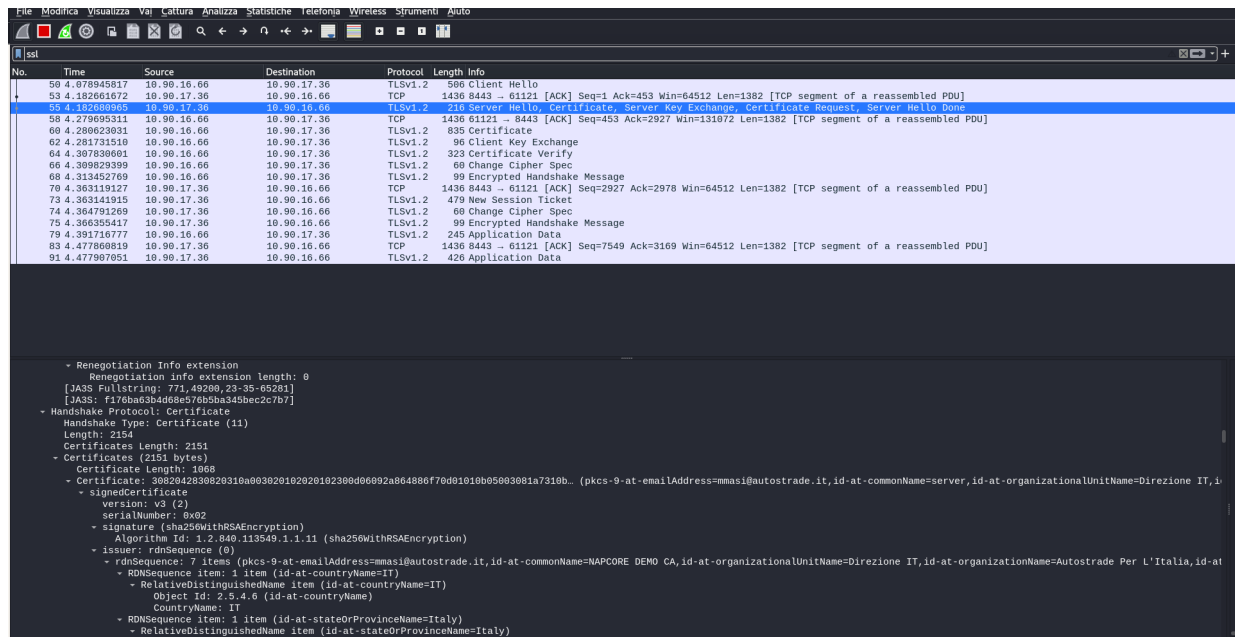
When starting the mutual authentication, the message flow is as follows.



No.	Time	Source	Destination	Protocol	Length	Info
1006	5.280890	192.168.1.36	192.168.1.2	TLSv1.2	530	Client Hello
1039	5.443528	192.168.1.2	192.168.1.36	TCP	1514	8443 → 56988 [ACK] Seq=1 Ack=465 Win=131264 Len=1448 TSval=1764672433 TSecr=1182907614 [TCP segment of a ...]
1041	5.443571	192.168.1.2	192.168.1.36	TLSv1.2	833	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
1074	5.671664	192.168.1.36	192.168.1.2	TCP	1514	56988 → 8443 [ACK] Seq=465 Ack=3664 Win=64512 Len=1448 TSval=1182908001 TSecr=1764672453 [TCP segment of a ...]
1076	5.672938	192.168.1.36	192.168.1.2	TLSv1.2	1505	Certificate
1078	5.676861	192.168.1.36	192.168.1.2	TLSv1.2	141	Client Key Exchange
1091	5.727040	192.168.1.36	192.168.1.2	TLSv1.2	335	Certificate Verify
1093	5.731428	192.168.1.36	192.168.1.2	TLSv1.2	72	Change Cipher Spec
1095	5.736054	192.168.1.36	192.168.1.2	TLSv1.2	111	Encrypted Handshake Message
1118	5.879185	192.168.1.2	192.168.1.36	TLSv1.2	72	Change Cipher Spec
1119	5.881158	192.168.1.2	192.168.1.36	TLSv1.2	111	Encrypted Handshake Message
1142	5.952651	192.168.1.36	192.168.1.2	TLSv1.2	100	Application Data

In this case, we see the entire protocol as described in the Technical Implementation section. It is worth noting that currently, the protocol after the handshake is totally encrypted.

The messages `Server Hello` and `Client Hello` contain the certificates, as shown in the following pictures. It is worth noting that the certificates SHALL be the one for the server and for the client.



TLSIBB-TA-2

Table 3

TA id	TLSIBB-TA-2
Normative Source:	TLSIBB-2-T, “The secure application SHALL provide secure interfaces to access data only through TLS.”
Target	The SuC in its entirety
Prerequisite	All the ports and services for the SuC are known and mapped
Prescription Level	Mandatory
Predicate	FOR EACH service, the connection is performed through the use of secure communication, using the ciphersuites defined in this IBB. The tester should provide, for each service, the evidence according to the RAT chosen
Variable	

This TA is like TLSIBB-TA-1 except that there is no need to check how the system is protecting the data from different method of access. The debug output is the same of TLSIBB-TA-1.

TLSIBB-TA-3

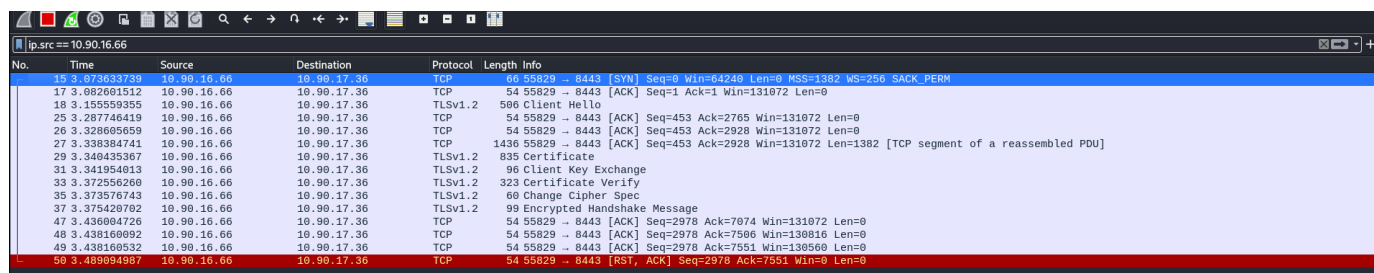
Table 4

TA id	TLSIBB-TA-3
Normative Source:	TLSIBB-2-T, “The validity of the certificate SHOULD be checked against a Certificate Revocation List



	(CRL) or via Online Certificate Status Protocol (OCSP) as indicated in the certificate.”
Target	The SuC in its entirety
Prerequisite	All the secure communication channels are known. A CA should have either a OCSP server or a distributed CRL, and its distribution points SHALL be part of the certificate
Prescription Level	Optional
Predicate	FOR EACH service, upon TLS handshaking, the certificate SHOULD be validated for revocation
Variable	

In this case, a failure in the communication has to be demonstrated. The tomcat has to be reconfigured to use the revoked certificate. The CRL has to be shared by the client, and the test is to prove that the client closes the connection upon verifying the certificate. In this case the overridden call `verify()` checks if the serial number of the certificate is listed in the crl file. The test is successful if in wireshark there is no application data message and the client calls a RST on the socket. Notably, this behaviour may change depending on the implementation.



No.	Time	Source	Destination	Protocol	Length	Info
15	3.972633739	10.90.16.66	10.90.17.36	TCP	60	55829 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1382 WS=256 SACK_PERM
17	3.982601512	10.90.16.66	10.90.17.36	TCP	54	55829 → 8443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
18	3.155559355	10.90.16.66	10.90.17.36	TLSv1.2	506	Client Hello
25	3.287746419	10.90.16.66	10.90.17.36	TCP	54	55829 → 8443 [ACK] Seq=453 Ack=2765 Win=131072 Len=0
26	3.328605659	10.90.16.66	10.90.17.36	TCP	54	55829 → 8443 [ACK] Seq=453 Ack=2928 Win=131072 Len=0
27	3.338384741	10.90.16.66	10.90.17.36	TCP	1436	55829 → 8443 [ACK] Seq=453 Ack=2928 Win=131072 Len=1382 [TCP segment of a reassembled PDU]
29	3.340435367	10.90.16.66	10.90.17.36	TLSv1.2	835	Certificate
31	3.341954013	10.90.16.66	10.90.17.36	TLSv1.2	96	Client Key Exchange
33	3.372556260	10.90.16.66	10.90.17.36	TLSv1.2	323	Certificate Verify
35	3.373576743	10.90.16.66	10.90.17.36	TLSv1.2	68	Change Cipher Spec
37	3.375420702	10.90.16.66	10.90.17.36	TLSv1.2	99	Encrypted Handshake Message
47	3.436004726	10.90.16.66	10.90.17.36	TCP	54	55829 → 8443 [ACK] Seq=2978 Ack=7074 Win=131072 Len=0
48	3.438160092	10.90.16.66	10.90.17.36	TCP	54	55829 → 8443 [ACK] Seq=2978 Ack=7506 Win=130816 Len=0
49	3.438160532	10.90.16.66	10.90.17.36	TCP	54	55829 → 8443 [ACK] Seq=2978 Ack=7551 Win=130560 Len=0
50	3.489994987	10.90.16.66	10.90.17.36	TCP	54	55829 → 8443 [RST, ACK] Seq=2978 Ack=7551 Win=0 Len=0

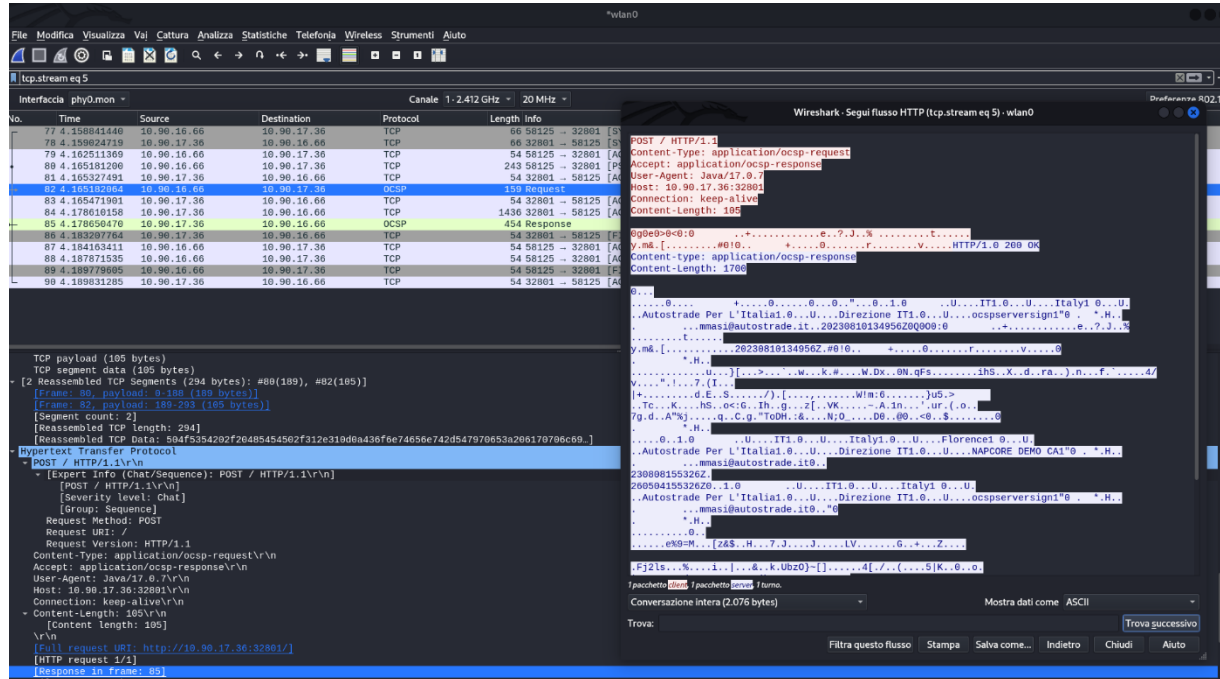
The other method to check for revocation is Online Certificate Status Protocol (OCSP). The standard is defined in rfc6066, rfc6960, and rfc 8954. OCSP is available in two modes:

- Peer-initiated OCSP: during the handshake, the peer obtains the Authority Information attribute from the certificate, and initiates an OCSP request/response protocol to check the revocation status
- OCSP Stapling: the client uses a TLS extension to request a server's copy of the certificate status.

When using OCSP, CRL should be used as a fallback mechanism in case the online OCSP server is not responding. It is suggested to read Section 5 of rfc6960 for further information.

Once instantiated the OCSP server as shown in Appendix, the method `TLSSocket.verify(X509Certificate cert, String ocsUrl)` performs the OCSP request over HTTP as defined in rfc6960 Appendix A. After receiving the client certificate, the method `verify()` overridden in `Main()` firstly obtain the AIA (Authority Info Access) URL from the leaf certificate. Then,

it uses this URL to send the OCSLP request. The evidence is shown in the following Wireshark capture:



The log obtained by OpenSSL is as follows.

```

OCSP Request Data:
  Version: 1 (0x0)
  Requestor List:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 8418D58765D4983F944AC68E2520EDD1EEABD6C4
      Issuer Key Hash: 947493D7D91A001F0D799D6D26CA5BDD038212F2
      Serial Number: 08
  Request Extensions:
    OCSLP Nonce:
      0410F8A079C77E80EB0F0D790160D42E3C86
OCSP Response Data:
  OCSLP Response Status: successful (0x0)
  Response Type: Basic OCSLP Response
  Version: 1 (0x0)
  Responder Id: C = IT, ST = Italy, O = Autostrade Per L'Italia, OU =
  Direzione IT, CN = ocspservernsign, emailAddress = mmasi@autostrade.it
  Produced At: Aug 10 13:51:07 2023 GMT
  Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 8418D58765D4983F944AC68E2520EDD1EEABD6C4
      Issuer Key Hash: 947493D7D91A001F0D799D6D26CA5BDD038212F2
      Serial Number: 08
    Cert Status: unknown
    This Update: Aug 10 13:51:07 2023 GMT
  Response Extensions:
    OCSLP Nonce:
      0410F8A079C77E80EB0F0D790160D42E3C86
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
      9f:34:9e:f2:b3:37:ed:3b:52:27:5f:3c:b3:31:a2:6f:e2:bc:
  
```

```

67:f0:09:de:de:51:f3:37:11:92:ab:b7:73:f8:bc:bc:8c:c1:
c7:27:92:4a:05:5d:85:b6:42:e2:ab:2a:20:c2:6a:9c:21:51:
82:2d:95:ab:a5:44:b0:6e:a8:ba:1e:ba:d6:7f:75:8a:8e:ba:
0e:a9:22:47:9f:7f:93:b7:fe:de:25:19:95:a4:43:e7:9a:f3:
02:44:15:af:82:8d:f5:0e:6b:86:67:d8:f6:e6:71:83:3f:f3:
3c:16:76:d1:ff:8e:94:a6:57:dc:af:37:8d:56:bf:9f:4f:be:
ab:be:87:9f:c2:c4:45:c8:99:46:32:fa:e7:43:61:e0:42:cd:
e0:34:e6:29:63:51:ee:5e:fd:46:cd:2a:ee:49:d7:6b:4a:0f:
42:d8:84:7e:01:65:ff:1b:cf:d7:1d:e1:6d:22:e1:f2:de:ad:
ea:ab:bd:03:d4:2b:76:41:43:90:38:b8:f6:f6:cf:c2:f7:1c:
2c:83:4a:f7:65:b3:e3:ec:8e:6d:1c:03:c1:74:6a:c2:6e:85:
58:a4:0e:f7:8d:57:4c:67:70:01:ad:82:ee:61:fc:8f:1b:77:
e9:00:39:ab:75:ab:b0:07:26:91:12:97:2c:a3:4e:92:e9:28:
c1:66:2b:32

```

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 7 (0x7)

Signature Algorithm: sha256WithRSAEncryption

Issuer: C=IT, ST=Italy, L=Florence, O=Autostrade Per L'Italia,
 OU=Direzione IT, CN=NAPCORE DEMO CA/emailAddress=mmasi@autostrade.it

Validity

Not Before: Aug 8 15:53:26 2023 GMT

Not After : May 4 15:53:26 2026 GMT

Subject: C=IT, ST=Italy, O=Autostrade Per L'Italia, OU=Direzione
 IT, CN=ocspserver/sign/emailAddress=mmasi@autostrade.it

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

Public-Key: (2048 bit)

Modulus:

```

00:b2:65:25:39:3d:4d:88:ad:c7:5b:7a:26:24:d6:
d3:48:d0:13:b1:37:05:4a:0e:9f:f4:be:4a:c6:bb:
db:18:03:4c:56:9a:93:ab:8d:85:8c:ab:47:da:9c:
2b:c7:d9:de:5a:9b:e0:0f:0c:0a:0a:e1:46:6a:32:
6c:73:12:c7:b1:25:de:16:ea:a3:69:82:a6:7c:80:
da:84:26:b2:15:6b:06:55:62:7a:4f:7d:7e:5b:5d:
cd:a9:d9:be:e8:cc:34:5b:a9:2f:a0:f7:28:9a:c9:
d8:ca:35:7c:4b:07:ca:30:e0:19:6f:ac:0d:7c:f7:
65:9b:cb:2d:d5:b8:ba:64:fc:6e:33:90:11:07:dd:
a6:ce:73:77:aa:6a:62:b9:f7:a7:72:9e:db:72:67:
48:94:a1:3e:3a:67:0a:f9:58:00:af:2c:01:bf:f2:
a0:b1:7d:e5:7e:4d:21:e1:74:8f:33:a9:15:1a:52:
49:3d:68:f0:87:cd:82:1e:f9:ad:3e:80:35:19:06:
40:2d:90:8c:f8:41:36:b7:c7:d7:9d:9c:fb:45:3e:
92:91:61:44:d3:3e:79:ca:e6:ec:24:f5:1b:73:77:
4a:7f:da:ec:e3:6f:f0:92:33:e9:24:16:51:8b:4a:
23:b8:05:4a:06:da:8f:0f:8c:b0:ec:05:59:0d:c0:
ef:8b

```

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Basic Constraints:

CA:FALSE

X509v3 Key Usage:

Digital Signature, Non Repudiation, Key Encipherment

X509v3 Subject Key Identifier:

71:DD:3B:F6:CB:FF:4D:B6:4E:50:D6:EE:46:FB:13:C3:8D:A8:9A:3B

X509v3 Authority Key Identifier:

BC:16:E1:90:D3:D6:AC:4F:DE:19:8E:3B:BE:F0:D2:B0:E1:C7:96:9F



```

X509v3 Extended Key Usage:
    TLS Web Server Authentication, TLS Web Client
Authentication, OCSP Signing
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
    85:9f:32:82:c9:cc:cc:99:83:a4:ba:94:34:8c:ea:ab:4b:9d:
    83:d9:e0:13:a0:97:5e:9f:43:fe:d4:f4:94:66:61:92:3b:d2:
    a3:23:27:1d:18:57:d2:bb:af:fa:98:40:04:84:fb:0c:8e:0e:
    25:1b:09:06:f6:aa:94:c0:53:c8:29:50:d3:40:fa:58:61:cc:
    0d:df:fa:63:6f:d4:b1:78:f3:49:4b:f6:84:d8:94:ff:19:06:
    42:4f:ec:61:7f:27:90:91:19:79:2e:9a:e9:4a:76:3b:57:ee:
    60:0e:f9:2b:8c:39:73:ef:08:93:76:a3:8b:67:1c:e7:2d:77:
    f2:67:d2:b2:7a:95:27

```

}Th

The OCSP request is aimed at validating the certificate with serial number “8”, and the response, successful, returns an unknown state.

On the server, in Java, OCSP can be automatically turned on by setting the system properties

```
-Dcom.sun.net.ssl.checkRevocation=true, and -Docsp.enable=true.
```

On the `setenv.sh`.

Certificate Pinning IBB

Name

Certificate Pinning IBB – CPIBB

Rationale

The establishment of secure communication channels is foundational for any message-based communication protocol. However, this is not always sufficient, as the semantics of a Datex II exchange may fall beyond the transmission layer, rising to the application. For instance, a message encoded in SOAP 1.2 may be sent by a *sender* to a *receiver*, but passing *n* different hops, everyone with TLS IBB. A relationship shall be established by the *sender* and *receiver*, which can’t be solely based on the mutual TLS handshake: trust should not be transitive.

