



The DELTA project has received funding from the EU's Horizon 2020 research and innovation programme under grant agreement No 773960



# DELTA

Project Acronym: **DELTA**

Project Full Title: **Future tamper-proof Demand rEsponse framework through seLf-configured, self-opTimized and collAborative virtual distributed energy nodes**

Grant Agreement: **773960**

Project Duration: **36 months (01/05/2018 – 30/04/2021)**

## DELIVERABLE D5.1

### Secure Information Exchange by Design in Energy Markets

Work Package **WP5 – Secure Data Handling and Exchange in future DR ecosystem**

Task **T5.1 – Secure Information Exchange by Design in Energy Markets**

Document Status: **Final**

File Name: **DELTA\_D5.1\_v1.0**

Due Date: **31.10.2019**

Submission Date: **05.11.2019**

Lead Beneficiary: **NTNU**

#### Dissemination Level

Public

X

Confidential, only for members of the Consortium (including the Commission Services)

## Authors List

Leading Authors				
First Name		Last Name	Beneficiary	Contact e-mail
Alessio		Baiocco	NTNU	<a href="mailto:alessio.baiocco@ntnu.no">alessio.baiocco@ntnu.no</a>
Georgios		Spathoulas	NTNU	<a href="mailto:georgios.spathoulas@ntnu.no">georgios.spathoulas@ntnu.no</a>
Co-Author(s)				
#	First Name	Last Name	Beneficiary	Contact e-mail
1	Juan	Cano-Benito	UPM	<a href="mailto:Jcano@fi.upm.es">Jcano@fi.upm.es</a>
2	Andrea	Cimmino	UPM	<a href="mailto:cimmino@fi.upm.es">cimmino@fi.upm.es</a>
3	Nikoleta	Andreadou	JRC	<a href="mailto:Nikoleta.ANDREADOU@ec.europa.eu">Nikoleta.ANDREADOU@ec.europa.eu</a>
4	Ioannis	Poursanidis	JRC	<a href="mailto:Ioannis.POURSANIDIS@ext.ec.europa.eu">Ioannis.POURSANIDIS@ext.ec.europa.eu</a>
5	Christos	Patsonakis	CERTH	<a href="mailto:cpatsonakis@iti.gr">cpatsonakis@iti.gr</a>
6	Sofia	Terzi	CERTH	<a href="mailto:sterzi@iti.gr">sterzi@iti.gr</a>

## Reviewers List

Reviewers			
First Name	Last Name	Beneficiary	Contact e-mail
Dimosthenis	Ioannidis	CERTH	<a href="mailto:djoannid@iti.gr">djoannid@iti.gr</a>
Ioannis	Moschos	CERTH	<a href="mailto:imoschos@iti.gr">imoschos@iti.gr</a>
George	Karagiannopoulos	HIT	<a href="mailto:g.karagiannopoulos@hit-innovations.com">g.karagiannopoulos@hit-innovations.com</a>

## Legal Disclaimer

The DELTA has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 773960. The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the Innovation and Networks Executive Agency (INEA) or the European Commission (EC). INEA or the EC are not responsible for any use that may be made of the information contained therein.

## Copyright

© NTNU. Copies of this publication – also of extracts thereof – may only be made with reference to the publisher.

## Executive Summary

This deliverable presents the results of “T5.1 - Secure Information Exchange by Design in Energy Markets”, which entails the definition, design and development of an end-to-end secure, privacy-preserving information exchange framework among all the entities that participate in DELTA’s ecosystem, as well as, all the layers comprising DELTA’s architecture.

The content provided in this document is structured in a way that illustrates the involved steps in our design methodology. The main building block upon which we design and develop DELTA’s secure information exchange framework are standard-based schemes for managing digital identities. We survey the most widely deployed standards on this topic that are commercially used by enterprises globally and present the construction of DELTA’s identity services. We elaborate on issues pertaining to the representation of the identities of all the actors that are involved in DELTA’s ecosystem. Furthermore, we present procedures that handle the lifecycle of those identities in the context of DELTA. This identity layer provides the necessary means to design and implement access control policies.

To regulate access to the data that is maintained throughout the DELTA platform, we present the design of DELTA’s access control layer. We illustrate our methodology in designing this layer by reviewing the most widely deployed schemes that provide this functionality, illustrating their appropriateness in terms of deployment and, more specifically, regarding their applicability in DELTA’s context.

As this is DELTA’s main security-related deliverable, we also tackle with the issue of privacy. More specifically, we provide a definition of the term privacy that is tailored to the DELTA’s ecosystem and illustrate how we employ the previously introduced layers to achieve it, while taking into consideration the outcomes of previous work packages.

DELTA’s inherent distributed architecture introduces additional challenges that, from a security point of view, need to be addressed. First, it is imperative to deal with the asynchrony of the underlying networking infrastructure, i.e., the Internet. We completely address this issue by introducing standardized services that will synchronize the clocks of all the components in DELTA’s architecture. We stress that the handling of timestamps is not only relevant security-wise, but it is inherent to DELTA’s functionality, especially in regards to the servicing and monitoring of the status of DR events.

Up to this point, we have introduced mechanisms that guarantee security and privacy of each layer, however, in isolation. To expand these properties across all layers that comprise DELTA’s architecture, we introduce the secure design of DELTA’s peer-to-peer (P2P) network, which is the main medium upon which DELTA components exchange data and is mandated by OpenADR 2.0b to guarantee that DELTA complies to that standard. We stress that we do not tackle only issues regarding confidentiality, integrity and authenticity of the transmitted data, but also fault tolerance, which provides liveness guarantees, i.e., the longevity and operation of the DELTA’s P2P network is guaranteed, even in the presence of failures.

Lastly, we conclude by presenting some additional, security-related use cases which resulted from the research and effort expended in drafting the content presented in this deliverable.

## Table of Contents

<b>1. Introduction .....</b>	<b>10</b>
<b>1.1 Scope and objectives of the deliverable.....</b>	<b>10</b>
<b>1.2 Structure of the deliverable.....</b>	<b>10</b>
<b>1.3 Relation to Other Tasks and Deliverables .....</b>	<b>10</b>
<b>2. Identities .....</b>	<b>11</b>
<b>2.1 X.509 Certificates.....</b>	<b>11</b>
2.1.1 Certificates file extensions.....	12
2.1.2 Certificate fingerprints.....	12
<b>2.2 DELTA Identity Services .....</b>	<b>12</b>
2.2.1 DELTA Identity handling procedures .....	12
2.2.1.1 Enrollment .....	13
2.2.1.2 Revocation.....	14
<b>2.3 OpenADR 2.0b &amp; XML Signatures.....</b>	<b>15</b>
2.3.1 Creation of XML signatures .....	15
2.3.2 Verifying XML signatures.....	16
<b>3. Access Control.....</b>	<b>17</b>
<b>3.1 Attribute Based Access Control (ABAC).....</b>	<b>17</b>
3.1.1 XACML.....	18
3.1.2 NIST Guide to ABAC: Definitions and Considerations.....	21
3.1.3 Next Generation Access Control (NGAC) .....	26
<b>3.2 Role-Based Access Control (RBAC).....</b>	<b>27</b>
3.2.1 RBAC Divisions .....	28
3.2.1.1 RBAC Core.....	28
3.2.1.2 Hierarchical RBAC: .....	29
3.2.1.3 Constrained RBAC: .....	30
<b>3.3 Access Control Schemes: Considerations .....</b>	<b>32</b>
<b>3.4 DELTA Blockchain: Access Control.....</b>	<b>33</b>
3.4.1 Network Level Access Control.....	34
3.4.2 Smart Contract Level Access Control .....	34
<b>3.5 Privacy .....</b>	<b>35</b>
<b>4. Auxiliary Security Infrastructures.....</b>	<b>36</b>
<b>4.1 NTP .....</b>	<b>36</b>
4.1.1 Security Considerations .....	37
<b>4.2 PTP.....</b>	<b>37</b>
4.2.1 Security Considerations .....	39
<b>4.3 RFC 3161 – Timestamp Authority .....</b>	<b>39</b>
<b>5. DELTA Peer-to-Peer Network: Architecture &amp; Security.....</b>	<b>42</b>
<b>5.1 OpenFire.....</b>	<b>43</b>
5.1.1 Configuration.....	43
5.1.2 Setup .....	43
5.1.2.1 Scalability & decentralization.....	43
5.1.3 Privacy .....	44
5.1.4 Authentication .....	45
5.1.5 Fault Tolerance: XMPP Server Redundancy.....	45

5.1.5.1	Computer cluster .....	45
5.2	Conflict policy.....	46
5.3	Username Binding.....	46
6.	Security-oriented aspects of FEID-related Use Cases .....	48
6.1	FEID.....	48
6.1.1	FEID Hardware Installation.....	49
6.1.2	FEID Hardware Setup and Configuration .....	50
6.1.3	FEID Component Registration .....	51
6.1.4	FEID Component Registration .....	52
7.	Conclusions.....	54
	References .....	55