The relationship is named *trust relationship* between two or more actors participating in an exchange or in a workflow. In case of TLS, for instance, basing the establishment of the secure channels only through DNS and PKI is exposing the system to attacks that have been seen in the history since years³³. For instance, in TLS IBB it is shown that the endpoint, in order to have a successful handshake, shall validate the certificate as issued by a trust anchor present in its local truststore, and check for its validity either through OCSP or CRL. However, if revocation is active, both methods can be defeated in

³³ See, e.g., https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning



a hostile environment. Moreover, anyone with a valid certificate in the network, can be a man-in-the-middle even for a mutual TLS connection (it will be shown in the exploitation suggestions section).

When performing authentication on the application layer, establishing trust is more cumbersome. In fact, a signed token (see JWTIBB, or SAMLBB) or a signed SOAP Body contain the certificate that has been used to sign along with the message. If the certificate (or the key) is not trusted, anyone can forge the signature or the token, bypassing the access control. Notably, we are under the assumption that the attacker is a valid user in the network (see Annex I). Such attacks, in SOAP, are known as XML Signature Wrapping (XSW)³⁴.

In 2011 the Chrome Web Browser has been able to detect the DigiNotar compromise which uncovered suspected interception in Iran³⁵. A fake website has been used to steal hundreds of passwords of Iranian citizens by using a compromised certificate of the CA. In the ENISA report, it is suggested that “Additional protection could be offered by techniques like CA pinning, whereby the website specifies, on the first visit, which CA it uses”.

Thus, establishing trust is foundational to any message exchange, in particular to mobile applications. Certificate Pinning is the process of associating a host with their *expected* X.509 certificate or public key. Pinning leverages knowledge of the pre-existing relationship between the two principals, and the pin is established usually by using an out-of-band channel (i.e., a different channel).

There are some limitations to certificate pinning, that hinders its deployment:

- It does not work when there are TLS Inspection proxies;
- It requires complex processes to distribute the keys: when a pinned certificate is revoked, all the pinners shall have the updated version;
- If the pinned certificate is hardcoded (e.g., into a static string), a library injection or memory manipulation may bypass the pin.

There are many methods and values to pin. For instance, the entire certificate, the `subjectPublicKeyInfo`, are examples. Standards exist, such as Certificate Transparency (CT) (see rfc9162), but manual method should also be considered. CT is available only for publicly trusted CA, and thus in private environments, other means should exist³⁶.

The out-of-band channel for distributing the keys plays an important role, and it is foundational to pin. Usually, a formal process should be in place to distribute the certificates. Manual distribution is typically used. It is worth noting that the storage of the pinned certificate is important. Having said that hardcoding the values to pin is discouraged, certificates usually go in the local truststore (in case of Java), or a Hardware Security Module. In cloud environments, it is mandatory to use a Bring Your Own Key (BYOK) model, in which the pinset (the set of certificates to pin) is stored in a secure vault of the cloud provider and accessible only to the user. Notably some SaaS models do not allow yet BYOK.

When implementing mobile applications, certificate pinning is of utmost importance³⁷. Insecure communication is ranked 3rd in the Top 10 Mobile Risks. In fact, “when designing a mobile application,

³⁴ See Höller, P., Krumeich, A., Lo Iacono, L. (2021). *XML Signature Wrapping Still Considered Harmful: A Case Study on the Personal Health Record in Germany*. In: Jøsang, A., Fitcher, L., Hagen, J. (eds) *ICT Systems Security and Privacy Protection*. SEC 2021. IFIP Advances in Information and Communication Technology, vol 625. Springer, Cham.

³⁵ See, e.g., <https://www.eff.org/deeplinks/2011/03/iranian-hackers-obtain-fraudulent-https> and the ENISA’s report on the Operation Black Tulip <https://www.enisa.europa.eu/media/news-items/operation-black-tulip>

³⁶ See, e.g., https://chromium.googlesource.com/chromium/src/+refs/heads/main/net/docs/certificate-transparency.md#Locally_trusted-CAs

³⁷ See, e.g., <https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication>



data is commonly exchanged in a client-server fashion. When the solution transmits its data, it must traverse the mobile device's carrier network and the internet. Threat agents might exploit vulnerabilities to intercept sensitive data while it's traveling across the wire. In particular, an adversary that share your local network (compromised or monitored Wi-Fi), Carrier or network devices (routers, cell towers, proxy's), or malware on the mobile device". In the Wi-Fi case, the exploitability is easy. To avoid Insecure Communication, certificate pinning is required. MITRE³⁸ defines certificate pinning as defense D3-CP, and it is suggested to harden the secure communication, by having a *local copy* of the certificate being sent by the server (or by the client, in case of either mutual TLS IBB or message signature WS-IBB or JWT-IBB).

Amongst the vulnerabilities, certificate pinning solves weakness CWE-296, *Improper Following of a Certificate's Chain of Trust*, CWE-295, *Improper Certificate Validation*, CWE-298, *Improper Validation of Certificate Expiration*, and CWE-299, *Improper Check for Certificate Revocation*.

In the information chain from the originator up to the final destination, several intermediaries may participate in the information exchange. In the cases where the information related to the Information Manager and the intermediaries should be supplied, the message should contain all references to the DATEX II nodes which should be referred to the pinned certificate.

Requirements

Brief name	Description	Variability Point
CPIBB-1-T	The certificate to pin SHALL be placed in a local trust file (e.g., the Java truststore) or in an HSM. Hardcoded pin means SHALL NOT be used (e.g., hardcoding the certificate in a constant)	
CPIBB-2-T	If cloud environment is used, the pinset (the set of certificates to pin) SHOULD be placed in a storage where only authorized personnel can have write permissions	
CPIBB-3-T	If a public CA is used, the CA SHALL implement Certificate Transparency	
CPIBB-4-T	When implementing a mobile application to access remote resources, certificate pinning SHOULD be used.	
CPIBB-5-T	All the relevant properties of the certificate and its chain SHALL be fully validated before the certificate is pinned.	
CPIBB-6-T	The DATEX II field <code>InternationalIdentifier</code> or <code>NationalIdentifier</code> SHOULD be checked and related to the pinned certificate associated to the relevant endpoints	Applies only if the <code>InternationalIdentifier</code> or <code>NationalIdentifier</code> are present

³⁸ See, e.g., <https://d3fend.mitre.org/technique/d3f:CertificatePinning/>

Similar IBB and Dependencies

Potentially, all ABBs and IBBs that make use of a certificate may have a dependency on CPIBB.

- [Data Provenance ABB]. Each provenance document is digitally signed. To authenticate the signer, a form of pinning should be performed.
- [Non-Repudiation ABB]. Each nonrepudiation token is digitally signed. In hostile environments, a form of pinning should be performed, to authenticate the relying party.
- [Data Integrity ABB]. This building block is aimed at providing digital signature. In cases where the signature event is performed in non-controlled environments, potentially hostile, a form of pinning should be performed.
- [Data Confidentiality ABB]. If the encryption is performed over a signature (Sign-Before-Encrypt), to authenticate the signed part, a form of pinning should be performed.
- [Authorization ABB]. When presenting a token (see JWT-IBB or SAML-IBB) the pinning of the certificate sent along the signature data structure in the message SHALL be performed. Additional provisioning such as SAML Subject Confirmation Methods may be further profiled.
- [Web Service Security IBB]. This building block should use certificate pinning to avoid such attacks as XML Signature Wrapping. Exploitation will be shown in the IBB.
- [TLS IBB]. This building block should use certificate pinning to establish secure, and mutually authenticated channels.
- [Digital Signature IBB] Authentication of digital signatures may be performed by pinning the signing certificate.
- [JWT IBB] Similarly to WS-IBB, the signing key material should be pinned to avoid that any valid user in the network may forge a valid JWT.

Known Use

N/A. Known use will be described in future versions of this document.

Technical Implementation

The implementation of certificate pinning strongly depends on the use case, and it will be described in the corresponding building block.

To place a certificate in a truststore, in Java, firstly it has to be encoded in DER. A PEM-encoded certificate can be encoded in DER as follows

```
> openssl x509 -in cert.pem -outform DER -out cert.der
```

Then, it can be added in the truststore as follows

```
> keytool -keystore truststore.jks -import -alias cert -file cert.der
```

The sample implementation is in the callback `verify()` of the Main class.

```
if (tlsPinning) {
    KeyStore truststore = socket.getTestTruststore();
    Collections.list(truststore.aliases()).forEach(x -> {
        try {
            X509Certificate tPCert = (X509Certificate)
truststore.getCertificate(x);
```

```

        System.out.println("Obtained certificate from TLS " +
System.lineSeparator() +
        leafCert.getSubjectDN() + System.lineSeparator() +
        " and the one that I have in my truststore is " +
System.lineSeparator() +
        tPCert.getSubjectDN());
        if (Arrays.equals(leafCert.getPublicKey().getEncoded(),
tPCert.getPublicKey().getEncoded())) {
            System.out.println("Found a certificate to be pinned with key " +

Base64.getEncoder().encodeToString(leafCert.getPublicKey().getEncoded()));
            setFoundpinnedCert(true);
            return;
        } else {
            System.out.println("[DEBUG] expected the key :" +
System.lineSeparator()
            +
Base64.getEncoder().encodeToString(tPCert.getPublicKey().getEncoded()) +
            System.lineSeparator() + ": while "
            + "I got :" + System.lineSeparator() +

Base64.getEncoder().encodeToString(leafCert.getPublicKey().getEncoded()));
        }
    }
}

```

The pin in this case is performed over the public key with the `Arrays.equals`, which checks if the two byte arrays are equal and ordered the same way.

If the pinning is performed over, e.g.,

```
d2:payload/com:publicationCreator/com:nationalIdentifier()
```

it has to be checked against a file that could be encoded, as, e.g.,:

```

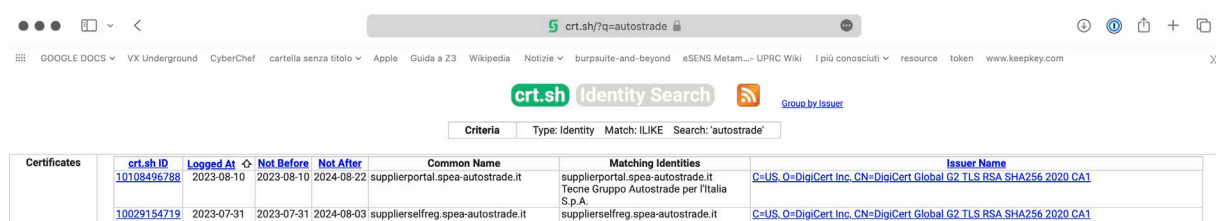
IT403
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAn+VZsEvgDYJ0GDJWl7Hx1Lu0ZA/v2jP
0TzafW6ZHaX0vgRfkel7KDDtOk6Z15hws37nVfuwM6nL/bNKgxJaS1158E09FJn1beF49+7Gvti
A1su0NSE6KhybLe67sQAss6GM1ilApl3w8utMnwDnnt9tC/QWAXWl1o2aqhe17PtUYrHK2n821C
F2CpR803vLZx+lwwMw5dnndEve/xHMA3tyZz54AA4UJcI4JFyEk1P0/AHcibRbhKnj1LXuHFPa
G5JMTcxcOrupEgSqPl//pqCa8NhMLog4XONq2tx1bHUYdH9N1ewtOjqWYnp1+U/k8iCX2Q/n7eD
5YxYhgBiTiQIDAQAB\n

```

This is an example of a space-separated couple as $\langle v_1, v_2 \rangle$ where v_1 is the identifier, while v_2 is the Base64-encoded value of the public key corresponding.

Notably enabling certificate pinning on a server side, it is depending on the application server, if used.

To check if a certificate for a website is emitted by a certificate authority which is supporting Certificate Transparency, an online tool can be used, such as <https://crt.sh>, as in the picture below.



Criteria	Type	Identity	Match	ILIKE	Search	'autostrade'
Certificates	crt.sh ID	Logged At	Not Before	Not After	Common Name	Matching Identities
	10108496788	2023-08-10	2023-08-10	2024-08-22	supplierportal.spea-autostrade.it	supplierportal.spea-autostrade.it
	10029154719	2023-07-31	2023-07-31	2024-08-03	supplierselfreg.spea-autostrade.it	supplierselfreg.spea-autostrade.it
						Tecne Gruppo Autostrade per l'Italia S.p.A.
						C=US, O=DigiCert Inc, CN=DigiCert Global G2 TLS RSA SHA256 2020 CA1
						C=US, O=DigiCert Inc, CN=DigiCert Global G2 TLS RSA SHA256 2020 CA1



At the moment there are no libraries that implement certificate transparency in the client side for the major programming languages (except the Go implementation³⁹). As CT is becoming a cornerstone of Internet security, this situation may change.

Test Assertions and exploitation suggestions

Table 5

TA id	CPIBB-TA-1
Normative Source:	The certificate to pin SHALL be placed in a local trust file (e.g., the Java truststore) or in an HSM. Hardcoded pin means SHALL NOT be used (e.g., hardcoding the certificate in a constant)
Target	The SuC in its production environment
Prerequisite	The source code for the SuC is available, and the SuC is up and running
Prescription Level	Mandatory
Predicate	The code should point to a file, and this file should be protected by a password. Write permissions should only be given to the sysadmin.
Variable	The pinned certificate can be in the HSM, and in this case the manual of the HSM vendor should be read.

If the key to pin is hardcoded, it is easy to bypass the pinning, as follows. In languages such as Java, if the key is hardcoded as follows (MainHardCodedPinning),

```
/** The hardcoded key. */
private static String hardcodedKey =
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQE7UR2ScyPlOf7H6GNvxFegtJVbdPH6i
d3db7ZcUMTMoSBBqSj8zQpuKVbnjntv2TYc6pZD+T5Xsg4Thg0Rfilst47S34i2alEtzDolBbu
I/wnkZ2wox6pmlDncWC5E1HG2/5Vcxi9XeT7abjFbQLNkvH9iKBTv2jJEjbbde6YlWjQCaziSKK
mIUrbz84IU/q584KaPsztmH1jFLLObOCCnWjnBr4Bfaz7LNIrIbz8ufpYubiF/cxKQ/WS1/7CfV
qWaYa2awX0D+zisWIVsYjLyuc5Z76aLLHunnfRyUZtoy/qLhlRPHOFFEjVg+4lcmCeUVsv3r4f
FUEB29knYuvwIDAQAB";
```

the pin can be implemented as follows

```
/*
 * Now we try to pin the certificate. The certificates to be pinned are all
 * the one which are verified before in the PKIX
 */
if (tlsPinning) {
    if (!Base64.getEncoder()
        .encodeToString(
            leafCert.getPublicKey()
                .getEncoded())
        .equals(hardcodedKey)) {
        throw new
```

³⁹ See, e.g., <https://github.com/google/certificate-transparency-go>

```

        SSLPeerUnverifiedException("The certificate is not pinned!");
    } else {
        System.out.println("Pin is ok");
    }
}

```

However, if the code is not obfuscated, it is relatively easy to use tools such as Frida⁴⁰ or plain java reflection to bypass the pinning. The following code shows how to wrap the main class to bypass pinning.

```

/*
 * Instantiate the decompiled class
 */
Object main = new MainHardCodedPinning();
Field[] fields = main.getClass().getDeclaredFields();
try {
    /*
     * Change the fields and the certificate value
     */
    fields[1].setAccessible(true);
    fields[1].set(main, "WrongKeyToBePinned");
} catch (IllegalArgumentException | IllegalAccessException e) {
    throw new IllegalStateException(e);
};
((MainHardCodedPinning)main).main(null);

```

In this way, when the `main()` method is called, the pin is performed with the key `WrongKeyToBePinned`, thus allowing the system to be pointed to a malicious proxy. It is worth mention that in order to execute the attack using the plain Java reflection API, the attacker has to be in the machine, disassemble the jar, and execute code on the machine. However, tools such as Frida will allow to attach to a running system and perform the same changes.

For other languages, library injection plays a similar role. For instance, in C/C++ and in addition to tools such as Frida, a library could be crafted to intercept the variable containing the public key and injected using the `LD_LIBRARY_PATH` environment variable.

If there are no other ways but to hardcode the materials to be pinned, at least the code should be obfuscated.

Table 6

TA id	CPIBB-TA-2
Normative Source:	If cloud environment is used, the pinset (the set of certificates to pin) SHOULD be placed in a storage where only authorized personnel can have write permissions
Target	The SuC in its production environment

⁴⁰ See, e.g., <https://frida.re>

Prerequisite	The source code for the SuC is available, and the SuC is up and running in a cloud instance
Prescription Level	Optional
Predicate	The pinset should be placed in systems like AWS secret, or Hashicorp Vault.
Variable	This test requires a cloud implementation

In order to contain the risk of Cloud Service Provider key compromise, it is suggested to use a Bring Your Own Keys (BYOK) policy, where the CSP has no means to tamper with the pinset. In fact, if the pinned certificates are stored in a file system, the administrative account of the server have access and may manipulate the system without having a log entry.

Table 7

TA id	CPIBB-TA-3
Normative Source:	If a public CA is used, the CA SHALL implement Certificate Transparency
Target	The SuC in its production environment
Prerequisite	The SuC has a certificate issued by a public CA
Prescription Level	Mandatory
Predicate	The CA shall be present in the Certificate Transparency Logs
Variable	This test requires to have a public CA

Since 2018, major web browsers, such as Safari and Chrome, require the certificates emitted for a website to be present in a Certificate Transparency Log. For this reason, when the SuC is implemented as a web application, to have the guarantee that the certificate may not be misused or the CA has not been compromised, a check for the presence of the certificate in the Certificate Transparency shall be done. The test is easily performed using the <https://crt.sh> as shown in Section Technical Implementation. Notably, that the situation may change, and libraries implementing CT will be available, and thus this check can be performed by, e.g., a RESTful or a SOAP client.

Table 8

TA id	CPIBB-TA-4
Normative Source:	When implementing a mobile application to access remote resources, certificate pinning SHOULD be used.
Target	The SuC in its production environment
Prerequisite	The SuC is implemented as a <i>native</i> app and is accessing data to a remote storage
Prescription Level	Optional
Predicate	The certificate pinning should be used
Variable	This test works only in case of mobile application

Certificate pinning in mobile application is suggested by OWASP MASVS and MASTG⁴¹ containing practical indication on how to pin. In particular, the following requirements apply.

- The pinned certificate shall not be stored in the local storage of the application
- The endpoint to be pinned shall be the endpoint under the control of the developer, or for which a pinning policy is provisioned

In general, both iOS and Android operating system implements pinning very easily.

For non-native apps, such as HTML5-based applications like Single Page Applications, certificate pinning should be carefully considered to **not** be implemented, as SPAs run in browser and they may access the local storage only, if not properly configured.

Table 9

TA id	CPIBB-TA-5
Normative Source:	All the relevant properties of the certificate and its chain SHALL be fully validated before the certificate is pinned.
Target	The SuC in its production environment
Prerequisite	The SuC is implemented and properly configured for PKI and certificate pinning
Prescription Level	Mandatory
Predicate	All the PKI checks (e.g., certificate validity, PKIX algorithm, OCSP/CRL, Certificate Transparency, and policy enforcement) are performed for all the certificates in the certificate chain
Variable	

The certificate checks shall be performed for every certificate in the chain. This means that, at least:

- Certificates have a valid expected syntax.
- Certificates are not expired.
- Certificates are emitted by a trusted certification authorities up to a trust anchor (the topmost certificate).
- If a certificate has a distribution point, then a CRL shall be retrieved, or OCSP must be contacted in order to check for revocation.
- Key sizes are according with the expected policy.
- Key usages are coherent with the use cases.
- Extended Key Usages are coherent with the use cases.
- Additional certificate policies are coherent with the message exchange (whose definition goes beyond this specification)

Those checks are defined according with *Prudent Engineering Techniques for Cryptographic Protocols*⁴².

⁴¹ <https://mas.owasp.org/MASTG/General/0x04f-Testing-Network-Communication/#restricting-trust-identity-pinning>

⁴² M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols," in *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 6-15, Jan. 1996, doi: 10.1109/32.481513.

In particular, we show how not pinning a certificate may be used to break a mutually TLS encryption performed using TLS IBB. In fact, although mutually authenticated TLS channels guarantee authentication on both sides, under the Dolev-Yao assumption, an attacker can intercept the message on its way. Thus, only checking the certificate validity and the certificate against the entire chain, not pinning would let a MITM proxy to intercept and manipulate the message. In this example we show that the client has a specific user agent as follows.

```
con.setRequestProperty("User-Agent", "Original Application");
```

We will show that using a MITM proxy (Burp Suite) with the certificate of Mallory, the connection between the client and the server, even mutually authenticated, is intercepted and manipulated.

BURP Suite is configured by using certificates with *attacker* in the DN. After convincing the client application to send the message to BURP (e.g., via ARP spoofing or DNS poisoning, beyond the scope of this deliverable), the client executes without any error.

```
<terminated> main (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-10.0.2.jdk/Contents/Home/bin/java (20 ago 2023, 15:40:
0010: 03 12 41 4E 03 03 9A 80 80 8C 3E 3D 2E 07 C7 7E ..UN.....
0020: A2 36 D0 81 7D ED 2F 32 80 4A 12 0D D8 CB 81 A5 .6..../2.J.....
... no MAC keys used for this cipher
Client write key:
0000: 7F 9B 4C 38 CB 7C 41 58 A4 9B F2 8A 9F 6A B6 82 ..L8..AX.....j..
0010: 11 22 5D F9 4B E0 D8 FC 91 62 A3 8C BB 91 45 8B .".K....b....E.
Server write key:
0000: A1 A8 A6 4E 2D 8B 33 25 C0 69 73 57 3E 4F 67 6F ...N-.3%.isW>0go
0010: A2 55 07 ED 0B 5D 1D AE E3 1D E7 65 27 3D AF 50 .U...].e'=.P
Client write IV:
0000: DB 16 C2 48 ...H
Server write IV:
0000: 2F E3 4E 9C /.N.
update handshake state: change_cipher_spec
upcoming handshake states: client finished[20]
upcoming handshake states: server change_cipher_spec[-1]
upcoming handshake states: server finished[20]
main, WRITE: TLSv1.2 Change Cipher Spec, length = 1
[Raw write]: length = 6
0000: 14 03 03 00 01 01 .....
*** Finished
verify_data: { 169, 28, 3, 245, 2, 241, 185, 22, 28, 206, 103, 252 }
***
update handshake state: finished[20]
upcoming handshake states: server change_cipher_spec[-1]
```

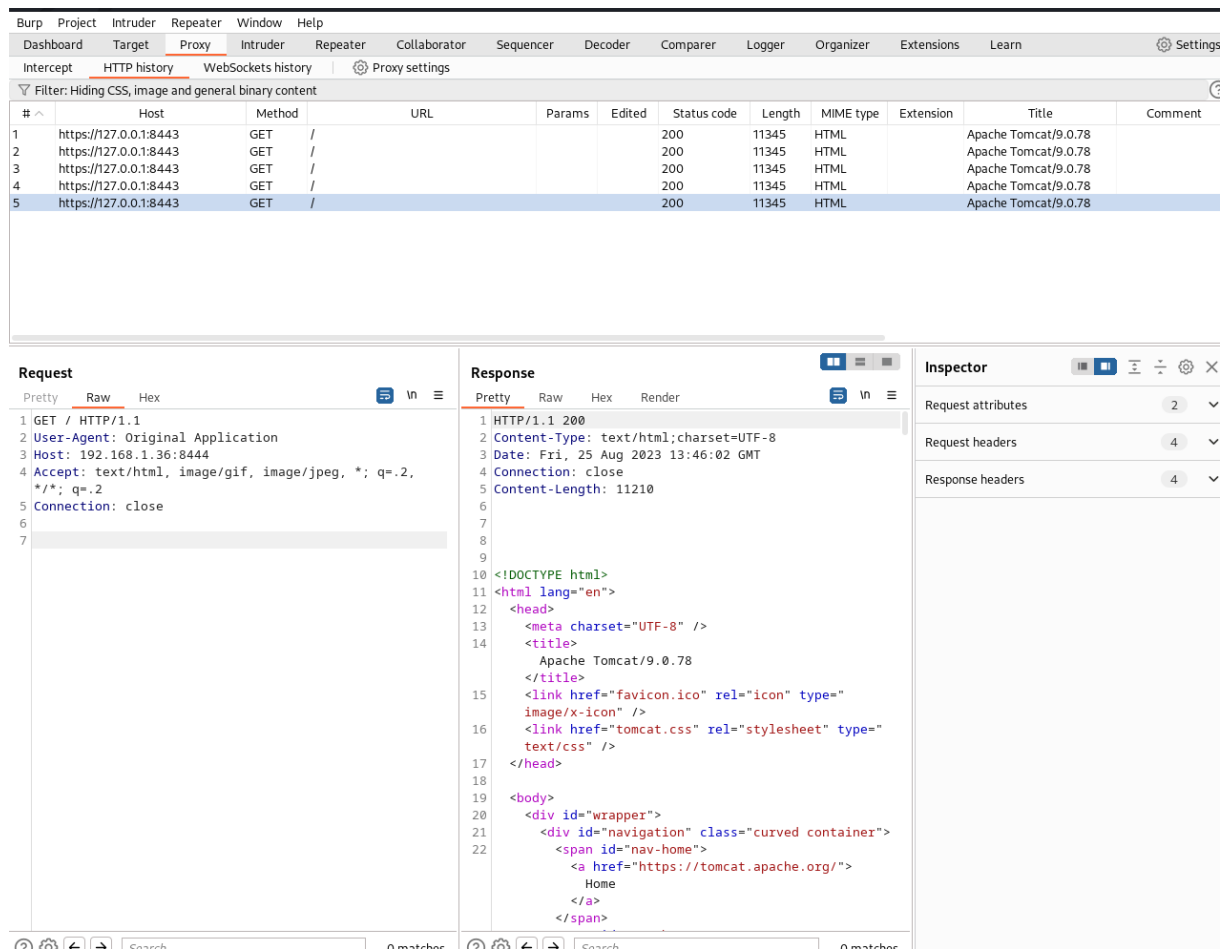
However, the ServerHello message contains the fake certificate.

```
upcoming handshake states: client_key_exchange[10]
upcoming handshake states: certificate_verify[15](optional)
upcoming handshake states: client_change_cipher_spec[-1]
upcoming handshake states: client finished[20]
upcoming handshake states: server_change_cipher_spec[-1]
upcoming handshake states: server finished[20]
*** Certificate chain
chain [0] = [
[
Version: V3
Subject: EMAILADDRESS=mmas@autostrade.it, CN=attacker, OU=Direzione IT, O=Autostrade Per L'Italia, ST=Italy, C=IT
Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11

Key: Sun RSA public key, 2048 bits
modulus: 25060427570742720894978831766104974061441133547530993836913517878721642367267574176094453832411253174461905894628275297716451995
public exponent: 65537
Validity: [From: Mon Aug 21 15:12:10 CEST 2023,
To: Sun May 17 15:12:10 CEST 2026]
Issuer: EMAILADDRESS=mmas@autostrade.it, CN=NAPCORE DEMO CA, OU=Direzione IT, O=Autostrade Per L'Italia, L=Florence, ST=Italy, C=IT
SerialNumber: [ 09]

Certificate Extensions: 6
[1]: ObjectId: 1.3.6.1.5.5.7.1.1 Criticality=false
```

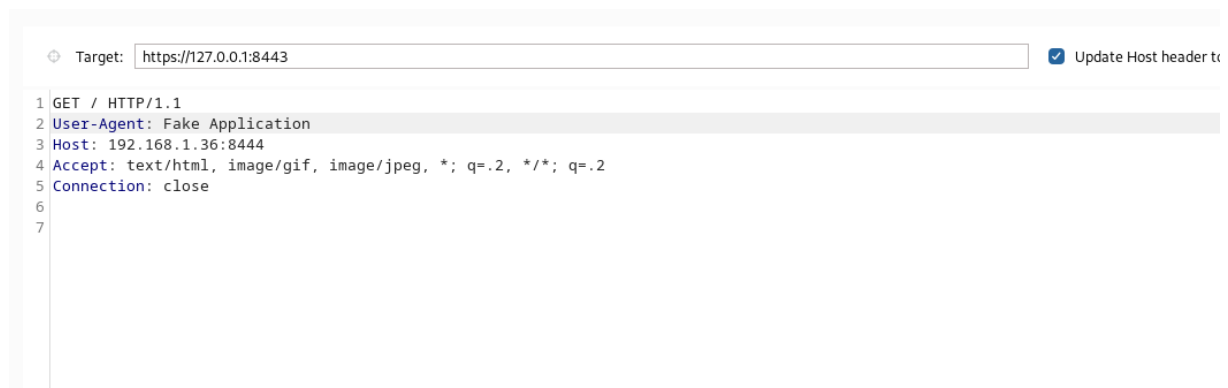
In BURP we see the original request, as follows.



The screenshot shows the Burp Suite interface. At the top, the menu bar includes Burp, Project, Intruder, Repeater, Window, and Help. Below the menu, the toolbar contains various tools like Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, Learn, and Settings. The main window displays a table of HTTP history with columns for #, Host, Method, URL, Params, Edited, Status code, Length, MIME type, Extension, Title, and Comment. The table shows five entries, all with a status code of 200 and a MIME type of HTML. The selected entry (row 5) is a GET request to https://127.0.0.1:8443.

Below the history table, the 'Request' and 'Response' panels are visible. The 'Request' panel shows a GET request to https://127.0.0.1:8443 with a User-Agent of 'Original Application'. The 'Response' panel shows the corresponding HTML response from Apache Tomcat/9.0.78. The 'Inspector' panel on the right shows the request and response headers.

By changing the message on the BURP's intruder, we have set the User Agent to *Fake Application* to demonstrate that we have been able to manipulate the payload in the mutual TLS connection.



The screenshot shows the Burp Suite interface with the 'Intruder' tool selected. The 'Target' field is set to https://127.0.0.1:8443. The 'Update Host header to' checkbox is checked. The 'Request' panel shows a modified GET request to https://127.0.0.1:8443 with a User-Agent of 'Fake Application'.

And then, forward it to the backend, which logs Fake Application.

```

)
javax.net.ssl|DEBUG|12|https-jsse-nio-8443-exec-3|2023-08-25 15:44:53.091 CEST|SSLEngineInputRe
javax.net.ssl|DEBUG|12|https-jsse-nio-8443-exec-3|2023-08-25 15:44:53.092 CEST|SSLCipher.java:1
0000: 47 45 54 20 2F 20 48 54 54 50 2F 31 2E 31 0D 0A GET / HTTP/1.1..
0010: 55 73 65 72 2D 41 67 65 6E 74 3A 20 46 61 6B 65 User-Agent: Fake
0020: 20 41 70 70 6C 69 63 61 74 69 6F 6E 0D 0A 48 6F Application..Ho
0030: 73 74 3A 20 31 32 37 2E 30 2E 30 2E 31 3A 38 34 st: 127.0.0.1:84
0040: 34 33 0D 0A 41 63 63 65 70 74 3A 20 74 65 78 74 43..Accept: text
0050: 2F 68 74 6D 6C 2C 20 69 6D 61 67 65 2F 67 69 66 /html, image/gif
0060: 2C 20 69 6D 61 67 65 2F 6A 70 65 67 2C 20 2A 3B , image/jpeg, *;
0070: 20 71 3D 2E 32 2C 20 2A 2F 2A 3B 20 71 3D 2E 32 q=.2, */*; q=.2
0080: 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C ..Connection: cl
0090: 6F 73 65 0D 0A 0D 0A connection: close....
)
javax.net.ssl|DEBUG|12|https-jsse-nio-8443-exec-3|2023-08-25 15:44:53.094 CEST|SSLEngineOutputR
javax.net.ssl|DEBUG|12|https-jsse-nio-8443-exec-3|2023-08-25 15:44:53.102 CEST|SSLCipher.java:1
0000: 48 54 54 50 2F 31 2E 31 20 32 30 30 20 0D 0A 43 HTTP/1.1 200 ..C
0010: 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 74 65 78 Content-Type: tex

```

When enabling the certificate pinning, the client immediately discovers that the connection has been established against the attacker.

```

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAXoR0rrFvHxKRUNkxLn6mhrHPGxbZCubuuWILVX6Znsa0gW0XbxLV70tHNC0irgwUcKvCRU7hFW0FL7H5pT0:
Obtained certificate from TLS
EMAILADDRESS=mmasasi@autostrade.it, CN=attacker, OU=Direzione IT, O=Autostrade Per L'Italia, ST=Italy, C=IT
and the one that I have in my truststore is
EMAILADDRESS=mmasasi@autostrade.it, CN=server, OU=Direzione IT, O=Autostrade Per L'Italia, ST=Italy, C=IT
[DEBUG] expected the key :
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAXoR0rrFvHxKRUNkxLn6mhrHPGxbZCubuuWILVX6Znsa0gW0XbxLV70tHNC0irgwUcKvCRU7hFW0FL7H5pT0:
: while I got :
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAXoR0rrFvHxKRUNkxLn6mhrHPGxbZCubuuWILVX6Znsa0gW0XbxLV70tHNC0irgwUcKvCRU7hFW0FL7H5pT0:
Obtained an exception when verifying the certificate The certificate [
[
Version: V3
Subject: EMAILADDRESS=mmasasi@autostrade.it, CN=attacker, OU=Direzione IT, O=Autostrade Per L'Italia, ST=Italy, C=IT
Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11

```

As a consequence, the TLS connection is dropped before the request arrives to the malicious proxy.

This test witness that even using mutual TLS connections, a Dolev-Yao attacker can intercept and manipulate messages in its way.

Certificate Pinning is of course a defense, but it should be carefully designed as it may hinder the availability of the system, by adding a significant burden to the system administrators or reliance on the PKI tools. Risk Analysis should consider, for instance:

- The risk of compromission of the Certificate Authority.
- The risk of an ARP or DNS spoofing by evaluating, e.g., how malicious devices or compromised devices can join the network.
- The level of detection of malicious traffic in the network.
- The availability of skilled System Administrators available to meet the Availability requirements when rekeying.

Table 10

TA id	CPIBB-TA-6
Normative Source:	The DATEX II field <code>InternationalIdentifier</code> or <code>NationalIdentifier</code> SHOULD be checked and related to the pinned certificate associated to the relevant endpoints
Target	The SuC
Prerequisite	The source code of the SuC
Prescription Level	Optional
Predicate	Check if the callback for certificate verification reads a database or a file containing the association between a public key (or other means

	to be pinned) and the host. Then control if the code presents a check that the leafCert attribute is equal to the one presented in the database file
Variable	Only in case of use of the corresponding DATEX II fields.

Web Service Security IBB

Name

Web Service Security IBB – WSS-IBB

Rationale

This building block defines the interoperability measures that web services implementation should follow. In particular, this building block, defines a profile for SOAP 1.2 Web Service Security⁴³ following the Web Service Basic Security Profile, WS-I BSP⁴⁴, and defines, in particular:

- How to **sign** parts of a SOAP Envelope
- How to **encrypt** parts of a SOAP Envelope
- How to **sign-then-encrypt** parts of a SOAP Envelope
- How to add a **timestamp** to an envelope
- How to add a **SAML Assertion** to an envelope

Although SOAP is a well-known protocol, interoperability problems may arise, even when a WS-SecurityPolicy⁴⁵ is attached to the WSDL definition of the web service. This is potentially due to different interpretation of the WS-Security specifications, meaning that it contains options that has to be profiled⁴⁶. WS-Security is composed by a main specification (WSS CORE) and additional profiles. In SOAP, web service messages have been designed to be manipulated in their way. Being based on XML, SOAP messages are strongly typed, with a syntactic structure of a tree. WSS enables the signature, encryption, of every subtree with different keys allowing, for instance, to have a message sent by *A*, to *C*, delivered via *B* (where *A*, *B*, and *C* are *Actors*). Each actor may have access to parts that it is entitled to do so, realising such as cases as:

- *A*, sends a message with information for *C*, that *B* is not entitled to access, but with some routing data only for *B*.
- *A*, sends a message to *C*, but in order to achieve an agreement, also *B* shall have read and understood the message. Thus, *A*, can sign parts of the message, *B* can sign others, and therefore *C* may authenticate both parts.

This flexibility comes with a cost. In fact, SOAP messages are prone to many types of attacks⁴⁷, example:

⁴³ See https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

⁴⁴ See <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.1.html#TransportLayerMechanisms>

⁴⁵ See <https://www.w3.org/Submission/WS-Policy/>

⁴⁶ XXXX MISSING REFERENCE.

⁴⁷ See, e.g., https://owasp.org/www-pdf-archive/07A_Breaking_XML_Signature_and_Encryption_-_Juraj_Somorovsky.pdf but in particular Tibor Jager, Juraj Somorovsky: *How To Break XML Encryption* - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.



- XML Signature Wrapping, an attack to the structure of the XML. In the picture it is shown that the attacker injects a body signed with a valid signature

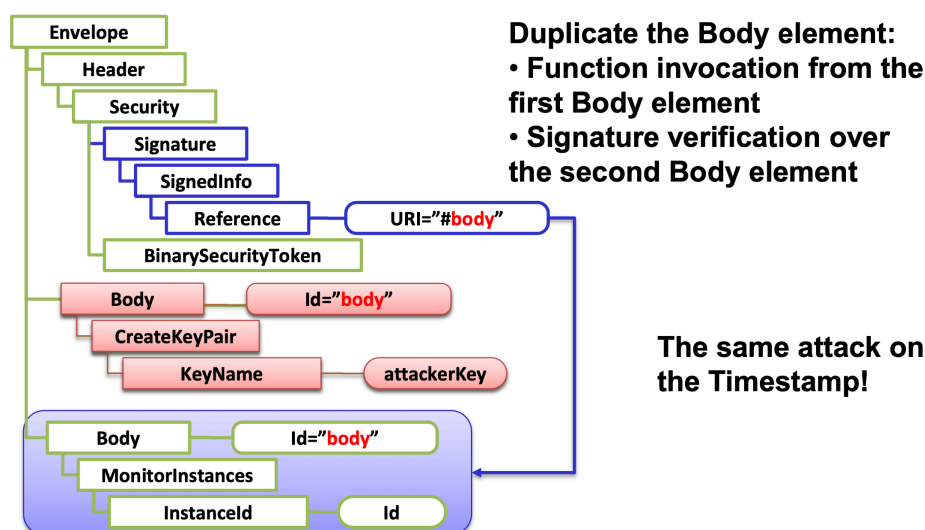


Figure 16 Picture from J. Somorowski et al, *All your clouds belong to us – Security Analysis of Cloud Management Interfaces*, In *Proceeding of the ACM Cloud Computing Security Workshop (CCSW)*, 2011

- SAML Rewrite Attack⁴⁸, again an attack to the structure of the SOAP envelope, where a valid SAML assertion is signed by the attacker and replaced in the SOAP envelope.

For these reasons, this IBB aims at identifying a profile for WSS which is aligned with the CEF building block for eDelivery⁴⁹, while keeping enough flexibility for the different use cases that may arise.

When dealing with SAML assertions, this IBB do not prescribe any means to obtain it.

This specification is built upon Web Service Security SOAP Message Security and Web Service Security X.509 Certificate Token Profile.

Requirements

Table 11

Brief name	Description	Variability Point
WSIBB-1-T	The SOAP Envelope SHALL remain a valid SOAP envelope even after encryption or signature: methods other than those defined by the WS-Security are not allowed.	
WSIBB-2-T	The elements in the security header of the SOAP envelope SHALL be ordered in the way the clients are able to process it	
WSIBB-3-T	The SOAP Envelope SHOULD contain a Timestamp element containing just exactly one Created and one Expired element. The timestamp SHOULD be signed	The timestamp may not be signed in case an envelope is to be re-sent after caching, and thus just the timestamp could be changed. This introduce the risk of

⁴⁸ See, e.g., *Secure Sessions for Web Services*, <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/secure-sessions-sws.pdf>

⁴⁹ Notably that this specification is only aimed at the security part of the eDelivery building block, as defined in section 3.2.6 of <https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/eDelivery+AS4+-+1.15>

		manipulation of the message that SHALL be evaluated.
WSIBB-4-T	The SOAP Header, after applying this building block, SHALL be compliant with the WSI-BSP v1.1	
WSIBB-5-T	When using SAML as token in the header, the OWASP Cheat Sheet ⁵⁰ SHALL be considered. When relying on digital signature, out-of-band channels for schema, keys, and certificates SHALL be used.	
WSIBB-6-T	The HashFunction SHALL be set to http://www.w3.org/2001/04/xmlenc#sha256 , while the Signature algorithm SHALL be set to http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 .	Applies only if the <code>InternationalIdentifier</code> or <code>NationalIdentifier</code> are present
WSIBB-7-T	A Schema Validation SHALL be performed every time a new incoming message is received	

Similar IBB and Dependencies

- CPIBB – certificate pinning is RECOMMENDED every time a signature should be checked (e.g., in case of using XML Signature or SAML assertions)
- CTIBB – Consistent time is REQUIRED.
- TLSIBB – Transport Layer Security should be used between the hops⁵¹

Known Use

N/A. Known use will be described in future versions of this document.