## List of Figures

Figure 1: Structure of a X.509 v3 digital certificate.....	11
Figure 2: TLS handshake .....	13
Figure 3: Layers of OpenADR communication.....	15
Figure 4: ABAC overview [8] .....	18
Figure 5: Sequence of steps on a request in a XACML (ABAC) mechanism [10].....	19
Figure 6: Generic and solution specific ABAC overview [10] .....	20
Figure 7: (A) XACML Policy Constructs (B) Attribute Names and Values and the Authorization State for Policy 1.....	20
Figure 8: XACML reference architecture .....	21
Figure 9: Basic ABAC scenario.....	22
Figure 10: Core ABAC mechanisms.....	22
Figure 11: Enterprise ABAC Scenario Example.....	23
Figure 12: An example of ACM functional points .....	23
Figure 13: ACM NIST System Development Life Cycle.....	24
Figure 14: ACL Trust Chain.....	25

<b>Figure 15: ABAC Trust Chain .....</b>	<b>25</b>
<b>Figure 16: Assignment and Association Graphs in NGAC .....</b>	<b>27</b>
<b>Figure 17: NGAC Standard Functional Architecture .....</b>	<b>27</b>
<b>Figure 18: RBAC core elements and their interconnections. ....</b>	<b>29</b>
<b>Figure 19: Hierarchical RBAC – Elements and their interconnection.....</b>	<b>30</b>
<b>Figure 20: Role hierarchies for a project: a) Role hierarchy, b) Administrative Role Hierarchy and, c) Private and Scoped Roles .....</b>	<b>31</b>
<b>Figure 21: Static Separation of Duty relations, Constrained RBAC – elements and their interconnection. ....</b>	<b>32</b>
<b>Figure 22: Dynamic Separation of Duty relations, Constrained RBAC – elements and their interconnection .....</b>	<b>32</b>
<b>Figure 23: An NTP Application Example with 4 Stratum.....</b>	<b>36</b>
<b>Figure 24: Simplified PTP communication/sync scheme.....</b>	<b>39</b>
<b>Figure 25: TSA timestamping procedure.....</b>	<b>40</b>
<b>Figure 26: P2P network classification and node interconnections.....</b>	<b>42</b>
<b>Figure 27 OpenFire “Clustering” function .....</b>	<b>44</b>
<b>Figure 28 OpenFire Encryption Protocols .....</b>	<b>44</b>
<b>Figure 29 OpenFire Authentication Functions .....</b>	<b>45</b>
<b>Figure 30: OpenFire Certificate Stores .....</b>	<b>47</b>
<b>Figure 31: Open Fire Trust Certificate Store .....</b>	<b>47</b>
<b>Figure 32: FEID board top view .....</b>	<b>48</b>
<b>Figure 33: Differences between the two TPM versions.....</b>	<b>49</b>
<b>Figure 34: High-level Use Case Diagram .....</b>	<b>50</b>
<b>Figure 35: High-level Use Case Diagram .....</b>	<b>51</b>
<b>Figure 36: High-level Use Case Diagram .....</b>	<b>52</b>

<b>Figure 37: High-level Use Case Diagram .....</b>	<b>53</b>
---	-----------

## List of Tables

<b>Table 1: ABAC (XACML) building blocks [8] .....</b>	<b>18</b>
<b>Table 2: Terminology pertaining to the main entities of XACML .....</b>	<b>19</b>
<b>Table 3: ABAC mechanism steps.....</b>	<b>19</b>
<b>Table 4: ABAC policies and their role.....</b>	<b>24</b>
<b>Table 5: RBAC vs ABAC characteristics .....</b>	<b>32</b>
<b>Table 6: Allowed PTP implementations and relative performances .....</b>	<b>38</b>



## List of Acronyms and Abbreviations

Term	Description
VPP	Virtual Power Plant
BRP	Balance Responsible Player
DR	Demand Response
DSO	Distribution System Operators
DVN	Delta Virtual Node
FEID	Fog Enabled Intelligent Device
I <sup>2</sup> C	Inter-Integrated Circuit
P2P	Peer-to-Peer
SPI	Serial Peripheral Interface
TSO	Transmission System Operator
UART	Universal Asynchronous Receiver-Transmitter
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PRP	Policy Retrieval Point
XACML	eXtensible Access Control Markup Language
ABAC	Attributes Based Access Control
ALFA	Abbreviated Language for Authorization
PBAC	Policy Based Access Control
DoD	Department of Defence
DAC	Discretionary Access Control
MAC	Mandatory Access Control
ACL	Access Control List
NPE	Non-Person Entity
RSA	Rivest-Shamir-Adleman
ECC	Elliptic Curve Cryptography
CA	Certificate Authority
CP	Certificate Policy
XML	eXtensible Markup Language

## 1. Introduction

### 1.1 Scope and objectives of the deliverable

---

This deliverable is associated with Task 5.1 of the DELTA project and provides the definition, design and development principles related to the DELTA's security mechanisms that will provide the necessary framework that will allow for secure information exchange amongst the involved stakeholders.

We survey a large number of standards regarding various topics related to security, such as confidentiality and integrity of transmitted information. Based on our research, we propose schemes that are relevant to DELTA's operations in order to provide for end-to-end security across all layers of the system, while also providing seamless integration of the involved components. On a high-level, this deliverable addresses issues that revolve around:

1. Authentication;
2. Access control;
3. Transport layer security;

The work presented here was tailored to account for diverse ecosystems that involve multiple parties or stakeholders, as well as, a myriad of services, all of which lie at the core of DELTA.

### 1.2 Structure of the deliverable

---

The work presented in this deliverable is structured as follows.

- Chapter 2 presents concepts related to digital identities that need to be included in the DELTA security framework, including the necessary adaptations required to support features of OpenADR
- Chapter 3 introduces relevant and mature standards for access control and how these can be applied in the DELTA layers. An initial design of the access control policies revolving one of DELTA's cybersecurity infrastructure services, i.e., DELTA's blockchain, pertaining to both the network, as well as, the smart contract level.
- Chapter 4 introduces additional security infrastructures that resolve the issues that stem from the asynchronous nature of the underlying network, i.e., the Internet, that, if left unchecked, can have severe consequences, both in terms of security, as well as, privacy.
- Chapter 5 presents the DELTA's P2P network architecture, security and configuration.
- Chapter 6 discusses additional, security-oriented details pertaining to DELTA's use cases with special focus on FEIDs.
- Finally, the manuscript is concluded in Chapter 7.

### 1.3 Relation to Other Tasks and Deliverables

---

The functional and technical requirements derived in WP1, in particular pertaining to the deliverables D1.2 and D1.3, constitute the basis upon which the contents of this deliverable were drafted. The outcome of this report provides valuable input to the development activities of WP3, WP4 and WP5 in regards to both the design, as well as, the implementation of all layers in DELTA's architecture.

## 2. Identities

The goal of this section is to introduce fundamental concepts related to digital identities and illustrate the means under which a unique identity will be associated with each actor or stakeholder that participates in the DELTA platform. Since the DELTA project employs the well-known standard of OpenADR, parts of the DELTA security framework are based on the ones specified in that standard. The security framework provided by the OpenADR Alliance establishes strict requirements to be met and allows the final security scheme to be determined, in a standardized fashion, by individual DR program deployments. OpenADR 2.0b uses common security mechanisms, like Transport Layer Security (TLS), and the DR program operator will need to enforce the appropriate level of security for their system, in order to, among others, define the permissions of some actors.

First, a description of the current certificate technology used by OpenADR is provided, i.e., X.509 certificates [1], describing the protocol overview and detailing the types of existing certificates. Then, we elaborate on the topic of Certification Authorities (CAs). Following, some bespoke features are defined or explained, such as the registration and the revocation processes of certificates and the operation of encryption mechanisms. Next, the implementation of XML signatures is explained, which is a feature suggested by the OpenADR Alliance to strengthen the security of DR deployments. Finally, the access roles for the access control list (ACL) of DELTA are detailed.

### 2.1 X.509 Certificates

Information systems usually handle the identity of the components within by providing them a certificate that uniquely identifies them. OpenADR provides for this via a well-known mechanism that binds identity names to their respective digital identities, which is based on X.509 certificates. As this is an industry-wide standardized approach, we adopt it as well in DELTA.

The application layer of the Open Systems Interconnection (OSI) model encompasses several security-related protocols, such as the X.500 protocol. This protocol documents a set of computer network standards that cover a large variety of use cases and services, such as electronic directory services. However, the specification that turned out to be the most widespread revolves around public-key certificates, a topic which is covered by the X.509 standard.

The X.509 standard was published in 1998 and assumes a strict hierarchical system of Certification Authorities (CAs) for issuing digital certificates to entities/actors. Version 3 of X.509 is flexible and extendable and provides the necessary means to support diverse hierarchies and topologies, such as bridges and meshes, while also being applicable to peer-to-peer (P2P) communication patterns that build on top of OpenPGP. A high-level view of the structure of a basic X.509 is illustrated in Figure 1.

Version
Serial Number
Signature Algorithm ID
Issuer Name
Validity Period
Subject name
Subject PKI
Issuer Unique Identifier
Subject Unique Identifier
Extension

Figure 1: Structure of a X.509 v3 digital certificate.

### 2.1.1 *Certificates file extensions*

There are several file extensions for X.509 that encode, following different approaches, the very same X.509 certificate, which we briefly introduce below:

- **Distinguished Encoding Rules (.der):** Contains the X.509 certificate in binary format; Defined in RFC 5280 [1];
- **Privacy-enhanced Electronic Mail (.pem):** Used for different types of X.509v3 files that contain ASCII data; Defined in RFC 6187 [2];
- **Cryptographic Message Syntax Standard (PKCS#7, .p7b, .p7c):** Defined in RFC 2315 [3]. Used by Windows operating systems for certificate exchange;
- **Personal Information Exchange Syntax Standard (PKCS#12, .p12):** Defined in RFC 7292 [4]. This is a potentially encrypted or signed archive file format for storing many cryptography objects as a single file. It is commonly used to bundle a private key with its X.509 certificate or to bundle all the members of a chain of trust;
- **Canonical Encoding Rules (.cer, .ctr):** This is a .pem or a .der formatted file with a different extension;

### 2.1.2 *Certificate fingerprints*

In an OpenADR deployment, certificate fingerprints are used by the VTN to identify one or more VENs when they connect to it for the first time. We stress that the VTN and the VEN are roles that OpenADR defines for systems depending on their implemented functionalities and involved data flows. In addition, depending on the characteristics of each particular deployment, it may be necessary to also install the certificate fingerprints in the VTN back-end server configuration in order to be able to start exchanging data with VTNs. In addition, VENs must facilitate the registration of a certificate fingerprint, which is assumed to be transmitted out-of-band to the VTN. The fingerprint can be generated by command line tools from an input PEM certificate, or via libraries in various programming languages, such as Python. Unlike normal certificates (encoded as a sequence of bytes), fingerprint certificates have a shorter size.

## 2.2 DELTA Identity Services

---

CA is a general term that encompasses various hardware and software components that create, sign, and issue public key certificates to entities/actors. The CAs are responsible of:

- Drafting and maintaining a Certification Policy (CP).
- Issuing certificates.
- Delivering certificates to its subscribers according to the CP and other applicable policies.
- Revocation of Certificates.
- Generating, protecting, revoking, and operating with CA private keys.
- Certificate lifecycle management ensuring that all aspects of the CA's services, operations, and infrastructure are performed in accordance with the requirements, representations and warranties of its CP.
- CAs act as trusted parties to facilitate the confirmation of the binding between a public key and an identity, as well as, other attributes of the "Subject" field of the certificate.

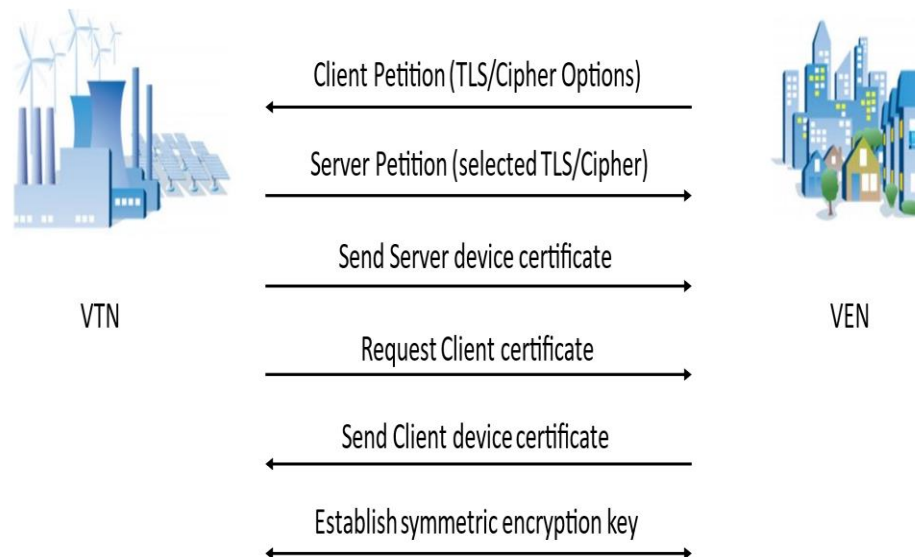
### 2.2.1 *DELTA Identity handling procedures*

DELTA builds on top of OpenADR 2.0b, which requires TLS 1.2, a cryptographic protocol designed to provide secure communications over a computer network. The TLS handshake (Figure 2) is capable

of exchanging client and server device certificates, validate that the certificates are trusted, establish a symmetric encryption key for data exchange, and, finally, set the corresponding cipher suites to use for data encryption and decryption, based on ECC or RSA, according to the following tags:

- ECC - TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256
- RSA - TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256

The RSA and ECC cipher suites used by OpenADR employ the collision resistant SHA256 hash function.



**Figure 2: TLS handshake**

Both VTNs and VENs can be configured to support any TLS version and cipher suite combination based on the needs of a specific deployment. The certificates can be encoded in several formats, as we have already illustrated in Section 2.1.1. These certificates follow a process of enrollment and revocation that will describe in the following sections.

### **2.2.1.1 Enrollment**

Certificate enrollment is the process by which a digital certificate is requested for a specific device. The CA issues and manages the digital certificate to be used within a public key infrastructure (PKI). This PKI is a combination of hardware, software and policies that provide reliable authentication between users and use the information generated between these users for encryption and decryption of messages, digital signatures and others.

Services such as authentication, confidentiality, and integrity in VENs and VTNs rely on PKI certificates, without being limited by specific proprietary technologies. Two levels of security are distinguished in OpenADR, i.e., 'Standard' and 'High'. The main difference between these two is the fact that the 'Standard' security mode uses TLS to establish secure communications between a VTN and one or more VENs, whereas the 'High' security uses, additionally, XML signatures to provide non-repudiation of the transmitted data.

PKIs rely on two main algorithms to digitally sign certificates and encrypt point-to-point communications, i.e., RSA and ECC. ECC has a shorter key size, which makes it more efficient for encryption, decryption and digital signatures. This feature constitutes ECC more favorable for

embedded, computationally restricted devices. However, OpenADR requires that VTNs must support both ECC and RSA.

VENs can use one or more PKI certificates. Since VTNs must support both RSA and ECC, VENs have an option on which one to choose.

In DELTA, the following parameters will be used for the certificates:

- **ECC** – 256 bits or longer keys.
- **RSA** – 3072 bits or longer keys.
- **Certificate types** – X.509v3.

Requiring PKI X.509v3 certificates to be incorporated in the devices ensures a secure two-way communication between compatible devices. The OpenADR CP is consistent with the policies for certification and practices established by X.509 PKI certificates.

### **2.2.1.2 Revocation**

One of the mechanisms that any system that incorporates new components must implement is revocation, which consists of an invalidation process for one or more previously issued certificates. This action causes the rest of the certified entities/actors to know that the credentials of another entity/actor have been revoked and should no longer be considered valid/trustworthy. There are several reasons to revoke certificates, such as:

- Private key is lost or is compromised due to an unauthorized theft or disclosure.
- Violation of agreements with the subscriber.
- The digital certificate agreement has been terminated.
- Incorrect issuance or defect in the certificate.
- For false information or any other circumstance that may affect the reliability, security or integrity of the certificate.
- The identity that the certificate has is no longer valid.
- Attributes asserted in the subscriber's certificate are incorrect.
- The certificate was issued without the consent of the certificate recipient.
- The subscriber's organization name changes.
- The use of the certificate is harmful due to improper use by the device.
- The CA determines that the certificate must be revoked.

The main entities that can revoke the certificates are: a) the owner of the certificate, or any authorized representative of the owner, b) the CA, as far as the certificates are within its domain and, c) the OpenADR PKI-PA.

A certificate revocation request must contain a timestamp, the certificate to be revoked, and a description of the incident that led to its revocation. The party issuing the request must also be properly authenticated by the CA. There several ways based on which the issuing party can authenticate itself, such as:

- Logging into his account and revoking the certificate through a web portal. These web portals typically employ a two-factor authentication system.
- Providing appropriately formatted confirmation via telephone, signed fax, signed email, postal mail, courier service, or any other similar procedure.
- Through a corporate representative, administrator, legal or technical contact.

Following the revocation of one or more certificates, the CA publishes a Certificate Revocation List (CRL) to its certificate repository, which is responsible for distributing digital certificates.