Technical Implementation

The WSS defines a way how security tokens must be used within a SOAP Envelope.

```
<soap:Envelope>
  <soap:Header>
    <wsse:Security>
      <wsse:BinarySecurityToken>ASDAHGSDKAJGSD
```

And it uses XML Signature as underlying standard.

```
<Signature ID?>
  <SignedInfo>
    <CanonicalizationMethod/>
    <SignatureMethod/>
    (<Reference URI? >
      (<Transforms>) ?
```

⁵⁰ See https://cheatsheetseries.owasp.org/cheatsheets/SAML_Security_Cheat_Sheet.html

⁵¹ See Security section of <https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/eDelivery+AS4+-+1.15>

```

    <DigestMethod>
    <DigestValue>
  </Reference>)+
</SignedInfo>
<SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
</Signature>

```

The signature data structure is composed by several parts which enable the deterministic verification process in two successive steps:

- **Step 1, *reference validation*:** the `SignedInfo` element is put in its canonical form using the algorithm specified in the `CanonicalizationMethod`. Then, for each Reference element, the referenced data object is retrieved, and the digest value is computed and compared with the `DigestValue` under the Reference element. If this reference validation succeeds then the signature validation is executed, otherwise it generates an error message.
- **Step 2, *signature validation*:** the keying information is retrieved under the `KeyInfo` element who points to an external source, and the signature method used is defined under the `SignatureMethod` element. These two elements are then used to validate the `SignedValue` element.

When applying this algorithm to SOAP, the processing rules can be sketched as follows. The receiver:

- Obtain the security header.
- Obtain the WS-Security header.
- Check the `SignedInfo` value.
- Resolve the references in the whole SOAP tree.
- Check the validity of the Signature as above.

An XML Rewrite Attack can be as follows. The client sends the following message.

```

<soap:Envelope>
  <soap:Header>
    <wsse:Security>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:Reference URI=#1/>
          <ds:Reference URI=#2/>
        </ds:SignedInfo>
        <ds:SignatureValue>
          ASDGKSDFATRBU
        </ds:SignatureValue>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body ID='`#1`'>
    <tns:TransferMoney From='`savings`' To='`massi`' />
    <tns:TAN>12345</tns:TAN>
  </soap:Body>
</soap:Envelope>

```

In this message, a signed SOAP body is referenced by URI #1, and it is supposed to transfer money from the “savings” account to the “massi”, legitimate, account, with the TAN number.

Under the Dolev-Yao assumption, an attacker can be a MITM as in CPIBB and manipulate the message as follows.

```

<soap:Envelope>

```

```

<soap:Header>
  <wsse:Security>
    <ds:Signature>
      <ds:SignedInfo>
        <ds:Reference URI=#1/>
        <ds:Reference URI=#2/>
      </ds:SignedInfo>
      <ds:SignatureValue>
        ASDGKSDFATRB
        <attacker:BogusHeader>
          <soap:Body ID='`#1`'>
            <tns:TransferMoney To='`massi`' />
            <tns:TAN>12345</tns:TAN>
          </attacker:BogusHeader>
        </ds:SignatureValue>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <tns:TransferMoney From='`savings`' To='`attacker_account`' />
    <tns:TAN>12345</tns:TAN>
  </soap:Body>
</soap:Envelope>

```

In this example, the attacker wraps the original body into a `BogusHeader`, and adds its own one, that transfers the data to the attacker account. If the algorithm is implemented according to the specifications, this message is considered valid, as the wrapping does not change the signature. In fact, the vulnerability is due to the fact that Signed object is referenced by an indirection from the Reference element of the signature. This indirect reference gives no information about the actual location of the signed element.

To avoid such problems, it is imperative that the schema of the message is checked in advance. Thus, the implementation of WSIBB-2-T shall be composed as follows:

- Underlying protocol: HTTPS
 - SOAP Envelope
 - SOAP Header
 - (Eventual eb:Messaging, in case of using CEF eDelivery)
 - wsse:Security
 - SAML Assertion (or empty)
 - Signature (or empty)
 - Encryption (or empty)
 - Timestamp
 - SOAP Body

The referenced element SHALL NOT be referenced using a flat linked list, but the entire XML tree SHALL be traversed.

Implementation of Message Signature with a Timestamp

When implementing the signature of a message, the following requirement SHALL apply.

- The Canonicalization algorithm SHALL be <http://www.w3.org/2001/10/xml-exc-c14n#>
- The Signature method SHALL be <http://www.w3.org/2001/04/xmldsig-more#rsa-sha256>
- The Digest method SHALL be <http://www.w3.org/2001/04/xmldsig-more#sha256>
- The KeyInfo material SHOULD contain a Security Token Reference. X509Data MAY be used. The content of X509Data MAY BE an issuer and serial number, or the

X509Certificate list, ordered from the leaf to the root. All these elements are defined in the `xmlns:ds` namespace.

- The signed element SHOULD BE the Soap Body

The following is an excerpt from a signed message, using SOAP 1.2, which is, *mutatis mutandis*, similar to SOAP 1.1.

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" env:mustUnderstand="true">
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="SIG-f1bba883-d71e-4d9c-b5b7-5bfb6ea57be">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
            Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
            <ec:InclusiveNamespaces
              xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="env"/>
            </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
          <ds:Reference URI="#id-765320ef-6b29-4523-82fa-9dfb75907be2">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            </ds:Transforms>
            <ds:DigestMethod
              Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha256" />
            <ds:DigestValue>FL30TAE5uZVgqQ/wyEgs7qLVOisnsA5hgmQ6k1Io9Jk=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>EC8qFkG1U+uL04sqT6gZ+J4huPORCERTTT/bsrKcLZV8i4iGcSXLoGZp
          S7kA0dzgayfeUR674PPl&#13;
          QiFrXENTWEffg2I/LiBZVTExAX35H+3ZuwycBFopx0TPp1Kc2K6sZq/00FlU1KUrr+yRgdIAZCmx
          J&#13;
          fqcryCQiZChjCBpeTe/Px9dDVrGk/YAA178pRDhNrR6HAaeCjKV7IEUcixS5vSKSOWVRPSOiYDG
          G&#13;
          Vw8UR6UVVnxBqYtU8cpBIYc8k+S7GTou5HN+5d1oIIMoZ0xccree87D8CvNekuyZfWtBD1JpMv
          J&#13;
          9wLUld56kncry1xzsuJ7ArplLgog0hHEIjgw6g==</ds:SignatureValue>
        <ds:KeyInfo Id="KI-fd0f753e-9d4a-47c4-98e3-685e9a4f81e4">
          <wsse:SecurityTokenReference wsu:Id="STR-2bef508d-907a-4772-9b8f-df97ea790456">
            <ds:X509Data>
              <ds:X509IssuerSerial>
                <ds:X509IssuerName>1.2.840.113549.1.9.1=#16136d6d617369406175746f7374726164
                  652e6974,CN=NAPCORE DEMO CA,OU=Direzione IT,O=Autostrade Per
                  L'Italia,L=Florence,ST=Italy,C=IT</ds:X509IssuerName>
                <ds:X509SerialNumber>1</ds:X509SerialNumber>
              </ds:X509IssuerSerial>
            </ds:X509Data>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </env:Header>
  <env:Body>
```

```

        </ds:X509Data>
        </wsse:SecurityTokenReference>
    </ds:KeyInfo>
</ds:Signature>
    <wsu:Timestamp wsu:Id="TS-bb470ff9-e35b-40eb-8be5-e73581852ead">
        <wsu:Created>2024-03-06T15:51:27.380Z</wsu:Created>
        <wsu:Expires>2024-03-06T15:56:27.380Z</wsu:Expires>
    </wsu:Timestamp>
</wsse:Security>
</env:Header>
    <env:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd" wsu:Id="id-765320ef-6b29-4523-82fa-
9dfb75907be2">
        <d2:payload xmlns:com="http://datex2.eu/schema/3/common"
xmlns:d2="http://datex2.eu/schema/3/d2Payload"
xmlns:d1m="http://datex2.eu/schema/3/laneManagement"
xmlns:loc="http://datex2.eu/schema/3/locationReferencing"
xmlns:tro="http://datex2.eu/schema/3/trafficRegulation"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" lang="en-US"
modelBaseVersion="3"
xsi:schemaLocation="http://datex2.eu/schema/3/d2Payload
DATEXII_3_D2Payload.xsd" xsi:type="d1m:LaneManagementPublication">
            <com:publicationTime>2006-05-04T18:13:51.0</com:publicationTime>
            <com:publicationCreator>
                <com:country>it</com:country>
                <com:nationalIdentifier>IT403</com:nationalIdentifier>
            </com:publicationCreator>
            <d1m:laneManagementSettingDescription
xsi:type="d1m:LaneManagementSetting">

```

Test Assertions and exploitation suggestions

TA id	WSIBB-TA-1
Normative Source:	The SOAP Envelope SHALL remain a valid SOAP envelope even after encryption or signature: methods other than those defined by the WS-Security are not allowed
Target	The SuC in its test environment
Prerequisite	A sample message for each given service
Prescription Level	Mandatory
Predicate	For each sample message, check if it is compliant with the SOAP specifications, both 1.1 and 1.2
Variable	

This TA does prescribe that, if using SOAP, Signature and Encryption shall be done using WS-Security only. No CaDES, or ASiC should be performed, unless the SOAP message is ancillary (e.g., is an attachment to other, undefined, transaction, for which ASiC can be implemented).

An easy way to test the TA is to Marshall the content of the SOAP message into a class using a library. For instance, Java has the class “SOAPMessage”, which represents a compliant implementation of the SOAP 1.1 and 1.2 protocol.

TA id	WSIBB-TA-2
Normative Source:	The elements in the security header of the SOAP envelope SHALL be ordered in the way the clients are able to process it
Target	The SuC in its test environment
Prerequisite	A sample message for each given service
Prescription Level	Mandatory
Predicate	For each sample message, check if the order of the elements in the SOAP Header are ordered according to this IBB
Variable	

This TA prescribe the test for the precise ordering of the SOAP Header, which can be of importance for some streaming XML parsers. The test requires a visit over the XML tree.

TA id	WSIBB-TA-3
Normative Source:	The SOAP Envelope SHOULD contain a Timestamp element containing just exactly one Created and one Expired element. The timestamp SHOULD be signed
Target	The SuC in its test environment
Prerequisite	A sample message for each given service
Prescription Level	Optional
Predicate	For each sample message, check the content of the Timestamp header
Variable	Only applicable if the timestamp is not signed

No comments are available for the previous TA.

TA id	WSIBB-TA-4
Normative Source:	The SOAP Header, after applying this building block, SHALL be compliant with the WSI-BSP v1.1
Target	The SuC in its test environment
Prerequisite	A sample message for each given service
Prescription Level	Mandatory
Predicate	For each sample message, check the content of the message for compliance with WSI Basic Security Profile
Variable	

This TA requires compliance with the WS-I Basic Security Profile. A simple test of the activity could be to use consolidated libraries (e.g., WSS4J) that can impose such compliance.

TA id	WSIBB-TA-5
--------------	------------

Normative Source:	When using SAML as token in the header, the OWASP Cheat Sheet SHALL be considered. When relying on digital signature, out-of-band channels for schema, keys, and certificates SHALL be used.
Target	The SuC in its test environment
Prerequisite	A sample assertion
Prescription Level	Mandatory
Predicate	For each sample assertion, it is required to go over the Cheat Sheet and check if all the requirements are met. Moreover, the documentation used to exchange schema, keys, and certificates should be available, and it should contain a procedure, based on the local context, that permits the exchange of such artifacts, securely.
Variable	

No additional comment is provided.

TA id	WSIBB-TA-6
Normative Source:	The HashFunction SHALL be set to http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 , while the Signature algorithm SHALL be set to http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 .
Target	The SuC in its test environment
Prerequisite	A sample assertion, or a sample envelope
Prescription Level	Mandatory
Predicate	For each sample assertion or envelope, it is necessary to check the values of the algorithms set in the metadata
Variable	

No additional comment is provided.

TA id	WSIBB-TA-7
Normative Source:	A Schema Validation SHALL be performed every time a new incoming message is received
Target	The SuC in its test environment
Prerequisite	A set of endpoints
Prescription Level	Mandatory
Predicate	For each endpoint it is required to send syntactically wrong messages to observe the schema validation. Evidence can be provided in terms of logs.
Variable	

Provenance Data Sharing IBB

Name

Provenance Data Sharing IBB – PDSIBB

Rationale

To enable provenance tracking for data, this IBB will provide a provenance *template* which will be used to track a generic data manipulation and transfer across several actors (named Data Sources) and the destination⁵².

This IBB defines an XML Template based on W3C PROV⁵³ and a set of actions to be tracked. The activities tracked defined by this IBB are as follows:

- ex:CREATE, the creation of data
- ex:MODIFIED, the modification of data

The following agent types are defined.

- 1, a sensor on the road, unattended
- 2, a secure application (according with NAABB)
- 3, a secure node (according with NAABB)
- 4, a NAP

It is assumed that all data is signed according with DSABB. For this reason, the data provenance of the exchange between subsequent hops is not tracked. Note that the digital signature form of the document (e.g., with its canonicalization) is considered to be the *canonical form* of the data.

The following table defines the provenance template for the ex:CREATE action.

Provenance Element	Optionality	Usage Convention
Version	R	It is the integer “1” indicating this IBB
UUID	R	An Unique Universal ID encoded as per rfc 4122
entity	R	<i>It contains the data about the entity that is involved in the action.</i>
@id	R	It contains the string “thedata”
label	R	It contains the human readable information about the provenance data being tracked, e.g., Situation Publication,
location	R	It contains the location, human readable, where the data has been tracked by the action
type	R	It contains the fixed string “XML”
value	R	It contains the SHA256 hash of the data, put in its canonical form
activity	R	<i>It contains the information about the activity being done</i>
@id	R	It contains the string “theobjectcreation”
type	R	It contains the string ex:CREATE
agent	R	<i>It contains the information about the agents that created the data</i>
@id	R	It contains the string “agentidentifier”
type	R	It contains the type of agent who performed the activity. It can be an integer between 1 and 4.
agentcertificate	O	For agent types 2-4, it contains the full certificate chain identifying the agent
KeyInfo	R	It contains the x509 data required to identify the agent

⁵² Notably, part of this work is obtained from M. Masi et al., *Decentralised provenance for healthcare data*, International Journal of Medical Informatics, Volume 141, 2020, Elsevier.

⁵³ See <https://www.w3.org/TR/prov-overview/>

		X509Data	R	It contains the certificates as a list (Maximum 6 elements) containing the leaf cert first
		X509Certificate	R	The PEMformat of the x509 certificate identifying the agent
wasGeneratedBy			R	<i>It contains the information about the relation wasGeneratedBy. Its semantics is to identify the agent that did the activity, when</i>
	entity		R	The link to the entity id
	@Id		R	The string "thedata"
	activity			
	@Id		R	The string "theobjectcreation"
	time		R	The date when the relation has took place in ISO 8601
wasAssociatedWith			R	<i>It contains the information that associates the activity to the agent</i>
	activity		R	It contains the string "theobjectcreation"
	agent		R	It contains the string "agentidentifier"
wasAttributedTo			R	<i>It contains the information that associates the entity to the agent</i>
	agent		R	It contains the string "agentidentifier"
	entity		R	It contains the string "thedata"

Table 12 Definition of provenance for the CREATE activity

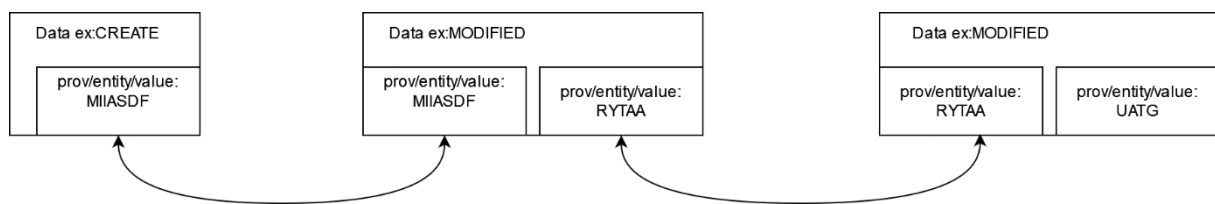
Similarly, the ex:MODIFIED activity tracks the change to a previous format performed to data. The provenance template tracks the activity in the following format.

Table 13 Definition of provenance for the MODIFIED activity

Provenance Element				Optionality	Usage Convention
Version				R	It is the integer “1” indicating this IBB
UUID				R	An Unique Universal ID encoded as per rfc 4122
entity				R	<i>It contains the data about the entity that is involved in the action. This is the original data</i>
	@id			R	It contains the string “thedata”
	label			R	It contains the human readable information about the provenance data being tracked, e.g., Situation Publication,
	location			R	It contains the location, human readable, where the data has been tracked by the action
	type			R	It contains the fixed string “XML”
	value			R	It contains the SHA256 hash of the data, put in its canonical form
entity				R	<i>It contains the data about the entity that is involved in the action. This is the modified data.</i>
	@id			R	It contains the string “themodifieddata”
	label			R	It contains the human readable information about the provenance data being tracked, e.g., Situation Publication,
	location			R	It contains the location, human readable, where the data has been tracked by the action
	type			R	It contains the fixed string “XML”
	value			R	It contains the SHA256 hash of the modified data, put in its canonical form
activity				R	<i>It contains the information about the activity being done</i>
	@id			R	It contains the string “theobjectmodification”
	type			R	It contains the string ex:MODIFIED
agent				R	<i>It contains the information about the agents modified the data</i>
	@id			R	It contains the string “agentidentifier”
	type			R	It contains the type of agent who performed the activity. It can be an integer between 1 and 4.
	agentcertificate			O	For agent types 2-4, it contains the full certificate chain identifying the agent
		KeyInfo		R	It contains the x509 data required to identify the agent
			X509Data	R	It contains the certificates as a list (Maximum 6 elements) containing the leaf cert first
			X509Certificate	R	The PEMformat of the x509 certificate identifying the agent
wasGeneratedBy				R	<i>It contains the information about the relation wasGeneratedBy. Its semantics is to identify the agent that did the activity, when</i>
	entity			R	The link to the entity id
		@Id		R	The string “thedata”
	activity				
		@Id		R	The string “theobjectmodification”
	time			R	The date when the relation has took place in ISO 8601
wasAssociatedWith				R	<i>It contains the information that associates the activity to the agent</i>

	activity	R	It contains the string "theobjectmodification"
	agent	R	It contains the string "agentidentifier"
used		R	It contains the information of the data that has been used to create the new data
	agent	R	It contains the string "theobjectmodification"
	entity	R	It contains the string "thedata"
wasDerivedFrom		R	It contains the information about the fact that the modified data is coming from the original data
	generatedEntity	R	It contains the string "themodifieddata"
	usedEntity	R	It contains the string "thedata"

Query lookup through the provenance documents, is done over the `document/entity/value` values, as shown in the following picture.



In the picture, data is created, signed, and the signature value is MIIASDF. The provenance record is persisted with the relevant agent. Data is then manipulated, and it is recorded both old signature value, and the new one. From now on, the provenance documents form a linked list.

The provenance service

The provenance documents can be either shared along the data or can be persisted in database and then shared through services. This IBB specifies an endpoint to be used to centrally store ex:CREATE and ex:MODIFIED provenance records.

The service SHALL be available as the endpoint `https://<fqdn>/provenance/` and it is contacted through GET and POST, and it SHALL implement the NODEABB with no certificate pinning and no mutual TLS.

The search parameters are the following (using HTTP GET):

Name	Type	Description	Example
entity.hash	string	The hash of the provenance record of the current item (e.g., the value of <code>prov/entity/value</code>)	<code>entity.hash=MIIASDF</code>
entity1.hash	string	The hash of the provenance record of the modified item	<code>entity1.hash=RYTAA</code>
recursive	boolean	Specify whether to return all the values of the linked list up to the root (e.g., the ex:CREATE event)	<code>recursive=true</code>
agent-type	integer	Returns all the documents for a given agent type	<code>agent-type=1</code>
agent-sdn	string	Returns all the documents for a given Agent Subject Distinguished Name in its leaf certificate	<code>agent-sdn=CN%20%3D%20datex2</code>

The request/response protocol SHALL be defined according to the following tables, differentiated for the provenance transaction. For the `ex:CREATE`, the client SHALL send the following data, in JSON format to the base endpoint `https://<fqdn>/provenance/create`. Notably, it SHALL be a HTTP POST with the value `Content-Type: application/json`.

Name	Type	Description	Example
entity.label	string	The human readable value that will be recorded as <code>prov/entity/label</code>	<code>entity.label="Situation Publication"</code>
entity.location	string	The human readable value that will be recorded as <code>prov/entity/location</code>	<code>entity.location="Italian CCISS"</code>
entity.value	string	Specify the hash of the provenance document, that will be recorded as <code>prov/entity/value</code>	<code>entity.value="RYTAA"</code>
agent.type	integer	Contains the type of the agent that will be recorded as <code>prov/agent/type</code>	<code>agent.type=1</code>
agent.certificate	Array of strings	Contains all the certificates that will be recorded in <code>prov/agent/certificate</code>	<code>agent-sdn=CN%20%3D%20datex2</code>

The following is an example.