## 2.3 OpenADR 2.0b & XML Signatures

XML is a W3C specification that defines a general purpose markup language, like HTML. The main purpose of the language is to share data across different systems. An XML signature is a W3C recommendation that defines XML syntax for digital signatures, which are used to sign data of resources of any kind based on cryptographic algorithms that provide for data integrity and non-repudiation. OpenADR establishes these signatures as additional security mechanisms to be considered.

These signatures can be applied to more than one type of resource, such as sections of a XML document, plain text, encoded files, and others. This feature characterizes the XML signature with the ability to sign sections of important resources when, e.g., the integrity of different sections of such resource must be preserved. This allows working on subsets of data that have multiple signatures. These XML signatures can be of three types:

- Enveloped, where the signed element is a parent and the signature element is a child.
- Enveloping, where the signature element is the parent of the element being signed.
- Detached, where the signature element and the signed element do not have a parent-child relationship.

To ensure high security in OpenADR, the last option will be used, which separates signatures from their siblings. Finally, conceptually, this XML layer will be encapsulated in the last layer of communication in OpenADR (Figure 3).

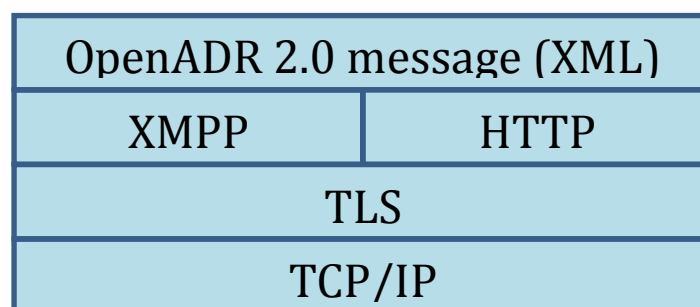


Figure 3: Layers of OpenADR communication

### 2.3.1 Creation of XML signatures

In an XML signature document, we must specify the following components, which will be necessary for the proper functioning of the encrypted communication:

- **Reference URI:** To allow the system to identify the signed resource, this element will provide a reference to the resource that is signed by a URI.
- **Digest value:** This element will contain the hash of the resource to be signed.
- **Signature value:** This element contains the digest value signed by the private key of the party transmitting the payload.
- **Key info:** The key info element contains the information of the key that should be used for signature verification. This contains information on the public key of the party transmitting the payload.

To create XML signatures, we must first identify the resource to be signed and add it to the Reference URI element.

### **2.3.2    *Verifying XML signatures***

To verify an XML signature, the following requirements must be met:

- Check that the verification value is the same as the digest value.
- Ensure that the calculated digest of the `<oadrSignedObject>` field is the same as the value in the `<DigestValue>` field.
- Verify if a `<ReplayProtect>` element is contained as `<SignatureProperty>`. Reject the payload if the current date and time on the device differs from the value in the `<ReplayProtect>` more than a predefined value. In addition, the nonce field may be used for further protection against replay attacks.



### 3. Access Control

In this section, we discuss issues revolving around access control standards, their appropriateness in regards to deployment, as well as, the employed approach for access control handling within DELTA's ecosystem.

The DELTA platform relies on the communication of different virtual entities that must meet some key functional requirements. These virtual entities are software components of the platform that must be uniquely identified with a specific role, which will define with whom they can communicate in the platform. In other words, these relationships are encoded in the access control list (ACL) that the consortium has established based on the data requirements and use cases (D1.1) [17], as well as, the data exchanges presented in D1.2 and D1.3. Based on these deliverables, we have identified the following entity roles:

- **DVNs:** The DELTA Virtual Nodes (DVNs) have direct control over the FEIDs deployed in DELTA. Although they can control such components by means of DR signals, fetch their data (or except the FEIDS to provide pieces of data), the DVNs are not able to choose with which FEIDS they are allowed to interact with.
- **Aggregator:** The Aggregator is the DELTA component that has the overall control of the platform. The Aggregator controls the DVNs in the platform and, among others, is responsible for assigning FEIDS to DVNs. Lastly, the Aggregator is a privileged user that has access to all of the interfaces that the FEIDs expose.
- **Trustee:** This role refers to some entities (e.g., DSOs/TSOs) that should be trusted and have various special privileges on the platform, which are granted to them by the Aggregator. These entities mainly interact with the Aggregator and the DVNs.
- **FEID:** The FEID is the component that lies closest to the end customer. Each FEID is assigned by the Aggregator to a specific DVN, following the output of the Aggregator's segmentation process.
- **Technicians:** These are experts that install the FEIDs at the customers' houses. They require some special access to the DELTA platform in order to check that the FEID's installation has been completed successfully, which entails some limited form of interaction with the interfaces of the Aggregator and the DVN.
- **Customers:** They are allowed to monitor their FEIDS, and therefore, they must be able to fetch data from this component. Generally speaking, customers are mainly able to read a subset of the data that the FEIDs maintain/report. Their write permissions revolve around availability schedules for DR events. Lastly, customers are able to interface with the Aggregator's social collaboration platform and (partially) with its gamification engine.

#### 3.1 Attribute Based Access Control (ABAC)

In this mechanism, attributes assigned to users and resources determine "who can access what" under specific conditions. In general, attributes characterize users, resources, and the operating environment. ABAC evolved from RBAC (Section 2.2.2) and it can subsume it. Key standards implementing ABAC are the eXtensible Access Control Markup Language (XACML) and the Abbreviated Language for Authorization (ALFA, [5]).

ABAC can be seen through the following dimensions: externalized authorization management, dynamic authorization management, policy-based access control, fine-grained authorization. It consists of an architecture, attributes and policies [6]. Applications of ABAC are available in the following areas:

- API and micro-services security
- Application security

- Database security
- Data security
- Big data security
- File server security

A prediction [7] made by Gartner states that by 2020 the 70% of organizations will use ABAC. Figure 4 introduces an overview of the ABAC mechanism.

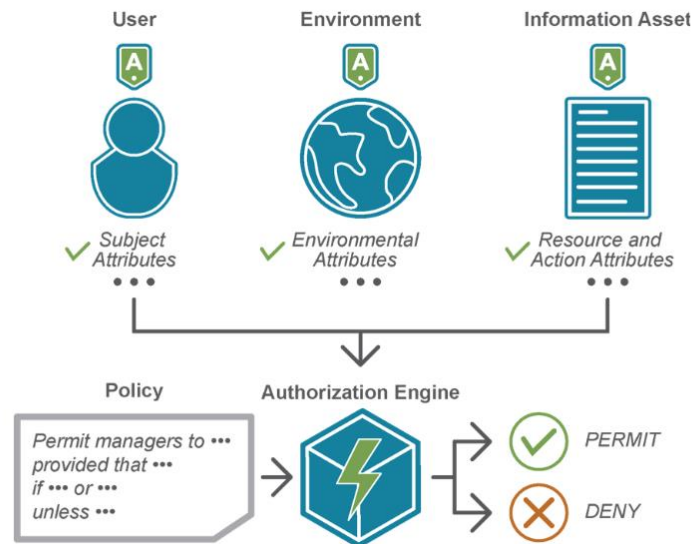


Figure 4: ABAC overview [8]

Table 1 introduces the building blocks of ABAC as implemented by employing the XACML standard.

Table 1: ABAC (XACML) building blocks [8]

Term	Definition
Subject	Who or what is demanding access to an information asset.
Action	Action the subject wants to perform
Resource	The information asset or object impacted by the action
Environment	The context in which access is requested

The involved steps that need to be followed in order to complete define an ABAC mechanism are as follows:

- Gathering the authorization requirements of an organization.
- Identifying required attributes for fulfilling the requirements.
- Authoring the authorization policies.
- Deploying the policies.
- Defining Policy Information Points (PIPs).
- Validating Policies.

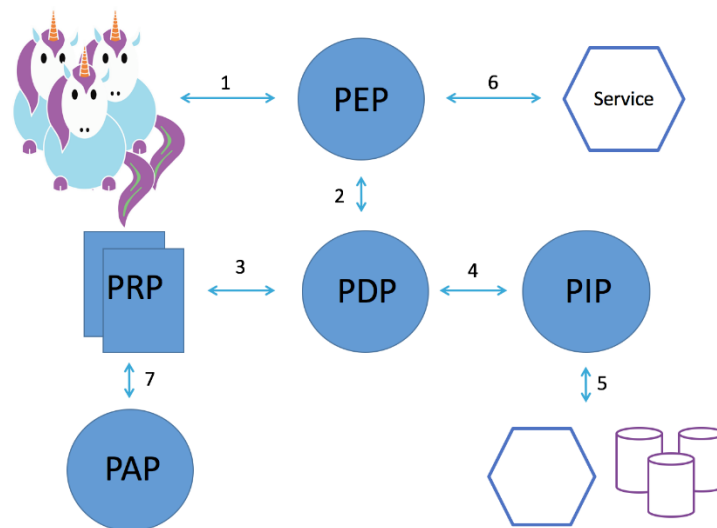
### 3.1.1 XACML

XACML is an attribute-based access control policy language and a processing model describing how to evaluate access requests according to the rules defined in policies [9]. Table 2 presents the terminology pertaining to the main entities of XACML.

**Table 2: Terminology pertaining to the main entities of XACML**

Abbr.	Term	Description
PAP	Policy Administration Point	Point which manages access authorization policies
PDP	Policy Decision Point	Point which evaluates access requests against authorization policies before issuing access decisions
PEP	Policy Enforcement Point	Point which intercepts user's access request to a resource, makes a decision request to the PDP to obtain the access decision (i.e. access to the resource is approved or rejected), and acts on the received decision
PIP	Policy Information Point	The system entity that acts as a source of attribute values (i.e. a resource, subject, environment)
PRP	Policy Retrieval Point	Point where the XACML access authorization policies are stored, typically a database or the filesystem

ABAC mechanisms are often called Policy-based Access Control (PBAC) mechanisms due to the key role of policies within the mechanisms. Access requests to resources are evaluated against defined policies. A “Permit” or “Deny” state is returned accounting for the user, resource, and environment attributes. Another example where the ABAC system is implemented through the XACML standard is the presented in Figure 5. The steps presented in the figure are outlined in Table 3.



**Figure 5: Sequence of steps on a request in a XACML (ABAC) mechanism [10]**

**Table 3: ABAC mechanism steps**

<b>Step 1</b>	A request comes in for a service
<b>Step 2</b>	The Policy Enforcement Point (PEP) intercepts the request and passes it along to the Policy Decision Point (PDP)
<b>Step 3</b>	The PDP fetches the policies from the Policy Retrieval Point (PRP)
<b>Step 4</b>	The PDP attempts to evaluate the policies and calls the Policy Information Point (PIP) for any missing attributes
<b>Step 5</b>	The PIP calls out to other services or data stores to retrieve the required attributes
<b>Step 6</b>	Assuming the policy returns Approve, the request is forwarded to the service
<b>Step 7</b>	While not being part of the main flow, the Policy Administration Points (PAP) allows for editing the policies.

Figure 6 presents an example where generic high-level interactions (Figure 6.A) are present and specific application steps for a solution provided for commercial purposes. Steps similar to what is described in Table 2 apply to the solution specific example of Figure 6.B.

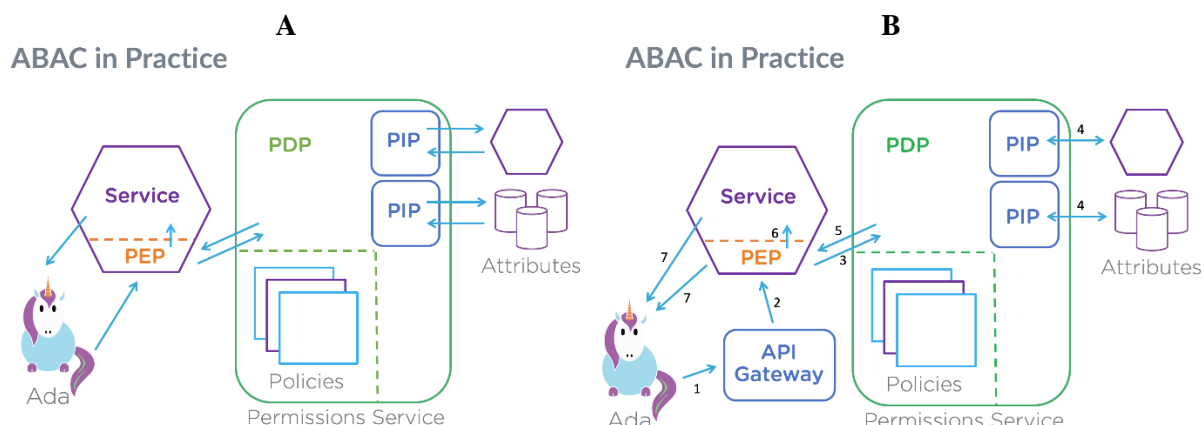


Figure 6: Generic and solution specific ABAC overview [10]

In XACML, policy definitions take place with the use of XML, which leads to limited readability issues. In order to overcome this problem the ALFA language was introduced. ALFA is a language that maintains XACML model but uses JSON instead of XML for policy definition, which simplifies a lot the process of policy definition as well as development work.

Figure 7.A outlines how XACML access policies are structured. Combinatorial algorithms (deny-overrides, permit-overrides) are used for combining multiple local decisions into a single global decision. Obligation is a directive from the PDP to PEP on what must happen before or after an access request is approved or denied. An example policy document is illustrated Figure 7.B

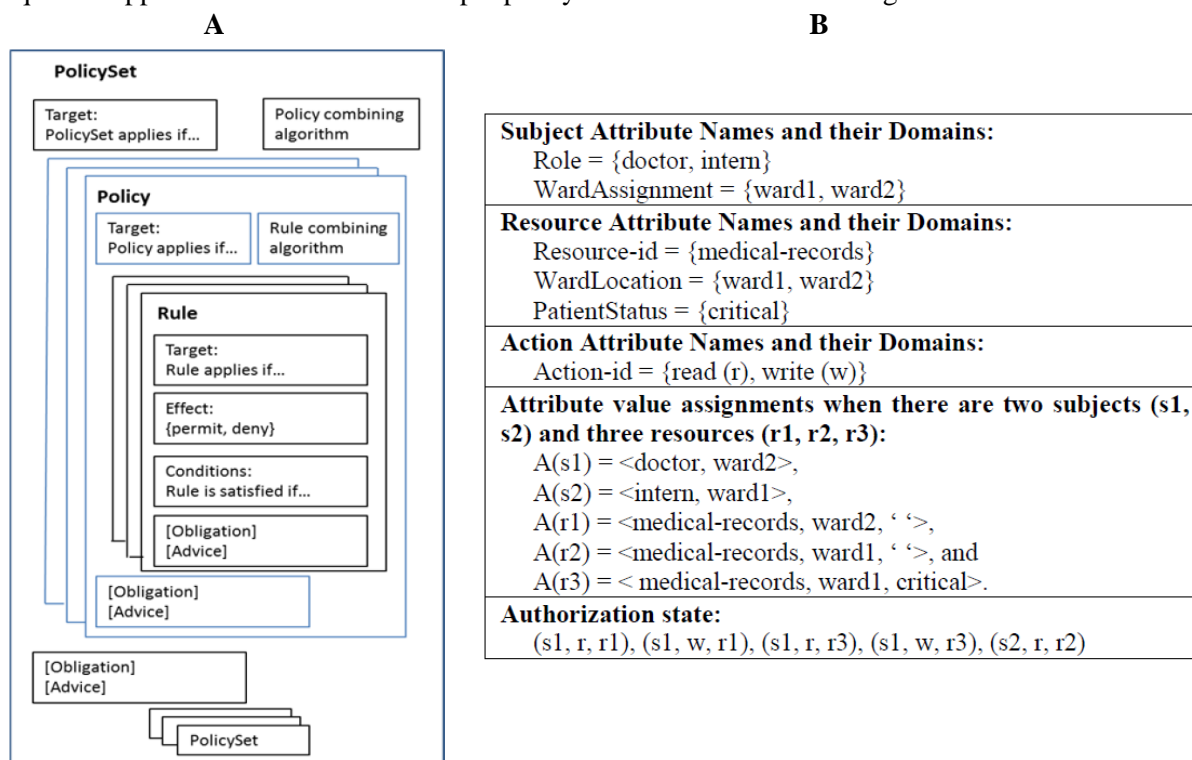
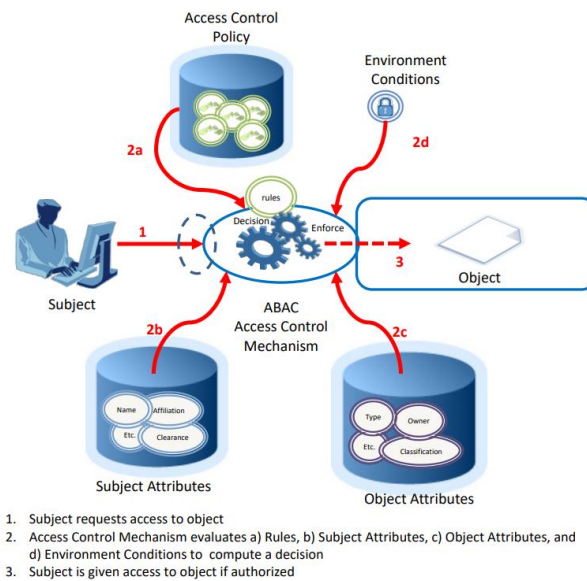


Figure 7: (A) XACML Policy Constructs (B) Attribute Names and Values and the Authorization State for Policy 1

The diagram illustrates the interaction between various components in a Policy Enforcement Point (PEP) and Policy Decision Point (PDP) architecture. At the top, an **Application** (represented by a person icon) sends **OE ops** (Operating Environment operations) to the **PEP**. The **PEP** is part of the **Operating Environment (OE)** and is connected to the **RAP** (Resource Access Point) and **content** (represented by a cylinder icon). The **PEP** also sends data to the **PDP** (Policy Decision Point). The **PDP** is connected to the **PIP** (Policy Information Point) and **PRP** (Policy Retrieval Point), which are both represented by cylinder icons. The **PIP** and **PRP** are connected to the **PAP2** (Policy Administration Point 2) and **PAP1** (Policy Administration Point 1), respectively, which are represented by person icons. The **PAP2** and **PAP1** are connected to the **Administration** layer. The **PEP** is also connected to the **Enforcement** layer, which is connected to the **Policy Enforcement Point Resource Access Point**. The **PDP** is connected to the **Decision** layer, which is connected to the **Policy Decision Point**. The **PIP** and **PRP** are connected to the **Access Control Data** layer, which is connected to the **Policy Information Point Policy Retrieval Point**.