```
{
  "entity.label": "Situation Publication",
  "entity.location": "Italian NAP, CCISS",
  "entity.value": "RYTAA",
  "agent.type": 2,
  "agent.certificate": [
    "LeafCert",
    "MiddleCert",
    "TrustAnchor"
  ]
}
```

Similarly, the `ex:MODIFIED` will contain the additional entity, as per the following table.

Name	Type	Description	Example
entity.label	string	The human readable value that will be recorded as <code>prov/entity/label</code>	<code>entity.label="Situation Publication"</code>
entity.location	string	The human readable value that will be recorded as <code>prov/entity/location</code>	<code>entity.location="Italian CCISS"</code>
entity.value	string	Specify the hash of the provenance document, that will be recorded as <code>prov/entity/value</code>	<code>entity.value="RYTAA"</code>
entity1.label	string	The human readable value that will be recorded as <code>prov/entity1/label</code>	<code>entity1.label="Situation Publication"</code>
entity1.location	string	The human readable value that will be recorded as <code>prov/entity1/location</code>	<code>entity1.location="Italian CCISS"</code>
entity1.value	string	Specify the hash of the provenance document, that will be recorded as <code>prov/entity1/value</code>	<code>entity1.value="MIIASDF"</code>

agent.type	integer	Contains the type of the agent that will be recorded as	agent.type=1
agent.certificate	Array of strings	Contains all the certificates that will be recorded in	agent-sdn=CN%20%3D%20datex2

The following code is an example.

```
{
  "entity.label": "Situation Publication",
  "entity.location": "Italian NAP, CCISS",
  "entity.value": "RYTAA",

  "entity1.label": "Situation Publication",
  "entity1.location": "German NAP, Mobilithet",
  "entity1.value": "MIIASDF",
  "agent.type": 2,
  "agent.certificate": [
    "LeafCert",
    "MiddleCert",
    "TrustAnchor"
  ]
}
```

For the `ex:MODIFIED`, the client SHALL send the following data, in JSON format to the base endpoint <https://<fqdn>/provenance/modified>. Notably, it SHALL be a HTTP POST with the value `Content-Type: application/json`.

The server SHALL respond with HTTP 200 in case of successful operation, while it shall return the corresponding HTTP value in case of errors.

Requirements

Brief name	Description	Variability Point
PDS-IBB-1	If an <code>ex:MODIFIED</code> action is requested, the service SHALL check for the existence of a provenance record <code>ex:MODIFIED</code> or <code>ex:CREATE</code> with the value, otherwise it shall return HTTP 405.	
PDS-IBB-2	The provenance service SHOULD authorize the access to the subcontext of the base url, using TLSIBB or JWTIBB.	Contractual agreements and local context may override this value

Similar IBB and Dependencies

Provenance can be implemented also by using Evidence. See NRABB.

Known Use

N/A. Known use will be described in future versions of this document.

Technical Implementation

This is an example XML for the specification defined in Table 12. The schema can be obtained from W3⁵⁴.

```
<?xml version="1.0" encoding="UTF-8"?>
<prov:document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/ns/prov#
http://www.w3.org/ns/prov.xsd" xmlns:ex="urn:d2:security:provenance"
xmlns:prov="http://www.w3.org/ns/prov#">
  <ex:version>1</ex:version>
  <ex:uuid>urn:uuid:...</ex:uuid>
  <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:id="thedata">
    <prov:label>Data (e.g. SituationPublication)</prov:label>
    <prov:location>ITALY_NAP</prov:location>
    <prov:type>XML</prov:type>
    <prov:value>S52fkpF2rCEArSuwqyDA9tVjawUdrkGzbNQLaa7xJfA=</prov:value>
  </prov:entity>
  <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:id="theobjectcreation">
    <prov:type>ex:CREATE</prov:type>
  </prov:activity>
  <prov:agent xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:id="agentidentifier">
    <prov:type>1.2.3.4</prov:type>
    <ex:agentcertificate>
      <ds:KeyInfo>
        <ds:X509Data>
          <!--Signer cert, issuer CN=arbolCA,OU=FVT,O=IBM,C=US, serial 4-->
          <X509Certificate>MIICXTCCA..</X509Certificate>
          <!-- Intermediate cert subject CN=arbolCA,OU=FVT,O=IBM,C=US
issuer CN=tootiseCA,OU=FVT,O=Bridgepoint,C=US -->
          <X509Certificate>MIICPzCCA...</X509Certificate>
          <!-- Root cert subject CN=tootiseCA,OU=FVT,O=Bridgepoint,C=US -->
          <X509Certificate>MIICSTCCA...</X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </ex:agentcertificate>
  </prov:agent>
  <prov:wasGeneratedBy>
    <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="theobject"/>
    <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="theobjectcreation"/>
    <prov:time>2023-11-10T12:15:55.028Z</prov:time>
  </prov:wasGeneratedBy>
  <prov:wasAssociatedWith>
    <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="theobjectcreation"/>
```

⁵⁴ See <https://www.w3.org/TR/2013/NOTE-prov-xml-20130430/>

```

    <prov:agent xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="agentidentifier"/>
  </prov:wasAssociatedWith>
  <prov:wasAttributedTo>
    <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="theobject"/>
    <prov:agent xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="agentidentifier"/>
  </prov:wasAttributedTo>
</prov:document>

```

Similarly, the following is an example of XML for the ex:MODIFIED provenance data.

```

<?xml version="1.0" encoding="UTF-8"?>
<prov:document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/ns/prov#
http://www.w3.org/ns/prov.xsd" xmlns:ex="urn:d2:security:provenance"
xmlns:prov="http://www.w3.org/ns/prov#">
  <ex:version>1</ex:version>
  <ex:uuid>urn:uuid:...</ex:uuid>
  <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:id="thedata">
    <prov:label>Data (e.g. SituationPublication)</prov:label>
    <prov:location>ITALY_NAP</prov:location>
    <prov:type>XML</prov:type>
    <prov:value>S52fkpF2rCEArSuwqyDA9tVjawUdrkGzbNQLaa7xJfA=</prov:value>
  </prov:entity>
  <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:id="themodifieddata">
    <prov:label>Data (e.g. SituationPublication)</prov:label>
    <prov:location>ITALY_NAP</prov:location>
    <prov:type>XML</prov:type>
    <prov:value>S52fkpF2rCEArSuwqyDA9tVjawUdrkGzbNQLaa7xJfA=</prov:value>
  </prov:entity>

  <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:id="theobjectcreation">
    <prov:type>ex:MODIFIED</prov:type>
  </prov:activity>
  <prov:agent xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:id="agentidentifier">
    <prov:type>1.2.3.4</prov:type>
    <ex:agentcertificate>
      <ds:KeyInfo>
        <ds:X509Data>
          <!--Signer cert, issuer CN=arbolCA,OU=FVT,O=IBM,C=US, serial 4-->
          <X509Certificate>MIICXTCCA..</X509Certificate>
          <!-- Intermediate cert subject CN=arbolCA,OU=FVT,O=IBM,C=US
issuer CN=tootiseCA,OU=FVT,O=Bridgepoint,C=US -->
          <X509Certificate>MIICPzCCA...</X509Certificate>
          <!-- Root cert subject CN=tootiseCA,OU=FVT,O=Bridgepoint,C=US -->
          <X509Certificate>MIICSTCCA...</X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </ex:agentcertificate>
  </prov:agent>

```

```

<prov:wasGeneratedBy>
  <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="theobject"/>
  <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="theobjectcreation"/>
  <prov:time>2023-11-10T12:15:55.028Z</prov:time>
</prov:wasGeneratedBy>
<prov:wasAssociatedWith>
  <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="theobjectcreation"/>
  <prov:agent xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="agentidentifier"/>
</prov:wasAssociatedWith>
<prov:used>
  <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="thedatamodification"/>
  <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="thedata"/>
</prov:used>
<prov:wasDerivedFrom>
  <prov:generatedEntity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="thedatamodified"/>
  <prov:usedEntity xmlns:ns1="http://www.w3.org/ns/prov#"
ns1:ref="thedata"/>
</prov:wasDerivedFrom>
</prov:document>

```

Test Assertions and exploitation suggestions

TA id	PDS-IBB-1
Normative Source:	If an ex:MODIFIED action is requested, the service SHALL check for the existence of a provenance record ex:MODIFIED or ex:CREATE with the value, otherwise it shall return HTTP 405.
Target	The SuC in its test environment
Prerequisite	Data is already created, and its provenance record is available.
Prescription Level	Mandatory
Predicate	Data is manipulated and its provenance ancestor is queried
Variable	

TA id	PDS-IBB-2
Normative Source:	The provenance service SHOULD authorize the access to the subcontext of the base url, using TLSIBB or JWTIBB.
Target	The SuC in its test environment

Prerequisite	Data is already created, and its provenance record is available. An authorization mechanism is available
Prescription Level	Optional
Predicate	Data is manipulated and its provenance record is available. The provenance record is queried using the schema provided in this building block. An authorization mechanism is tested in order to have both permit or deny
Variable	In case the system implements it.

Digital Signature IBB

Name

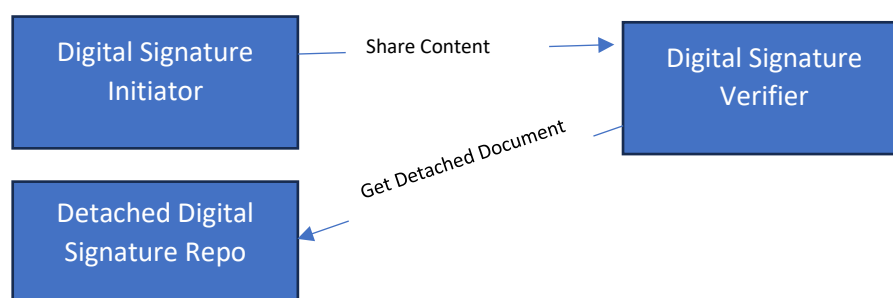
Digital Signature IBB – DSIBB

Rationale

This building block describes a general-purpose method based on a set of free standards, tools, and services aimed at facilitating the secure exchange of messages by enforcing such concepts as integrity and authentication. This building block provide the following services:

- Reference to the standards for eSignature (*aDES), to facilitate business transactions across public administrations and private sector guaranteeing cross-border legal assurance through the eIDAS regulation (EU 910/2014)⁵⁵
- Inline signature for XML documents through XML DSG⁵⁶
- Inline signature for generic objects through CMS⁵⁷
- Detached signature document, that can be downloaded ex-post or ex-ante via a defined endpoint
- Support for long-term signatures
- It does not describe “wet signatures” over documents.

The IBB is based on the following services.



⁵⁵ See, e.g., <https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Standards+and+specifications>

⁵⁶ See, e.g., <https://www.w3.org/TR/xmlsig-core/>

⁵⁷ See, e.g., <https://www.rfc-editor.org/rfc/rfc8933>

A *digital signature initiator* (DSI) is a service who has content at rest to share, in any form. The content is retrieved by a *digital signature verifier* (DSV) who needs to guarantee the authentication and the integrity of the data shared. In some case the DSV may use this IBB to guarantee a weak fair form of non-repudiation (see NRABB), as if the digital signature enjoys authentication the DSI cannot refuse to have originated the message.

In some cases, the digital signature may not be present in the message, but it could be *detached*, meaning that it is contained in a separate document and retrieved when the signature verification process takes place. Examples may include sending messages small in size that could not contain signature, for which the signature verification is made independently by the content documents; thus, there is a risk they will be made unavailable through revision or access control.

Requirements

Brief name	Description	Variability Point
DSIBB-1-T	The XML content MAY conform to the XAdES schema for signatures as defined in: - ETSI EN 319 132 Part 1, and Part 2 OR ETSI TS 101 903 and ETSI TS 103 171 “Electronic Signature and Infrastructures” - This requirement does not further constrain any requirement as defined by the CEF eSig BB⁵⁸	If the content is in form of XML, and if the signature enjoys legal value
DSIBB-2-T	Non-XML content MAY conform to CAdES technical specifications as defined in - ETSI EN 319 122 CMS Advanced Electronic Signatures (CAdES) Part 1, and Part 2 - This requirement does not further constrain any requirement as defined by the CEF eSig BB	If the content is in form of object (neither XML or PDF) and the signature enjoys legal value
DSIBB-3-T	PDF content MAY be signed using PAdES as defined in - ISO 32000-1, ETSI TS 102 778-1, ETSI TS 102 778-2, ETSI TS 102 778-3, ETSI TS 102 778-4, and ETSI TS 103 172 - This requirement does not further constrain any requirement as defined by the CEF eSig BB	If the content is in form of PDF and the signature enjoys legal value
DSIBB-4-T	The Digital Signature block (either enveloping or detached) - SHALL use the hash algorithm sha256 - SHALL use the canonicalization algorithm “Canonical XML 1.1 with Comments” (http://www.w3.org/2006/12/xml-c14n11#WithComments) - MAY use the policy identifier as urn:napcore:detached:2023 if the signature is Detached or urn:napcore:enveloping:2023 if the signature is enveloping.	

⁵⁸ See, e.g., <https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Standards+and+specifications>

	• SHOULD include a CommitmentTypeIndication element for the Purpose of Signature. Projects MAY agree on vocabulary, e.g., ETSI TS 119 172-1 V1.1.1. At minimum the following purposes SHALL be supported ○ EMERGENCY, in case the content contain situation where the quality of service requires immediate attention ○ NORMAL_OPERATION, in case the content should follow normal operations	
DSIBB-5-T	The DSI should make available the detached signature document through the following endpoint. Grouping with CPIBB is REQUIRED. <a href="https://<fqdn>/<ctx>/dsi/retrieveDetachedSignature">https://<fqdn>/<ctx>/dsi/retrieveDetachedSignature where • <fqdn> is the fully qualified domain name • <ctx> is any given context The request SHALL be a GET containing the following parameters • Id, a multivalue, comma-separated, string containing the identifier of the document for which the signature is to be verified.	
DSIBB-6-T	The certificate SHALL have: • KeyUsage bits (rfc5280): digitalSignature (asserted), nonrepudiation/contentCommitment (asserted), • Basic Constraints: the certificate SHALL not have the CA bit asserted • Extended Key Usage: according with rfc 5280, "if the extendend key usage is present, then the certificate MUST only be used for one of the purposes indicated". The KeyPurposeId SHALL BE id-kp-documentSigning (rfc 9336) 1.3.6.1.5.5.7.3.36. Other certificate KeyUsage SHALL NOT be permitted.	
DSIBB-7-T	The certificate verifier SHALL perform the checks on the KeyUsage, BasicConstraints, and ExtendedKeyUsage as defined in DSIBB-6-T	

Similar IBB and Dependencies

The DEIBB is a similar IBB that may provide message signature over encryption, and it uses this IBB to provide signature details.

The CPIBB is required whenever a signature is used to be authenticated, thus the signer information is used by comparing with the expected certificate stored out of band.

Technical Implementation

In order to implement DSIBB-5-T, the unique identifier of the document is its DigestValue after performing the canonicalization algorithms.

Particular attention shall be given to the implementation which perform manipulation to the data (e.g., indentation to XML, or changing endianness), as it may break the signature.

How to create a CaDES signature which is eiDAS compliant.

```
openssl cms -nosmimecap -md sha256 -nodetach -binary -cades -stream -
outform DER -sign -signer demoCA/clientcert.pem -inkey demoCA/clientkey.pem
-in tbsDocument.txt -out signedDocument.txt
```

The following is an example of XaDES detached signature.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Signature Id="urn:oid:1.2.3.4"
xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315#WithComments"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
    <Reference Type="http://www.w3.org/2000/09/xmldsig#Manifest"
URI="#NAPCOREManifest">
      <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <DigestValue>FPekeMbf3zvJcPuRIzt9jMgLA2MVbgMkHtoQLjcuHMU=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>ZGxcXfZw</SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509Certificate>MIICs</X509Certificate>
    </X509Data>
  </KeyInfo>
  <Object Id="object">
    <xad:QualifyingProperties Target="#NAPCOREManifest"
xmlns:xad="http://uri.etsi.org/01903/v1.3.2#">
      <xad:SignedProperties>
        <xad:SignedSignatureProperties>
          <xad:SigningTime>2023-09-15T18:35:07.417+02:00</xad:SigningTime>
          <xad:SigningCertificate>
            <xad:Cert>
              <xad:CertDigest>
                <ds:DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                  <ds:DigestValue
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">Ve9ESrQ1jfiTuTSE+0Bbn5XNfiU=<
/ ds:DigestValue>
                </xad:CertDigest>
              <xad:IssuerSerial>
```

```

    <ds:X509IssuerName
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">C=AU,ST=Some-State,O=Internet
Widgits Pty Ltd</ds:X509IssuerName>
    <ds:X509SerialNumber
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">16950076479900136087</ds:X509
SerialNumber>
    </xad:IssuerSerial>
  </xad:Cert>
</xad:SigningCertificate>
<xad:SignaturePolicyIdentifier>
  <xad:SignaturePolicyId>
    <xad:SigPolicyId>
      <xad:Identifier>urn:napcore:enveloping:2023</xad:Identifier>
    </xad:SigPolicyId>
    <xad:SigPolicyHash>
      <ds:DigestMethod
Algorithm="http://www.w3.org/2001/04/xmenc#sha256"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
        <ds:DigestValue
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">P92j5890EzVuc5ciSQXjQPt63+Q=<
/ ds:DigestValue>
      </xad:SigPolicyHash>
    </xad:SignaturePolicyId>
  </xad:SignaturePolicyIdentifier>
</xad:SignedSignatureProperties>
<xad:SignedDataObjectProperties>
  <xad:CommitmentTypeIndication>
    <xad:CommitmentTypeId>
      <xad:Identifier>NORMAL_OPERATION</xad:Identifier>
    </xad:CommitmentTypeId>
    <xad:AllSignedDataObjects/>
  </xad:CommitmentTypeIndication>
</xad:SignedDataObjectProperties>
</xad:SignedProperties>
</xad:QualifyingProperties>
<Manifest Id="NAPCOREManifest">
  <Reference URI="urn:oid:1.2.3.4.5.6.7.8">
    <Transforms>
      <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
    </Transforms>
    <DigestMethod
Algorithm="http://www.w3.org/2001/04/xmenc#sha256"/>
      <DigestValue>ZS/RHCdXoTMpNvm0IhfshiI7DIE=</DigestValue>
    </Reference>
    <Reference URI="urn:oid:1.5.6.7.9">
      <!--Reference to all the detached documents -->
    </Reference>
  </Manifest>
</Object>
</Signature>

```

Open Points

- 1) When performing CMS encoding, should we use MIME or PKCS7? If SMIME, should we prescribe a content-type?



- 2) Should we claim eIDAS compliance, thus should the requirements defined in this IBB reference the eIDAS technical implementation?

Test Assertions and exploitation suggestions

TA id	DSIBB-1-T
Normative Source:	The XML content MAY conform to the XAdES schema for signatures as defined in: ETSI EN 319 132 Part 1, and Part 2 OR ETSI TS 101 903 and ETSI TS 103 171 “Electronic Signature and Infrastructures” This requirement does not further constrain any requirement as defined by the CEF eSig BB
Target	The SuC in its test environment
Prerequisite	Data is already created and signed
Prescription Level	Optional
Predicate	The signature data is checked against the prescription of the ETSI standard. In particular it is mandatory that at least the XAdES schema is validated. Eventual tests available from the CEF eSig BB or ETSI PLUG (e.g. TS 119 134-2) test may be considered
Variable	In case the system implements it.