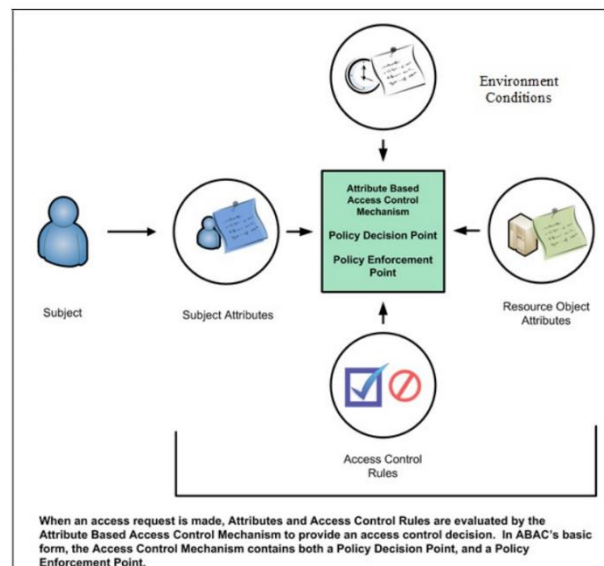
### 3.1.2 NIST Guide to ABAC: Definitions and Considerations

Next, RBAC systems [12] emerged by “employing pre-defined roles that carry a specific set of privileges associated with them and to which subjects are assigned”. Both ACLs and RBAC can be considered as special cases of ABAC model since ACLs are associated with the attribute of “identity” and RBACs with the identity of “role” [13]. The main characteristic of ABAC is the use of policies for the evaluation of a complex Boolean rule set. As mentioned above, XACML is an access control mechanism consistent with ABAC. In XACML, elements, such as rules, policies, rule and policy combining algorithms, attributes (subject, object/resource, action and environment conditions), obligations and advice, are employed. Figure 9 presents a basic ABAC scenario with all the relevant entities, i.e. subject(s) and object(s) with their attributes, access control mechanisms and policies, as well as environment conditions.



**Figure 9: Basic ABAC scenario**

Figure 10 presents the core ABAC mechanism. A request event leads to the evaluation of Attributes and Access Control Rules by the control mechanism and a control decision takes place. Boolean combinations of attributes and conditions, as well as, relations associating subject and object attributes and allowable operations are used.



**Figure 10: Core ABAC mechanisms**

An enterprise is defined as a collaboration or federation among entities for which information sharing is required and managed [6]. Figure 11 presents such a scenario.



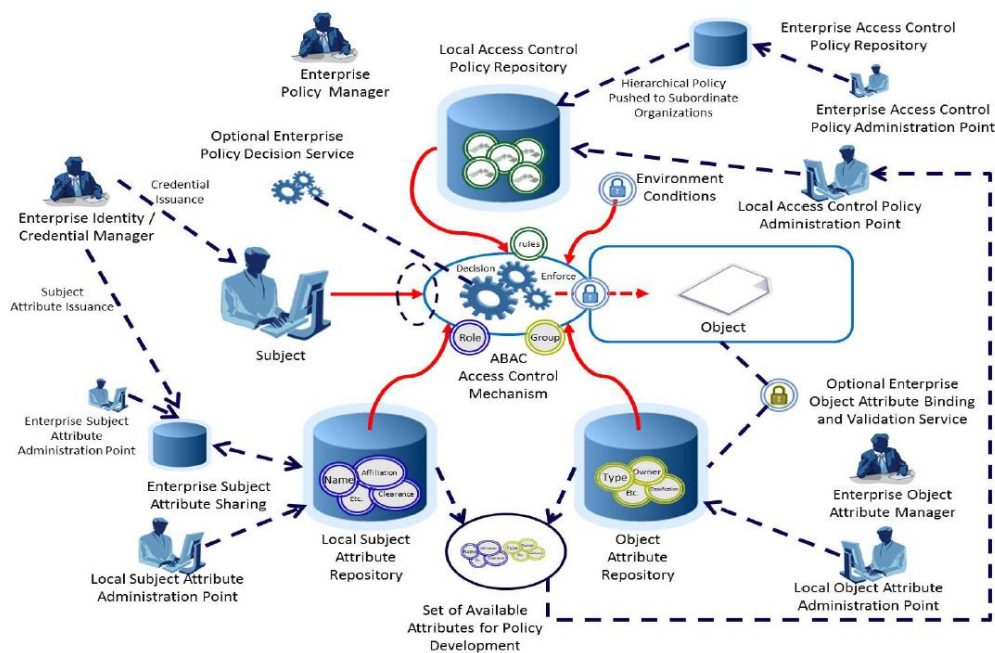


Figure 11: Enterprise ABAC Scenario Example

Natural Language Processes (NLPs), Digital Policies (DPs), Meta-policies (MP) and Meta-attributes are all utilized in a specific way for the realization of an enterprise ABAC solution. Figure 12 presents an example of Access Control Mechanism (ACM) functional points.

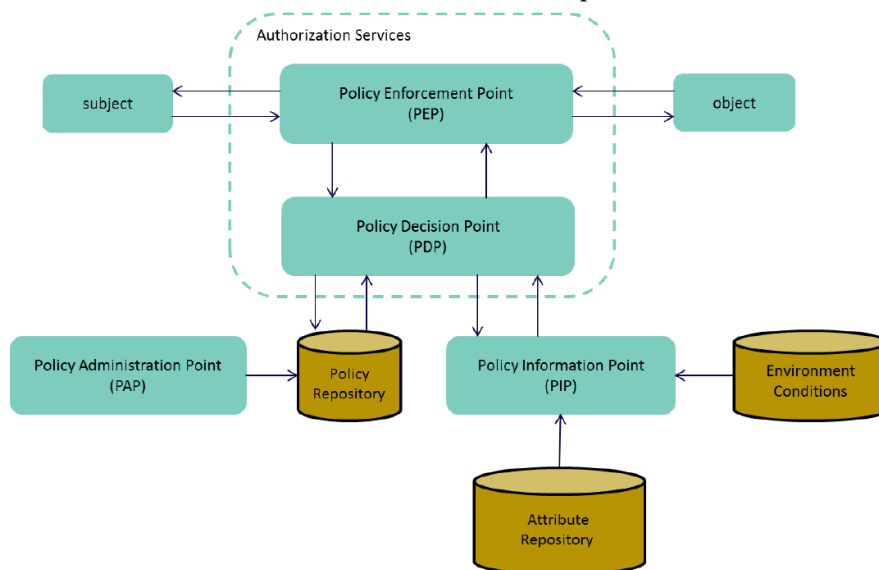


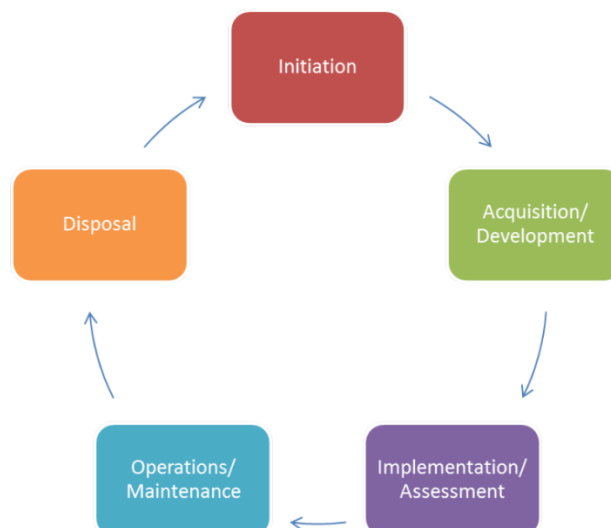
Figure 12: An example of ACM functional points

**Table 4: ABAC policies and their role**

<b>Policy Decision Point (PDP)</b>	Computes access decisions by evaluating the applicable DPs and MPs. One of the main functions of the PDP is to mediate or deconflict DPs according to MPs.
<b>Policy Enforcement Point (PEP)</b>	Enforces policy decisions in response to a request from a subject requesting access to a protected object; the access control decisions are made by the PDP.
<b>Policy Information Point (PIP)</b>	Serves as the retrieval source of attributes, or the data required for policy evaluation to provide the information needed by the PDP to make the decisions.
<b>Policy Administration Point (PEP)</b>	Provides a user interface for creating, managing, testing, and debugging DPs and MPs and storing these policies in the appropriate repository

Figure 13 presents the ACM NIST System Development Life Cycle (SDLC). Each phase is further elaborated in the document. In the Initiation Phase, the following considerations are accounted for:

- Building the business case for deploying ABAC capabilities
- Scalability, feasibility, and performance requirements:
  - Development and maintenance cost
  - Cost of transition to ABAC
  - Need to review privilege and monitor authorizations
  - Understanding object protection requirements
  - Enterprise governance and control
- Developing operational requirements and architecture:
  - Object identification and policy assignment
  - Attribute architecture
  - Subject attributes
  - Object attributes
  - Environment condition
  - Access control rules
  - Access control mechanism and context handling



**Figure 13: ACM NIST System Development Life Cycle**

In this phase, the ACL trust chain and ABAC trust chain are defined. Both are presented in Figure 14 and Figure 15 below.



### ACL Trust Chain

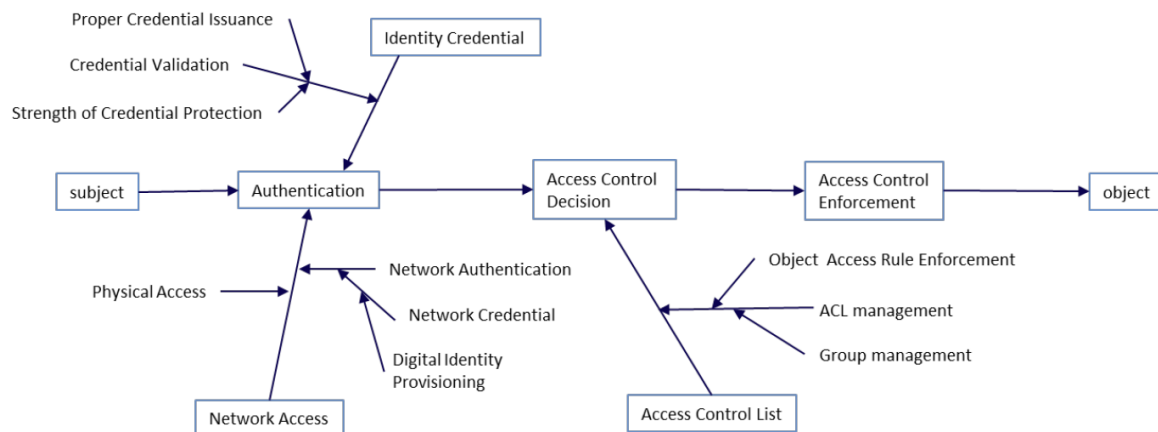


Figure 14: ACL Trust Chain

### ABAC Trust Chain

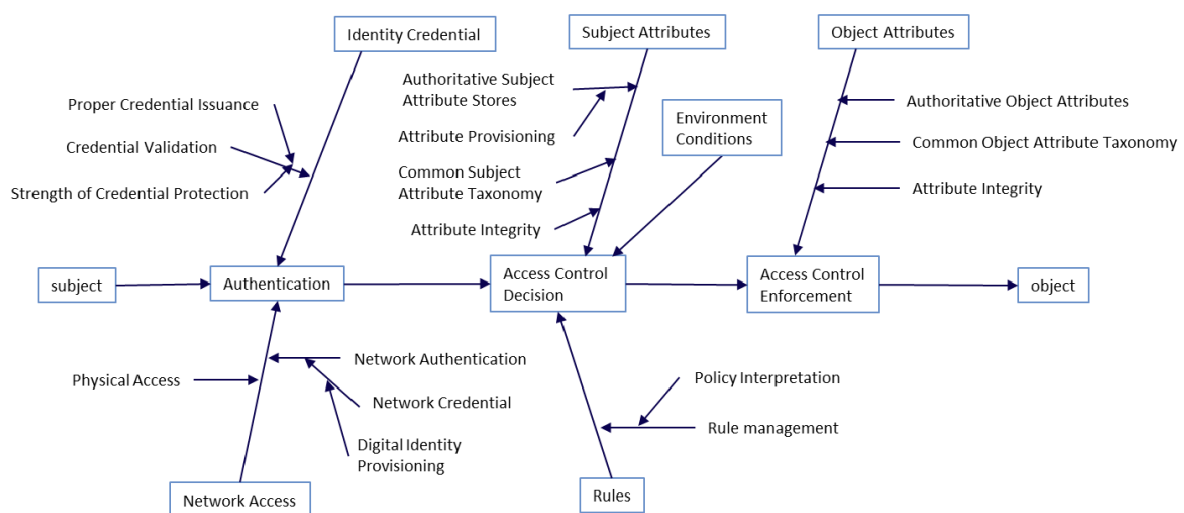


Figure 15: ABAC Trust Chain

In the Acquisition/Development Phase, the following considerations are accounted for:

- Business process generation and deployment preparation:
  - Documentation of rules
  - Customizing policy
  - Agreement and understanding of attributes
  - Understanding meaning of attributes
  - Processes and procedures for access failures
  - Attribute privacy considerations
  - Digital policy creation and maintenance
- System development and solution acquisition considerations:
  - Standardization and interoperability within the enterprise
  - Identity management integration
  - Support of NPEs (Non-person entities)

- Authentication and data integrity between ABAC components
- Integrating other controls with ABAC
- Selection and accessibility of attribute sources
- A shared repository for subject attributes
- Minimum attribute assignments
- Environment conditions
- Attribute management
- NLP/DP traceability
- Rules or Policies based on the agreed attributes
- Externalization of policy decision services
- Considerations for other enterprise ABAC capabilities:
  - Confidence in access control decisions
  - Mapping attributes between organizations

In the Implementation/Assessment Phase, the following considerations are accounted for:

- Attribute caching
- Attribute source minimization
- Interface specifications

In the Operations/Maintenance Phase, the following considerations are accounted for:

- Availability of quality data

### 3.1.3 Next Generation Access Control (NGAC)

The Next Generation Access Control (NGAC) mechanism [14]-[15] is a fundamentally different approach than XACML in terms of representing requests, defining and managing policies, attributes, as well as, computing and enforcing decisions. The terms *user*, *operation* and *object* are used instead of *subject*, *action*, and *resource* in XACML. Object attributes are used in a similar manner as in XACML. For subject attributes, NGAC uses containers that represent roles, affiliations and other common characteristics relevant to policies, e.g., security clearances.

In NGAC, policies are expressed through configurations of relations of four types: *assignments* (define membership in containers), *associations* (to derive privileges), *prohibitions* (to derive exceptions), and *obligations* (to dynamically alter access state). Figure 16 presents two examples with assignment and association graphs in NGAC. Tuples are used for assignment specification.

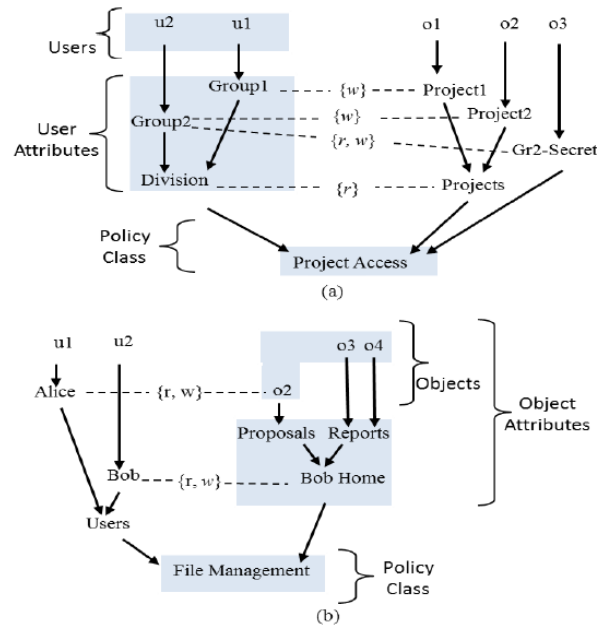


Figure 16: Assignment and Association Graphs in NGAC

In NGAC, three types of prohibition relations are included, i.e., user-deny, attribute-deny, process-deny. Obligations are exclusively used for the creation of process-deny relations. They consist of a pair of an event pattern and a response. If *event* pattern conditions are met, then responses are executed. NGAC decision functions control accesses in terms of processes. Other aspects such as *Delegation*, *NGAC Administrative Commands and Routines*, and *Arbitrary Data Service Operations* are elaborated as well. Figure 17 presents the NGAC standard functional architecture. The chain of events that take place when an application (a request for accessing content) is initiated is presented.