TA id	DSIBB-2-T
Normative Source:	Non-XML content MAY conform to CAdES technical specifications as defined in ETSI EN 319 122 CMS Advanced Electronic Signatures (CAdES) Part 1, and Part 2 This requirement does not further constrain any requirement as defined by the CEF eSig BB
Target	The SuC in its test environment
Prerequisite	Data is already created and signed
Prescription Level	Optional
Predicate	Tests shall be evaluated according with TS 119 124-4
Variable	In case the system implements it.

TA id	DSIBB-3-T
Normative Source:	PDF content MAY be signed using PAdES as defined in ISO 32000-1, ETSI TS 102 778-1, ETSI TS 102 778-2, ETSI TS 102 778-3, ETSI TS 102 778-4, and ETSI TS 103 172 This requirement does not further constrain any requirement as defined by the CEF eSig BB

Target	The SuC in its test environment
Prerequisite	Data is already created and signed
Prescription Level	Optional
Predicate	Tests shall be evaluated according with TS 119 144-4
Variable	In case the system implements it.

TA id	DSIBB-4-T
Normative Source:	The Digital Signature block (either enveloping or detached) • SHALL use the hash algorithm sha256 • SHALL use the canonicalization algorithm “Canonical XML 1.1 with Comments” (http://www.w3.org/2006/12/xml-c14n11#WithComments) • MAY use the policy identifier as urn:napcore:detached:2023 if the signature is Detached or urn:napcore:enveloping:2023 if the signature is enveloping. • SHOULD include a CommitmentTypeIndication element for the Purpose of Signature. Projects MAY agree on vocabulary, e.g., ETSI TS 119 172-1 V1.1.1. At minimum the following purposes SHALL be supported ○ EMERGENCY, in case the content contain situation where the quality of service requires immediate attention ○ NORMAL_OPERATION, in case the content should follow normal operations
Target	The SuC in its test environment
Prerequisite	Data is already created and signed
Prescription Level	Mandatory
Predicate	The test will check the syntax of the elements in the data structure
Variable	

TA id	DSIBB-5-T
Normative Source:	The DSI should make available the detached signature document through the following endpoint. Grouping with CPIBB is REQUIRED. <a href="https://<fqdn>/<ctx>/dsi/retrieveDetachedSignature">https://<fqdn>/<ctx>/dsi/retrieveDetachedSignature where

	• <fqdn> is the fully qualified domain name • <ctx> is any given context The request SHALL be a GET containing the following parameters Id, a multivalue, comma-separated, string containing the identifier of the document for which the signature is to be verified.
Target	The SuC in its test environment
Prerequisite	Data is already created and signed, and the detached signature is available in the indicated service. A PKI is serving the certificate for the TLS connection
Prescription Level	Optional
Predicate	Data is canonicalized and digested. Let $H = \text{HASH}(\text{c14n}(\text{data}))$. The SuC should expose a service using the following endpoint: <a href="https://<fqdn>/<ctx>/dsi/retrieveDetachedSignature?Id=H">https://<fqdn>/<ctx>/dsi/retrieveDetachedSignature?Id=H . The returned certificate SHALL be pinned
Variable	In case the system implements it.

TA id	DSIBB-6-T
Normative Source:	The certificate SHALL have: • KeyUsage bits (rfc5280): digitalSignature (asserted), nonrepudiation/contentCommitment (asserted), • Basic Constraints: the certificate SHALL not have the CA bit asserted • Extended Key Usage: according with rfc 5280, "if the extendend key usage is present, then the certificate MUST only be used for one of the purposes indicated". The KeyPurposeId SHALL BE id-kp-documentSigning (rfc 9336) 1.3.6.1.5.5.7.3.36. Other certificate KeyUsage SHALL NOT be permitted.
Target	The SuC in its test environment
Prerequisite	Data is already created and signed, and signing certificates are available
Prescription Level	Mandatory
Predicate	The test is performed checking the information in the certificate
Variable	

TA id	DSIBB-7-T
Normative Source:	The certificate verifier SHALL perform the checks on the KeyUsage, BasicConstraints, and ExtendedKeyUsage as defined in DSIBB-6-T
Target	The SuC in its test environment
Prerequisite	Data is already created and signed, and signing certificates are available
Prescription Level	Mandatory
Predicate	The test is performed by submitting signed data and signature structure to the systems. The system should pass a valid signature and should reject an invalid signature.
Variable	

Audit Syntax IBB

Name

Audit Syntax IBB – ASIBB

Rationale

The standardization of the syntax of logs and audit trail is crucial to achieve interoperability in several ways. Forensic analysis, for instance, may leverage a common set of audit trail with the same syntax shared in a distributed setting. This IBB is intended to provide a surveillance logging function. In particular it captures all security events that are detected. Notably, logs provided by this IBB are not intended for debugging purpose.

The format of the audit trail is JSON, and the format is as follows.

Log Element		Optionality	Usage Convention
AuditEventId		R	Unique ID according with rfc 4122
role		R	The role, e.g., server or client
category		R	Contains the system category, i.e., the event being performed
	coding.system	R	Contains the string urn_napcore_audit_2023
	coding.code	R	Contains the string according with the following table 110000, Physical Access 110001, Login 110002, Logout 110003, Access Control Decision 110004, Backup 110005, Restore 110006, Configuration Change 110007, Potential Reconnaissance activity (e.g., nmap scan) 110008, Log System Events, log system start 110009, Log System Events, log system stop 110010, TLS Certificate Usage 110011, Rekeying 110012, Session Management Failures
	coding.display	O	Contains the description of the audit
recorded		R	Contain the event time in ISO 8601
outcome		R	Contains the information about the outcome of the event
	system	O	Contains the string urn_napcore_audit_2023
	code	O	It contains the string according with the following table 111000, SUCCESS 111001, FAILURE 111002, MAJOR_FAILURE
	display	O	Contains the description of the event



severity	O	Contains the information about the severity of the event
system	O	Contains the string urn_napcore_audit_2023
code	O	It contains the string according with the following table 112000, EMERGENCY 112001, ALERT 112003, INFO
display	O	Contains the description of the event
who	R	Contains the information about who did the event
system	O	Contains the string urn_napcore_audit_2023
code	O	It contains the string according with the following table 113000, Indicates that the description contains the application identification (e.g., "Dell Computer") 113001, Indicates that the description contains the hostname 113002, Indicates that the description contains the ip address 113003, Indicates that the description contains the Subject Alternative Name
display	O	Contains the description of the event according with the code
idp	O	Contains the human readable name of the identity provider
destination	R	Contains the information about the destination of the event
system	O	Contains the string urn_napcore_audit_2023
code	O	It contains the string according with the following table 113000, Indicates that the description contains the application identification (e.g., "Dell Computer") 113001, Indicates that the description contains the hostname 113002, Indicates that the description contains the ip address 113003, Indicates that the description contains the Subject Alternative Name
display	O	Contains the description of the event according with the code
origin	R	Contains the information about the origin of the event
system	O	Contains the string urn_napcore_audit_2023
code	O	It contains the string according with the following table 113000, Indicates that the description contains the application identification (e.g., "Dell Computer") 113001, Indicates that the description contains the hostname 113002, Indicates that the description contains the ip address 113003, Indicates that the description contains the Subject Alternative Name
display	O	Contains the description of the event according with the code

The following contains an example of the audit trail

```
{
  "AuditEventId": "b0b40be1-c7ad-4724-8676-9f42d5026347",
  "role": "server",
  "category": [
    {
      "coding": [
        {
          "system": "urn_napcore_audit_2023",
          "code": "110000",
          "display": "Physical Access"
        }
      ]
    }
  ]
}
```

```

],
"recorded":"2022-10-31T09:00:00.594Z",
"outcome":[
  {
    "system":"urn_napcore_audit_2023",
    "code":"111000",
    "display":"SUCCESS"
  }
],
"severity":[
  {
    "system":"urn_napcore_audit_2023",
    "code":"112002",
    "display":"INFO"
  }
],
"who":[
  {
    "system":"urn_napcore_audit_2023",
    "code":"113001",
    "display":"mmas@autostrade.it",
    "idp":"NAPCORE IdP"
  }
],
"destination":[
  {
    "system":"urn_napcore_audit_2023",
    "code":"113001",
    "display":"192.168.1.1"
  }
],
"origin":[
  {
    "system":"urn_napcore_audit_2023",
    "code":"113001",
    "display":"192.168.1.2"
  }
]
}

```

Requirements

Brief name	Description	Variability Point
ATIBB-1-T	Each system SHALL have a list of implemented actions according with the category.coding.code event list.	
ATIBB-2-T	For each implemented service, an audit trail for both client and server SHALL be available.	

Similar IBB and Dependencies

No similar IBBs are relevant for this system. The ATIBB SHALL run after the implementation of CTABB.



Technical Implementation

The Audit Trail shall be sent to a log repository using a mechanism not further specified by this IBB. Use of syslog MAY be required in future. Sending an audit trail SHOULD NOT halt the transaction. Audit Trails are sent in best effort.

Test Assertions and exploitation suggestions

TA id	ATIBB-1-T
Normative Source:	Each system SHALL have a list of implemented actions according with the category.coding.code event list.
Target	The SuC in design phase
Prerequisite	n/a
Prescription Level	Required
Predicate	It shall be checked the existence of a list of implemented actions
Variable	

TA id	ATIBB-2-T
Normative Source:	For each implemented service, an audit trail for both client and server SHALL be available
Target	The SuC in test environment
Prerequisite	The list of implemented service is available for both client and server, and the systems are implemented
Prescription Level	Required
Predicate	After triggering all the transactions needed to reproduce all the services implemented, it shall be checked if the required artifacts are correctly available
Variable	

JWT IBB

Name

JWT IBB – JWTIBB

Rationale

This IBB implements Authorization on RESTful transactions by means of JSON Web Tokens requested using OAuth 2.0, without restricting the way how the token is obtained (e.g., using one of the available OAuth 2.0/2.1 methods, including PKCE). This building block provides a minimum set of attributes set of claims.

JWT is a compact means to represent claims transferred by two parties, encoded in JSON, and signed using JSON Web Signatures. Thus, a JWT is integrity protected, and can be authenticated (signed by the Authorization Server). In fact, when using a flow, OAuth 2.0 SHALL return a JSON web token.

Requirements

Brief name	Description	Variability Point
JWTIBB-1-T	The JSON Web Token SHALL be signed	

Similar IBB and Dependencies

No similar IBBs are available. This IBB SHALL BE implemented after the CTABB.

Technical Implementation

JWT token shall be signed as specified in JSON Web Signature [rfc7515]. If signed, the JWS Compact Serialization, shall be used as described in RFC7515. The following signature algorithms SHALL be supported:

- HS256: HMAC using SHA-256 hash algorithm
- RS256: RSA using SHA-256 hash algorithm

The following algorithms SHOULD be supported

- ES256: ECDSA using P-256 curve
- SHA-256 hash algorithms ("ES256").

Note: it is recommended to use asymmetric methods for signing access token to avoid the storage and hardcoding of sensitive symmetric keys.

The following claims MUST be present, and the values MUST be used by the Audit Trail SBB:

- *iss*: a single valued string identifying the instance which issued the access token.
- *sub*: if known, an unique identifier of the principal; the *client_id* otherwise
- *client_id*: identifier of the client for which the token is issued
- *aud*: an array of identifier strings for the resource server endpoints to be accessed [JWT Section 2.2]. In the special case when the JWT has one audience, the "aud" value may be a single case-sensitive string. In special environments, it may be a generic string identifying the "context"
- *jti*: a unique identifier for the JWT access token
- *exp*: Expiration time of the token in Numeric Date Format
- *scope*: (OPTIONAL) the scope granted by the Authorization server
- *nbf*: the earliest time in Numeric Date Format the token shall be accepted



- *iat*: The issuing date in Numeric Date format

An example of token is as follows (without the signature)

```
{
  "iss": "urn:vendor:sts",
  "sub": "b3ca1045-aa8b-42f9-9fd9-e0cbf5cb90a7",
  "aud": "http://openapi.datex/res1",
  "exp": 1438251487,
  "nbf": 1438251187,
  "iat": 1438251187,
  "scope": "PIM-DEFINED-TRANSACTION",
}
```

Test Assertions and exploitation suggestions

TA id	JWTIBB-1-T
Normative Source:	The JSON Web Token SHALL be signed
Target	The SuC in test environment
Prerequisite	The system produces a JWT, or it obtain from an IdP
Prescription Level	Required
Predicate	After triggering all the transactions needed to reproduce the token, it shall be verified the signature using ancillary services such as jwt.io.
Variable	

NAP Security Reference Architecture

The NAP Security Reference Architecture (SRA) is shown in Figure 17 The NAP Security Reference Architecture. The architecture is the evaluation of the relevant building blocks that are needed to guarantee the following objective of the NAP: to deliver authenticated data to authorised users, enabling tracking of the origin and the destination of data, and being able to solve disputes when they arise. Notably, the SRA is intended for both types of NAP: data provider or data broker.

The NAP SRA assumes the following:

- The existence of a Security Module (that can be, e.g., Hardware or Software), capable to securely store keys.
- An external PKI (not further described in this deliverable), capable to *Manage Key Pairs*
- An external NTP server (available from the public NTPs), capable to *Synchronise Time*
- An external *Authorization Server*, capable to *Authorize transaction*. Notably this authorization server can be either open source or commercial, such as, e.g., Redhat Keycloak, Okta, Ping Identity.
- An external *Remote Site*, which is the peer who is feeding data to the NAP or being brokered by the NAP, connected either via private network (e.g., VPN, MPLS VRF) or via public internet. This remote site connects to the *Server Application (Data In)*, a service exposed by the NAP.

- An external *Any Client*, which is the peer consuming the data (e.g., a remote app) connected through the public internet.
- Internal to the NAP, in addition to the Security Module⁵⁹, the following services are present
 - A Storage, for audit records, evidence, or detached signatures
 - Three services for storing evidence, provenance tracking records, and detached signatures
 - An NTP Client

The ABB used by the NAP SRA are the following.

- Node Authentication ABB by Remote Site and Any Client.
- Authorisation ABB by the Server Application (Data In) and Server Application. Notably, the Authorization Server shall be ex-ante shared with the Remote Site, but this agreement goes beyond the scope of this deliverable.
- Data Integrity ABB by the Server Application (Data In) and the Server Application.
- Data Provenance ABB by the Server Application (Data In) and the Server Application.
- Data Encryption ABB by the Server Application (Data In) and the Server Application.
- Non-Repudiation ABB by the Any Client. Notably, this is optional as the prescription on the data consumers goes beyond this specification, however the NAP SHOULD make available the evidence storage.
- Audit Trail ABB, by all the NAP services, in particular used by the Server Application (Data In) and the Server Application.
- Consistent Time ABB, used by all actors.

The implementation of the ABBs SHALL be done by the following IBBs.

- TLS IBB: all the communications between the Remote Site and Any Client SHALL be done using the specifications made by TLS IBB without certificate pinning but with mutual authentication, to guarantee the authenticity of the clients and the NAPs' endpoints.
- Provenance and Data Sharing IBB: provenance tracking SHALL be implemented using the prescriptions of the IBB
- Digital Signature IBB: data SHOULD be signed either inline or detached
- Audit Syntax IBB: NAP SHALL provide logs according to the syntax defined
- JWT IBB: the authorisation of the transactions, in REST format, SHALL be made using the JWT format described.

Conceptually, the NAP SHALL perform the following tasks at start time.

- Synchronise the time with a NTP server
- Assure that all the needed keys are available in the Security Module

The flow is as follows. A remote site obtains an authorization token and then, *securely*, connects to the server application (data in) of the NAP, implementing Secure Data Exchange. If the communication channel is established (e.g., TLS handshake success), the server application introspects the received authorisation token (the JWT) to authorise the transaction. The Server Application (Data In) writes a corresponding audit trail containing category.coding.code 11010 "TLS Certificate Use" and role as

⁵⁹ Notably, the Security Module can also be external to the NAP itself.

“server”. Eventually, if the data is signed, the Server Application (Data In) checks the validity of the signature, and in case of Detached signature, it stores in the Storage of the NAP.

When the Any Client access data, this is done similarly: the channel is secured (e.g., TLS IBB), the authorisation token introspected (the JWT), and an audit trail is written with role as “client”. Any Client eventually verifies the detached signature by accessing the Detached Signature service.

Both the Remote Site and the Any Client tracks provenance using the ex:CREATE or ex:MODIFIED accordingly.β

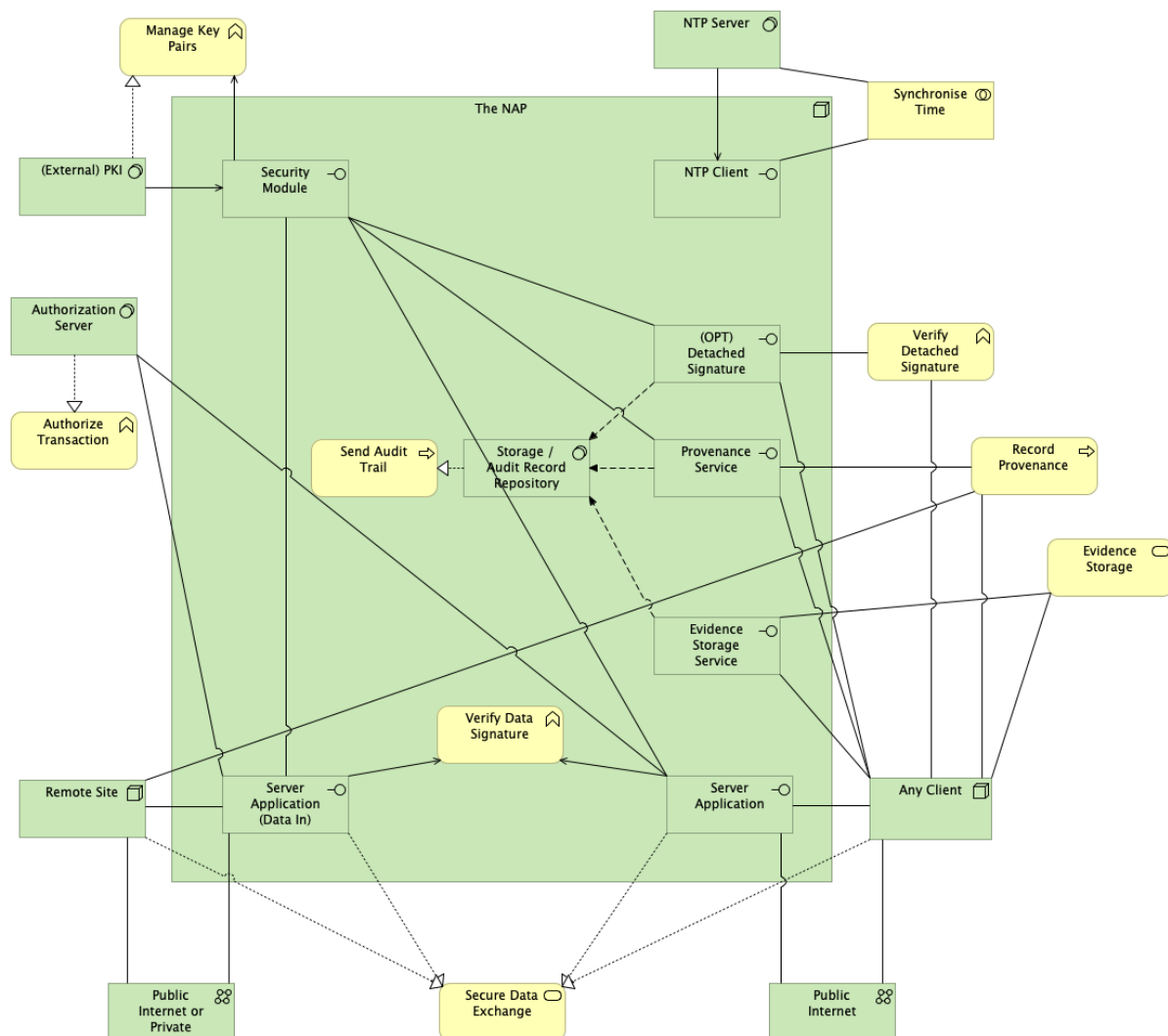


Figure 17 The NAP Security Reference Architecture

The Methodology

In this section it will be presented a methodology that could be used to derive RATs and ITTs. The methodology is designed upon a set of questions that should assist in creating a list that will match the Tag field of the **ABB**⁶⁰.

ABB	Tag Implemented
Node Authentication ABB	Client Identification
Node Authentication ABB	Supplier Identification
Data Integrity ABB	Integrity
Data Encryption ABB	Confidentiality In Transit
Node Authentication ABB, Authorization ABB	State the intended Recipient
Node Authentication ABB	Client Authentication
Node Authentication ABB	Supplier Authentication
Authorization ABB	Client Authorization
Non-Repudiation ABB	Non-Repudiation
Audit Trail ABB	Logging
Data Classification ABB	Data Classification
Data Encryption ABB	Confidentiality At Rest
Data Provenance ABB	Provenance
	Availability

ABB1
 ... **Tag:** client identification ...

ABB2
 ... **Tag:** integrity ...

Notably, additional tags may be added to the list of the building blocks, that may exceed the Datex II features.

The methodology acts systematically by defining a set of questions defined upon the concepts of ISO 27001:2013, namely Confidentiality, Integrity, and Availability (in this order). The CIA triad is evaluated upon an initial data classification. Additional requirements, features, and building block may arise upon a risk assessment, or stricter legal compliance.

The methodology is applied typically by the **system architect** interviewing the **domain experts**. The methodology applies to both PIM and PSM and it is as follows.

Table 14

1	The System Architect describe the use case in plain English, or uses one of the available DATEX II Recommended Profiles
2	Is there a RAT available? (e.g., did already someone solve the problem?)

⁶⁰ Highlighted cells are non-Datex II features

3		Yes: Use it. <i>EndOfMethodology</i> .
4		No: create the RAT by performing the following activities:
5		Analyse the high-level goals in terms of <i>data classification</i> :
6		Is the data public or should it be kept confidential and restricted to a group of users? (<i>Tag: Client Authorization</i>)
7		Does the context have already some policy or regulations in terms of data classification? (<i>Tag: Data Classification</i>)
8		Should the data in transit be observed passively? (<i>Tag: Confidentiality In Transit</i>)
9		Am I using a cloud solution? (<i>Tag: Confidentiality At Rest</i>)
10		Analyse the high-level goals in terms of integrity:
11		What could happen if the data is manipulated in its way? (<i>Tag: Integrity</i>)
12		Should I have logs available of who accessed the data? (<i>Tag: Logging</i>)
13		What is the impact if a message is not coming from/going to the intended actor? (<i>Tag: Non-Repudiation</i>)
14		Analyse the high-level goals in terms of availability:
15		what could happen if the system gets compromised and therefore it cannot deliver the service? (<i>Tag: Availability</i>)
16		Define the Exchange Platform Independent Model (PIM), the Exchange Pattern, and the functional exchange profile.
17		For each transaction in the UML Sequence Diagram and For Each Item in the UML Class Diagram:
18		Map the answer given to each item or transaction.
19		Select the ABBs implementing the TAG and the related ABBs. This is the <i>Reference Architecture Template</i> for the PIM.
20		Define the Exchange Platform Specific Model (PSM).
21		Select the IBBs that implements the Reference Architecture, and profile the optional requirements according to the <i>variability points</i> .
22		From the IBB set, discard those who are not related to the technology of the PSM. This new set is the <i>Interoperability Technological Template</i> (Solution Architecture). The IBBs contain the Security profiles of the standards pertaining the technology (e.g., JWT for RESTful authentication, WS-Security for SOAP-based communications)

Application of the methodology

The methodology has been tested with two use cases: the implementation of the German NAP (Mobilithek) and the Center 2 Center use case. In this section the two use cases will be presented.

The German NAP (Mobilithek)

Mobilithek aims to support the exchange of data between data providers and data users, with the help of interfaces and at the same time represents a central portal with the collected information on available online traffic data of individual data providers⁶¹. Mobilithek is made up of four components. This document will concentrate on the Security component, that "can be used to authenticate the data recipient system/data provider system or the data recipient/data provider in order to be able to use services".

Mobilithek supports both XML and JSON Datex II format through native interfaces

⁶¹ Federal Ministry for Digital Transport, *Mobilithek, Technical interface description, V 1.1 - 10.11.2022*



The German NAP (Mobilithek) obviously did not yet have a RAT (2), so the questions asked are from 5, 10, and 14. In particular,

- Answer 8: the data shall enjoy confidentiality while in transit, using standard technology such as TLS 1.2. PKI is provided by the operator. *Tag: Confidentiality In Transit*
- Answer 12: Mobilithek performs security logging of the following events (*Tag: Logging*):
 - Incorrect login attempts
 - System access from user accounts with administrative rights
 - Administration of accounts
 - Change of group memberships of accounts
 - Critical increase in system values such as memory load, CPU utilisation over a longer period of time

Clearly, Mobilithek does not have a PIM nor a PSM, thus no additional steps are performed. The RAT is therefore composed by the following ABB exposing the two tags: “Confidentiality In Transit” and “Logging”. Those two tags belong to the

- Node Authentication ABB
- Audit Trail ABB

Mobilithek has additional features such as use of JWT tokens for authorizing access to the user interface (which is out of scope of the RAT).

Evaluation of NODEABB. the requirements NODEABB-1-T and NODEABB-2-T are in scope, thus the German NAP should provide evidence that the administrator’s access is trusted (e.g., by showing that the Administrator log is performed through the use of a Secure Remote Access Solution). Moreover, the German NAP should make available documentation on two policies: *i) certificate provisioning and PKI contracts*, and *ii) Secure Data Exchange policy*, such as ciphersuite used, certificate key usages and certificate policies. Moreover, for ISO 27001 the following requirements should be implemented by providing documented evidence.

- A.8.2.3, Handling of Assets. *Procedures for handling assets shall be developed and implemented in accordance with the information classification scheme adopted by the organization.* This documentation has to be provided, eventually involving Data Classification ABB.
- A.13.1.1, Network controls. *Networks shall be managed and controlled to protect information in systems and applications.* This documentation has to be prepared, taking into consideration that the channel authentication and confidentiality is protected by this ABB.
- A.13.2.1, Information transfer policies and procedures. *Formal transfer policies, procedures and controls shall be in place to protect the transfer of information through the use of all types of communication facilities.* This building block defines the policies through its implementation.
- A.13.2.3, Electronic Messaging. *Information involved in electronic messaging shall be appropriately protected.* This is provided by the implementation of this ABB.
- A.14.1.2, Securing application services on public networks. *Information involved in application services passing over public networks shall be protected from fraudulent activity, contract dispute and unauthorized disclosure and modification.* This control is provided by this ABB.
- A.14.1.3, Protecting application services transactions. *Information involved in application service transactions shall be protected to prevent incomplete transmission, misrouting, unauthorized message alteration, unauthorized disclosure, unauthorized message duplication or replay.* This is provided by this ABB.

The implementation is performed through **TLS IBB**. Mobilithetk do not use certificate pinning with public clients, and it does not use certificate transparency, but implements TLSIBB-2-T. Not using certificate pinning is compliant with the TLS IBB via the *TLS non mutually authenticated, and no certificate pin made by the client* option. As the PKI is public, a CRL and a documented Registration Authority interaction should be made available.

In case with private client, TLS is performed via certificate authentication, thus using certificate pinning. Ciphersuites are in both cases compliant with the one provisioned by the IBB.

Implementation should provide evidence as shown in TLSIBB-TA-2 and TLSIBB-TA-3, although the use of HTTPS-only interfaces is described in the Mobilithetk specification document.

TN-ITS use case

TN-ITS is an end-user driven standardized data sharing platform, with a focus on the exchange of map attribute updates between road authorities and digital map service providers. The TN-ITS framework includes a standardized data format and protocols for data exchange, ensuring compatibility and interoperability between different systems and stakeholders. The main objective of TN-ITS is to ensure that public road authorities, the data hubs, create and make authoritative data available, with the highest quality possible, to meet the needs of the end-user. Within these regards, the establishment of trust among the various stakeholders is crucial. TN-ITS stakeholders are defined throughout its data chain, and they are as follows.

- Data Providers and Data Owners, responsible for generating and supplying accurate and reliable data related to traffic and navigation
- Data Access Enablers (e.g., the NAP), responsible to facilitate the exchange and distribution of data within the TN-ITS data chain
- Data Consumers, stakeholders that utilize the data available in the TN-ITS data chain to provide services and applications for the end users⁶².

The TN-ITS data chain is composed by five phases, as follows.

- Data Collection, where road authorities collect data related to changes on their road infrastructure
- Data Processing, where the collected data are processed and converted into standardized format
- Data Exchange, where the processed data are exchanged through an XML format
- Data Integration, where map providers process the data and integrate them into their digital map,
- Data Usage, where the updated digital maps are used in various applications

Establishing trust in the TN-ITS ecosystem is a multidimensional task. In fact, trust is composed of various aspects such as brand recognition, service quality, compliance, standard adherence and ICT aspects. In this section we will consider this latter concept, related to ICT. In particular, we consider those aspects as:

- Cybersecurity
- Authorisation
- Authentication

⁶² Analysis made upon *TN-ITS Inventory on requirements related to trust, quality, integrity, security and sovereignty of data, contribution from Sub-Working Group 4.2*, 29 August 2023

- API Security (and clock synchronization)
- Integrity
- Sovereignty

Other aspects of trust, such as brand recognition, have also ICT aspects that are out of scope of this document. For instance, company security policies or Cyber Threat Intelligence can increase the Brand Reputation by observing the public exposure of vulnerable services.

TN-ITS obviously did not yet have a RAT (2), so the questions asked are from 5, 10, and 14. In particular:

- Point 5: Data needs authorization, and thus should be protected in transit, and delivered through specified Access Enablers exposing secure APIs (e.g., a NAP) *Tag: Client Authorization, Supplier Identification, Confidentiality in Transit*
- Point 10: Data should enjoy integrity and should be secured throughout the entire chain. *Tag: Integrity, Provenance, Availability*

Clearly, TN-ITS does not have a PIM nor a PSM, thus no additional steps are performed. The RAT is therefore composed by the following ABB exposing the tags defined above.

- Authorization ABB
- Node Authentication ABB
- Data Encryption ABB
- Non-Repudiation ABB
- Data Integrity ABB
- Data Provenance ABB

ANNEX A – Lab System Setup

In this section it will be shown the setup used for the various evidence gathering and reasonings made on the building blocks.

The kill chain – assumptions

The evidence gathering examples make the following assumptions.

The attacker is omnipotent, e.g., it is a valid user in the network that can do anything, but not perform brute force attacks on the encrypted data. This assumption leverages the typical attacker known as Dolev-Yao⁶³;

The reason behind this decision is due to the scope of this deliverable, which focuses on the data exchange rather than the security of a company. In fact, being a valid user in the network means that the already performed steps on the *kill-chain*, e.g., the phases performed in a modern cyberattack, or (according to MITRE) the tactics performed. This document assumes that the attacker:

⁶³ Dolev, D.; Yao, A. C. (1983), "On the security of public key protocols", IEEE Transactions on Information Theory, IT-29 (2): 198–208, doi:10.1109/tit.1983.1056650, S2CID 13643880

- Has already an account in the corporate/OT networks (e.g., stolen from leak sites, or by social engineering, or other techniques)
- Is capable to steal a key pair (e.g., by purchasing a valid certificate from a known CA, or compromising the local CA⁶⁴)
- Has a knowledge of the domain.

The lab system setup

The evidence gathering scenarios are usually composed by three actors:

- Alice, the sender,
- Bob, the receiver,
- Mallory, the attacker.

Every user (including Mallory) needs to own a key pair. The following pre-requisites are setup:

- A Certification authority, named demoCA, created using openssl
- 3 certificates: a client, a server, and one for the attacker, Mallory
- The client is a Java Application which opens a socket and sends raw data
- The server is an Apache Tomcat

Creating the CA

Firstly we need to create the file structure:

```
> mkdir demoCA
> cd demoCA
> mkdir newcerts
> touch index.txt
> echo 01 > serial
```

Creating the CA private key

```
> openssl genrsa -out cakey.pem 4096
> mkdir -p demoCA/private
> mv cakey.pem demoCA/private
```

Create CA public key (note the certificate will last 1000 days)⁶⁵

```
>
openssl req -new -x509 -key demoCA/private/cakey.pem -out demoCA/cacert.pem
-days 1000
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

⁶⁴ This is an attack initially appeared in *ENISA, the operation Black Tulip: How the Certificate Authority Lose Authority*, available at <https://www.enisa.europa.eu/media/news-items/operation-black-tulip> (Last Access July 26, 2023) and in Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra. 2008. *Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps*. In Proceedings of the 6th ACM workshop on Formal methods in security engineering (FMSE '08). Association for Computing Machinery, New York, NY, USA, 1–10. <https://doi.org/10.1145/1456396.1456397>

⁶⁵ Note: to avoid adding 1000 days always, change the default_days entry in openssl.cnf

```
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Italy
Locality Name (eg, city) []:Florence
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Autostrade Per
L'Italia
Organizational Unit Name (eg, section) []:Direzione IT
Common Name (e.g. server FQDN or YOUR name) []:NAPCORE DEMO CA
Email Address []:mmasasi@autostrade.it
```

See the certificate created and its attributes

```
> openssl x509 -in cacert.pem -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            31:29:fa:d6:ad:f6:6b:83:c1:7a:df:cc:f8:60:4d:6d:ff:9d:d1:88
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = IT, ST = Italy, L = Florence, O = Autostrade Per
L'Italia, OU = Direzione IT, CN = NAPCORE DEMO CA, emailAddress =
mmasasi@autostrade.it
        Validity
            Not Before: Jul 27 08:18:17 2023 GMT
            Not After : Apr 22 08:18:17 2026 GMT
        Subject: C = IT, ST = Italy, L = Florence, O = Autostrade Per
L'Italia, OU = Direzione IT, CN = NAPCORE DEMO CA, emailAddress =
mmasasi@autostrade.it
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                00:9f:e5:59:b0:4b:e0:0d:82:74:18:32:56:d7:b1:
                f1:94:bb:b4:64:0f:ef:da:3a:74:4f:36:9f:5b:a6:
                47:69:7d:2f:81:17:e4:10:be:ca:0c:3b:4e:93:a6:
                75:e6:1c:12:df:b9:d5:7e:ec:0c:ea:72:ff:6c:d2:
                a0:c4:96:92:d6:5e:7c:13:4f:45:26:7d:5b:78:5e:
                3d:fb:b1:af:b6:20:35:b2:ed:0d:48:4e:8a:87:26:
                cb:7b:ae:ec:40:0b:2c:e8:63:35:8a:50:29:97:7c:
                3c:ba:d3:27:c0:39:e7:b7:db:42:fd:05:80:5d:69:
                75:a3:66:aa:85:ed:7b:3e:d5:18:ac:72:b6:9f:cd:
                b5:08:5d:82:a5:1f:0e:de:f2:d9:c7:e9:70:c0:cc:
                39:76:79:dd:12:f7:bf:c4:73:00:de:dc:99:67:1e:
                78:00:0e:14:25:c2:38:24:5c:84:93:53:f4:fc:01:
                dc:89:b4:5b:84:a9:e3:d4:b5:ee:1c:53:da:1b:92:
                4c:4d:cc:5c:3a:bb:a9:12:04:aa:3e:5f:ff:a6:a0:
                9a:f0:d8:4c:2e:88:38:5c:e3:6a:da:dc:75:6c:75:
                18:74:7f:4d:d5:ec:2d:3a:3a:96:62:7a:75:f9:4f:
                e4:f2:20:97:d9:0f:e7:ed:e0:f9:63:16:21:80:18:
                93:89
            Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                BC:16:E1:90:D3:D6:AC:4F:DE:19:8E:3B:BE:F0:D2:B0:E1:C7:96:9F
            X509v3 Authority Key Identifier:
                BC:16:E1:90:D3:D6:AC:4F:DE:19:8E:3B:BE:F0:D2:B0:E1:C7:96:9F
            X509v3 Basic Constraints: critical
                CA:TRUE
```



```
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
09:f6:67:9e:c3:b3:18:3c:e6:fd:cd:89:d0:4c:06:a3:3b:34:
bc:95:e3:1c:81:80:ad:72:5e:a7:bb:c8:33:8b:35:f8:7d:66:
b8:cb:b0:c0:de:3f:25:65:3c:30:ad:ce:6c:78:8b:4a:67:cd:
23:f8:a1:d2:46:11:7e:2d:42:70:cc:45:fe:17:69:57:b8:02:
ed:20:2a:2e:73:00:44:09:ee:85:cd:5c:1e:15:ef:df:3c:c0:
99:f8:ed:0c:06:ee:82:d5:dd:c6:8f:23:c8:55:0f:a1:40:54:
aa:e9:0e:5d:51:f5:62:f2:bf:0a:3a:95:23:55:57:87:7c:5e:
ba:66:1b:60:d7:95:d8:2c:62:74:60:c2:11:89:3e:44:1c:1b:
85:48:36:e8:b9:5d:2e:6d:57:08:a0:ff:a5:2d:24:a9:94:9b:
72:56:5c:8e:da:fd:ec:24:7d:81:6f:ea:09:23:22:76:05:d2:
d7:4f:f1:4a:98:ed:bc:94:91:df:6a:87:53:9a:87:eb:9e:1e:
3c:b7:3b:84:10:2b:b7:3b:10:52:5a:a5:e2:2c:d6:0f:eb:e4:
98:7b:6a:39:58:e9:9c:92:f3:0c:86:e6:95:98:ab:f8:a0:bc:
d3:76:5e:a1:13:38:4c:22:23:36:c7:c6:59:b5:12:41:4b:4d:
e2:9e:8a:4c
```

Create Client Certificate

Notably, in order to perform TLS authentication, we changed the openssl.cnf to add two extended key usage attributes, 1.3.6.1.5.5.7.3.1 (server authentication) and 1.3.6.1.5.5.7.3.2 (client authentication).

```
> openssl genrsa -out clientkey.pem 2048
> openssl req -new -key clientkey.pem -out clientreq.pem -config
demoCA/openssl.cnf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Italy
Locality Name (eg, city) []:Florence
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Autostrade Per
L'Italia
Organizational Unit Name (eg, section) []:Direzione IT
Common Name (e.g. server FQDN or YOUR name) []:client
Email Address []:mmasi@autostrade.it

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

> openssl ca -in demoCA/clientreq.pem -out demoCA/clientcert.pem -days 1000
-config demoCA/openssl.cnf
Using configuration from demoCA/openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
```

```

Serial Number: 1 (0x1)
Validity
  Not Before: Jul 27 08:42:08 2023 GMT
  Not After : Apr 22 08:42:08 2026 GMT
Subject:
  countryName           = IT
  stateOrProvinceName   = Italy
  organizationName      = Autostrade Per L'Italia
  organizationalUnitName = Direzione IT
  commonName            = client
  emailAddress          = mmasi@autostrade.it
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  X509v3 Key Usage:
    Digital Signature, Non Repudiation, Key Encipherment
  X509v3 Subject Key Identifier:
    2F:7F:8E:FC:3C:08:0C:0D:86:4B:27:6E:16:B2:92:A9:9E:FD:93:FB
  X509v3 Authority Key Identifier:
    BC:16:E1:90:D3:D6:AC:4F:DE:19:8E:3B:BE:F0:D2:B0:E1:C7:96:9F
  X509v3 Extended Key Usage:
    TLS Web Server Authentication, TLS Web Client
Authentication
Certificate is to be certified until Apr 22 08:42:08 2026 GMT (1000 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

```

Generate Server Certificate

```

> openssl req -new -key serverkey.pem -out serverreq.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Italy
Locality Name (eg, city) []:Florence
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Autostrade Per
L'Italia
Organizational Unit Name (eg, section) []:Direzione IT
Common Name (e.g. server FQDN or YOUR name) []:server
Email Address [[:mmasi@autostrade.it

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

> openssl ca -in demoCA/serverreq.pem -out demoCA/servercert.pem -days 1000
-config demoCA/openssl.cnf

```



```

Using configuration from demoCA/openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 2 (0x2)
    Validity
        Not Before: Jul 27 08:43:56 2023 GMT
        Not After : Apr 22 08:43:56 2026 GMT
    Subject:
        countryName           = IT
        stateOrProvinceName   = Italy
        organizationName      = Autostrade Per L'Italia
        organizationalUnitName = Direzione IT
        commonName            = server
        emailAddress          = mmasi@autostrade.it
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        X509v3 Key Usage:
            Digital Signature, Non Repudiation, Key Encipherment
        X509v3 Subject Key Identifier:
            62:5D:1F:77:65:18:11:69:97:46:E7:6C:AC:48:DE:0B:4C:9B:79:33
        X509v3 Authority Key Identifier:
            BC:16:E1:90:D3:D6:AC:4F:DE:19:8E:3B:BE:F0:D2:B0:E1:C7:96:9F
        X509v3 Extended Key Usage:
            TLS Web Server Authentication, TLS Web Client
Authentication
Certificate is to be certified until Apr 22 08:43:56 2026 GMT (1000 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

> ls
clientcert.pem  clientreq.pem  servercert.pem  serverreq.pem
clientkey.pem   demoCA         serverkey.pem

```

Generate Mallory Certificate

```

> openssl genrsa -out mallorykey.pem 2048

> openssl req -new -key mallorykey.pem -out malloryreq.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Italy
Locality Name (eg, city) []:Florence
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Autostrade Per
L'Italia

```



```
Organizational Unit Name (eg, section) []:Direzione IT
Common Name (e.g. server FQDN or YOUR name) []:mallory
Email Address []:mmasi@autostrade.it
```

Please enter the following 'extra' attributes
to be sent with your certificate request

A challenge password []:

An optional company name []:

```
> openssl ca -in demoCA/malloryreq.pem -out demoCA/mallorycert.pem -days
1000 -config demoCA/openssl.cnf
```

Using configuration from demoCA/openssl.cnf

Check that the request matches the signature

Signature ok

Certificate Details:

Serial Number: 3 (0x3)

Validity

Not Before: Jul 27 08:45:26 2023 GMT

Not After : Apr 22 08:45:26 2026 GMT

Subject:

countryName = IT

stateOrProvinceName = Italy

organizationName = Autostrade Per L'Italia

organizationalUnitName = Direzione IT

commonName = mallory

emailAddress = mmasi@autostrade.it

X509v3 extensions:

X509v3 Basic Constraints:

CA:FALSE

X509v3 Key Usage:

Digital Signature, Non Repudiation, Key Encipherment

X509v3 Subject Key Identifier:

DD:98:76:F7:11:1A:B5:FB:C2:2A:B3:71:49:66:0F:D7:C8:FD:06:D2

X509v3 Authority Key Identifier:

BC:16:E1:90:D3:D6:AC:4F:DE:19:8E:3B:BE:F0:D2:B0:E1:C7:96:9F

X509v3 Extended Key Usage:

TLS Web Server Authentication, TLS Web Client

Authentication

Certificate is to be certified until Apr 22 08:45:26 2026 GMT (1000 days)

Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y

Write out database with 1 new entries

Data Base Updated

In order to test OCSP validations, a new certificate is created with the Authority Information Access (AIA) pointing to an OCSP server stored in the local server. To do so, we added the following lines to the openssl.cnf

```
authorityInfoAccess = OCSP;URI:http://192.168.1.2:32800
```

Under the `usr_cert` section. Note that the uri is set to the host which will have an OCSP responder. In this test we will use OpenSSL to perform this task.

```
> openssl ocsp -index demoCA/index.txt -port 32801 -rsigner
demoCA/ocspServerSignature.pem -rkey demoCA/ocspServerSignature.key -CA
demoCA/cacert.pem -text -out demoCA/OCSPLLOG.txt
```

Generate Java Keystore

```
> openssl pkcs8 -topk8 -in clientkey.pem -out clientKey.der -outform DER -
nocrypt
> openssl pkcs8 -topk8 -in serverkey.pem -out serverKey.der -outform DER -
nocrypt
> openssl pkcs8 -topk8 -in mallorykey.pem -out malloryKey.der -outform DER
-nocrypt
> openssl x509 -in clientcert.pem -out clientCert.der -outform DER
> openssl x509 -in servercert.pem -outform der -out servercert.der
> openssl x509 -in mallorycert.pem -outform der -out mallorycert.der
```

```
openssl pkcs12 -export -in clientcert.pem -inkey clientkey.pem -out
clientKeystore.p12 -name "client"
Enter Export Password:
Verifying - Enter Export Password:
> openssl pkcs12 -export -in servercert.pem -inkey serverkey.pem -out
serverKeystore.p12 -name "server"
Enter Export Password:
Verifying - Enter Export Password:
> openssl pkcs12 -export -in mallorycert.pem -inkey mallorykey.pem -out
malloryKeystore.p12 -name "mallory"
Enter Export Password:
Verifying - Enter Export Password:
```

```
> keytool -importkeystore -srckeystore clientKeystore.p12 -srcstoretype
pkcs12 -destkeystore clientKeystore.jks
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -
Dswing.aatext=true
Importazione del keystore clientKeystore.p12 in clientKeystore.jks in
corso...
Immettere la password del keystore di destinazione:
Immettere nuovamente la nuova password:
Immettere la password del keystore di origine:
La voce dell'alias client è stata importata.
Comando di importazione completato: 1 voce/i importata/e, 0 voce/i non
importata/e o annullata/e
```

```
> keytool -importkeystore -srckeystore serverKeystore.p12 -srcstoretype
pkcs12 -destkeystore serverKeystore.jks
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -
Dswing.aatext=true
Importazione del keystore serverKeystore.p12 in serverKeystore.jks in
corso...
Immettere la password del keystore di destinazione:
Immettere nuovamente la nuova password:
Immettere la password del keystore di origine:
La voce dell'alias server è stata importata.
```

```

Comando di importazione completato: 1 voce/i importata/e, 0 voce/i non
importata/e o annullata/e

> keytool -importkeystore -srckeystore malloryKeystore.p12 -srcstoretype
pkcs12 -destkeystore malloryKeystore.jks
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -
Dswing.aatext=true
Importazione del keystore malloryKeystore.p12 in malloryKeystore.jks in
corso...
Immettere la password del keystore di destinazione:
Immettere nuovamente la nuova password:
Immettere la password del keystore di origine:
La voce dell'alias mallory è stata importata.
Comando di importazione completato: 1 voce/i importata/e, 0 voce/i non
importata/e o annullata/e

keytool -keystore truststore.jks -import -file cacert.der -alias ca
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -
Dswing.aatext=true
Immettere la password del keystore:
Immettere nuovamente la nuova password:
Proprietario: EMAILADDRESS=mmasi@autostrade.it, CN=NAPCORE DEMO CA,
OU=Direzione IT, O=Autostrade Per L'Italia, L=Florence, ST=Italy, C=IT
Emittente: EMAILADDRESS=mmasi@autostrade.it, CN=NAPCORE DEMO CA,
OU=Direzione IT, O=Autostrade Per L'Italia, L=Florence, ST=Italy, C=IT
Numero di serie: 3129fad6adf66b83c17adfccf8604d6dff9dd188
Valido da: Thu Jul 27 10:18:17 CEST 2023 a: Wed Apr 22 10:18:17 CEST 2026
Impronte digitali certificato:
  SHA1: CB:44:9C:42:53:D7:10:3A:02:14:81:0D:57:31:5A:3E:F8:C1:ED:15
  SHA256:
B6:D3:E3:29:7E:BA:00:27:7A:84:14:35:72:AB:67:97:B7:8D:D8:26:09:89:92:44:62:
35:3F:CA:32:96:C7:E3
Nome algoritmo firma: SHA256withRSA
Algoritmo di chiave pubblica oggetto: Chiave RSA a 2048 bit
Versione: 3

Estensioni:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: BC 16 E1 90 D3 D6 AC 4F  DE 19 8E 3B BE F0 D2 B0  ....O...;....
0010: E1 C7 96 9F                                     ....
]
]

#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen: no limit
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: BC 16 E1 90 D3 D6 AC 4F  DE 19 8E 3B BE F0 D2 B0  ....O...;....
0010: E1 C7 96 9F                                     ....
]
]

```

```
]

Considerare sicuro questo certificato? [no]:  si
Il certificato è stato aggiunto al keystore

> keytool -keystore truststore.jks -import -file servercert.der -alias
server
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -
Dswing.aatext=true
Immettere la password del keystore:
Il certificato è stato aggiunto al keystore

> java -version
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -
Dswing.aatext=true
openjdk version "17.0.6" 2023-01-17
OpenJDK Runtime Environment (build 17.0.6+10-Debian-1)
OpenJDK 64-Bit Server VM (build 17.0.6+10-Debian-1, mixed mode, sharing)
```

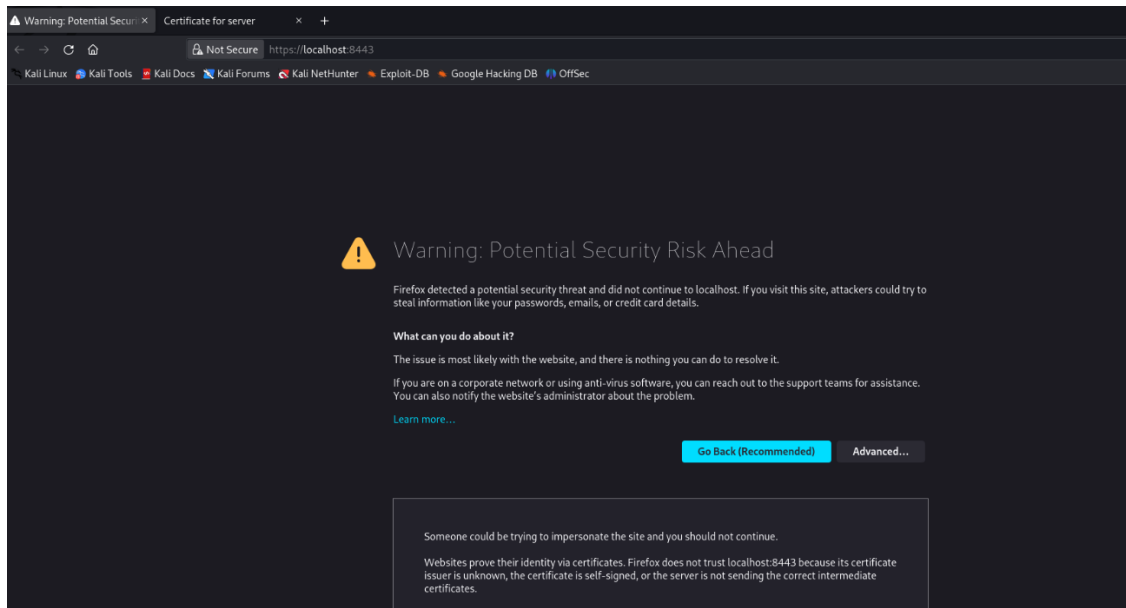
Setup Tomcat

Setup connector in tomcat in conf/server.xml

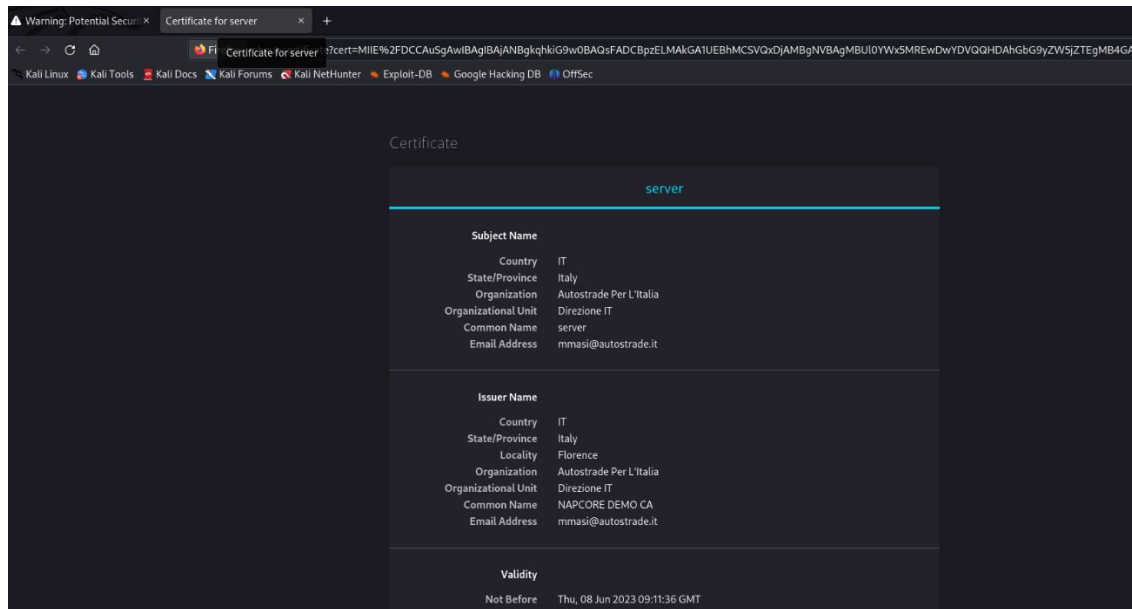
```
<Connector
protocol="org.apache.coyote.http11.Http11NioProtocol"
  port="8443" maxThreads="200"
  maxParameterCount="1000"
  scheme="https" secure="true" SSLEnabled="true"
  SSLCertificateFile="/opt/tomcat/serverCert.pem"
  SSLCertificateKeyFile="/opt/tomcat/serverKey.pem"
  SSLCertificateChainFile="/opt/tomcat/cacert.pem"
  SSLVerifyClient="require" SSLProtocol="TLSv1+TLSv1.1+TLSv1.2"/>
```

execute `catalina.sh run`.

When connecting to the url (usually <https://localhost:8443>) via a web browser, you should see the warning of the certificate as in the figure below.



When checking for the certificate details, you should see the created certificate as in the picture below.



Generate truststore

```
> openssl x509 -in cacert.pem -outform der -out cacert.der

> keytool -keystore ../truststore.jks -importcert -alias ca -file
cacert.der
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -
Dswing.aatext=true
Immettere la password del keystore:
Immettere nuovamente la nuova password:
Proprietario: EMAILADDRESS=mmasasi@autostrade.it, CN=NAPCORE DEMO CA,
OU=Direzione IT, O=Autostrade Per L'Italia, L=Florence, ST=Italy, C=IT
```



```

Emittente: EMAILADDRESS=mmas@autostrade.it, CN=NAPCORE DEMO CA,
OU=Direzione IT, O=Autostrade Per L'Italia, L=Florence, ST=Italy, C=IT
Numero di serie: 55bd6bbf0bf21c30f21cfa83bf6af5f8c5f30204
Valido da: Thu Jun 08 10:48:21 CEST 2023 a: Sat Jul 08 10:48:21 CEST 2023
Impronte digitali certificato:
    SHA1: 6B:61:D4:95:2F:0C:C9:35:71:5D:6C:BD:A3:83:2E:64:82:31:29:F9
    SHA256:
F3:72:74:99:BE:51:FE:72:30:72:6E:4C:DA:E4:56:8B:6D:C9:CD:17:3F:DD:FA:8C:55:
D6:9B:97:62:2D:8F:06
Nome algoritmo firma: SHA256withRSA
Algoritmo di chiave pubblica oggetto: Chiave RSA a 4096 bit
Versione: 3

```

Estensioni:

```

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 55 F8 43 C5 56 0D 90 C6   FE 82 00 C6 05 F6 DA A2   U.C.V.....
0010: 16 2B 4B AF                               .+K.
]
]

```

```

#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
    CA:true
    PathLen: no limit
]

```

```

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 55 F8 43 C5 56 0D 90 C6   FE 82 00 C6 05 F6 DA A2   U.C.V.....
0010: 16 2B 4B AF                               .+K.
]
]

```

```

Considerare sicuro questo certificato? [no]: si
Il certificato è stato aggiunto al keystore

```

Similarly, to the client certificate, a new certificate is created, named `revokedcert`, which is used to test the certificate revocation. In fact, once the certificate is created, we remove it as follows.

```

openssl ca -revoke demoCA/revokedcert.pem -config demoCA/openssl.cnf
Using configuration from demoCA/openssl.cnf
Revoking Certificate 04.
Data Base Updated

```

We witness it as follows



```
openssl ca -status 4
Using configuration from /usr/lib/ssl/openssl.cnf
04=Revoked (R)
```

We generate the Certificate Revocation List (CRL) file to distribute.

```
> touch demoCA/crlnumber
> echo 01 > demoCA/crlnumber
> openssl ca -gencrl -out demoCA/crl.pem
> openssl crl -in demoCA/crl.pem -text
Certificate Revocation List (CRL):
    Version 2 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = IT, ST = Italy, L = Florence, O = Autostrade Per
L'Italia, OU = Direzione IT, CN = NAPCORE DEMO CA, emailAddress =
mmasi@autostrade.it
    Last Update: Aug  2 14:13:21 2023 GMT
    Next Update: Sep  1 14:13:21 2023 GMT
    CRL extensions:
        X509v3 CRL Number:
            2
RevokedCertificates:
    Serial Number: 04
    Revocation Date: Aug  2 14:07:45 2023 GMT
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
    37:c0:ca:07:84:1b:47:05:cb:4f:ed:46:ff:63:87:6c:15:9d:
    4a:ab:f9:2e:c7:29:6a:ed:40:48:f6:a1:2f:89:0c:f5:35:b9:
    b8:3d:8f:62:df:bb:92:8a:a5:ae:9c:e5:16:b9:77:d5:3d:2f:
    69:c8:d9:46:dc:7a:d8:65:f2:38:57:63:76:ee:dc:44:9e:78:
    08:39:c1:60:25:28:f4:01:86:f4:c0:0e:9c:df:c9:69:58:20:
    a5:ba:04:c4:93:21:97:66:4e:d8:cc:4b:30:5d:76:e1:33:7b:
    69:ef:82:94:63:02:11:c3:c5:7e:7c:f2:cb:44:5b:7f:0b:2d:
    d5:cd:e9:80:d3:20:e3:d2:34:6d:db:2c:c2:9f:26:f5:23:55:
    01:69:55:6e:50:09:9a:cf:5c:22:3c:55:21:e4:ed:af:ca:90:
    95:8c:55:b2:2f:5c:e6:04:52:50:f1:a3:f0:43:79:e8:2d:5c:
    92:9e:f0:2e:7f:c3:b2:23:f9:32:3b:df:7d:df:06:ec:c1:75:
    c8:64:36:67:23:20:4b:71:3a:af:d9:f7:2b:57:d1:a3:af:a2:
    66:fa:6d:a9:51:f6:81:e4:94:16:f5:4c:71:59:f9:17:58:0e:
    51:65:82:c1:0e:cb:c3:5e:35:b3:01:1c:f0:59:de:53:9f:27:
    0e:4a:71:29
-----BEGIN X509 CRL-----
MIICGDCCAQAQAEwDQYJKoZIhvcNAQELBQAwgacxCzAJBgNVBAYTAklUMQ4wDAYD
VQQIDAVJdGFseTERMA8GA1UEBwwIRmxcvcmVuY2UxIDAeBgNVBAoMF0F1dG9zdHJh
ZGUgUGVyeIEwnSXRhbGhlMRUwEwYDQQLDAxEaXJlemlvbmUgSVQxGDAwBgNVBAMM
D05BUENPUkUgREVNTyBDQTEiMCAGCSqGSIb3DQEJARYTBW1hc2lAYXV0b3N0cmFk
ZS5pdBcNMjMwODAyMTQxMzIxWhcNMjMwOTAxMTQxMzIxWjAUMBICAQQXDTIzMDgw
MjE0MDc0NVqgDjAMMAoGA1UdFAQDAgECMA0GCSqGSIb3DQEBCwUAA4IBAQA3wMoH
hBtHBctP7Ub/Y4dsFZ1Kq/kuxylq7UBI9qEviQz1Nbm4PY9i37uSiqWunOUWuXfV
PS9pyNlG3HrYZfI4V2N27txEnngIOcFgJSj0AYb0wA6c38lpWCClugTEkyGXZk7Y
zEswXXbhM3tp74KUYwIRw8V+fPLLRft/Cy3VzemA0yDj0jRt2yzCnyb1l1UBaVVu
UAmaz1wiPFUh502vypCVjFWyL1zmBFJQ8aPwQ3noLVySnnAuf8OyI/kyO9993wbs
wXXIZDZnIyBLcTqv2fcrV9Gjr6Jm+m2pUfaB5JQW9UxxWfkXWA5RZYLBDsvDXjWz
ARzwWd5TnycOSnEp
-----END X509 CRL-----
```