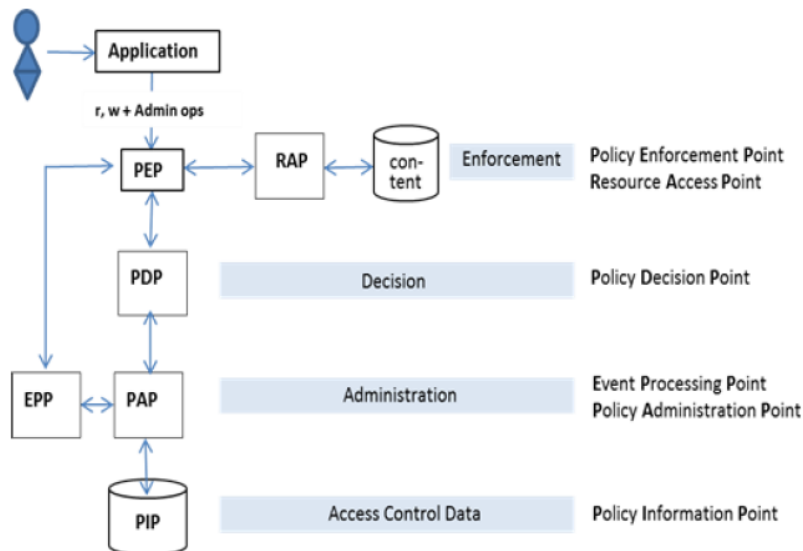


Figure 17: NGAC Standard Functional Architecture

### 3.2 Role-Based Access Control (RBAC)

One of the access mechanisms that is well-described in the literature is the Role-Based Access Control (RBAC) mechanism. It originated back in the 1970's for the needs of multi-user, multi-application systems. This technique is based on the concept of roles and groups of users to whom these roles are

assigned. The roles are representing various tasks to be performed and the users are associated with permissions to the tasks, as well as, constraints with respect to the access to these roles (tasks). The permissions to roles can change and user groups can change as well. It is considered hard to assign permissions to users without the RBAC mechanism. A role represents a gathering of users on one side, whereas on the other it shows a gathering of permissions.

RBAC encloses in an access control model the concepts of roles, hierarchies, role activation, constraints on user/role membership and role set activation. RBAC posits a good solution when simplification of security policy administration is sought. It is a technique that experiences an increase in the number of vendors that offer its features.

In RBAC [16], there are four types of components which are presented and the functional specifications for each one of them is given. The RBAC features that are described represent a stable set of RBAC features and a variety of products entail these features. These features fall into three categories, i.e., administrative operations, administrative reviews, system level functionality. In order to identify such features, a sponsored market analysis has been conducted by NIST. In [17], the reference models of RBAC are presented, which correspond to the components introduced in [16].

RBAC has also been discussed in order to produce a standard. For this there have been panel sessions during the 2000 ACM Workshop on Role-Based Access Control. Related to standards, it should also be mentioned here that the concept of roles, as they are used in RBAC is used in the SQL3 standard for database management systems, which is based on their implementation in Oracle 7.

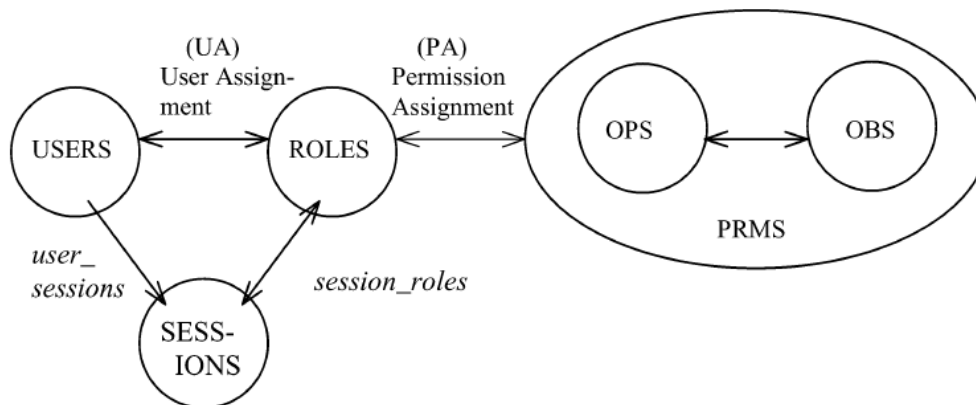
### **3.2.1 RBAC Divisions**

RBAC is falsely considered as a single access control and authorization model. In reality, it consists of several models, which are suitable for different security management applications.

RBAC entails a family of three models. The RBAC core is the basic model that is needed in order to have the RBAC mechanism. The other two models are built on top of the core model and referred to as hierarchical RBAC and constrained RBAC. Hierarchical RBAC introduces the concept of hierarchies, meaning that permissions can be inherited between different roles. Constrained RBAC adds the notion of constraints, meaning that there are restrictions among its defined components, as well as, restrictions regarding the permissions given to the user. In the following subsections, we provide a brief overview of these models.

#### **3.2.1.1 RBAC Core**

This model encompasses the basic concepts of RBAC. On the one hand, users on one hand and permissions are assigned to roles. On the other hand, users acquire permissions when they are members of roles. Users and roles should be defined carefully. Activation and deactivation of roles can be done selectively, through the concept of user sessions. The characteristics of group-based access control apply in general lines for the base model of RBAC. Excluding specific features is an important issue in core RBAC [17]. We stress that RBAC core is the building block for all the previously cited versions of RBAC. Figure 18 illustrates how the elements of RBAC core are interconnected.



**Figure 18: RBAC core elements and their interconnections.**

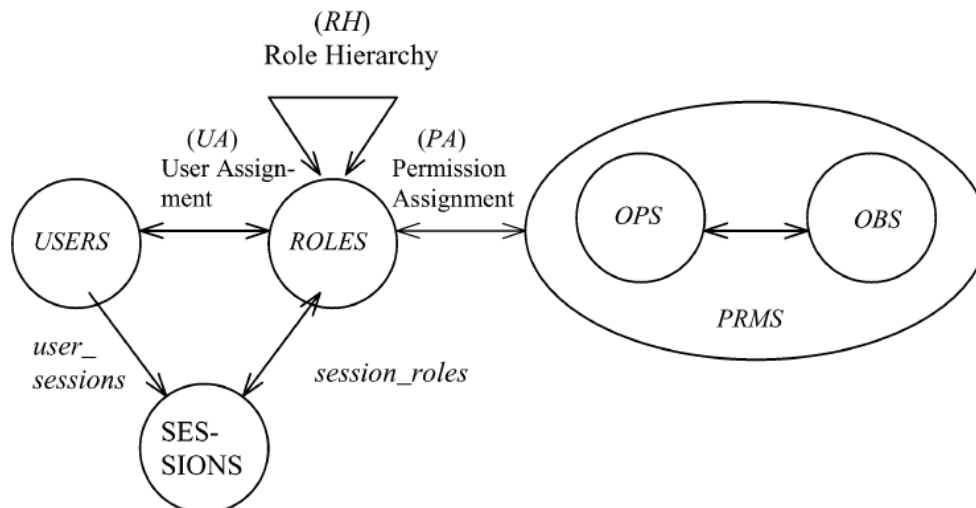
The arrows in Figure 18 entail that multiple roles can be assigned to one user and vice versa; the same goes for permissions and role assignments. Permissions are exercised by users. The session stands for the assignment of many roles to one user, i.e., a user sets a session where some of the roles are activated from the total set of roles that this user is assigned to.

Regarding the functional specifications of RBAC core, the functions needed to be dealt with are: administrative functions, supporting system functions and review functions. Administrative functions revolve around the creation of element sets. For instance, users and roles are created and deleted by the administrator, whereas the administrator is responsible for setting the relationships between roles, operations and objects. The main relationships to be defined are role-to-user and role-to-permission. Supporting system functions revolve around session creation and deletion. The review functions provide the necessary interfaces to review the results of the actions created by the administrative functions.

### **3.2.1.2 Hierarchical RBAC:**

The main difference of hierarchical RBAC from RBAC core is that it introduces role hierarchies. There is a relation between roles according to which senior roles acquire the permissions of junior roles, whereas junior roles acquire user membership of seniors. Hierarchical RBAC is partitioned into two categories (i.e., general and limited). According to the former one, multi-layer inheritance of permissions and user membership of intermediary roles is allowed. The latter scheme implies that restrictions are set on the role hierarchy; (inverted) tree-like structures are supported. In this technique, multiple inheritances are usually not supported, whereas roles are limited to one immediate actor/ user lower in the hierarchy [17].

This model can be efficient when it comes to general-permission attributes. For instance, in cases where several general permissions are assigned to a large number of users, it is considered more effective not to assign general permissions repeatedly. Moreover, common permissions can be attributed to users belonging to different roles. In hierarchical RBAC, a user can set a session with the roles that are considered junior in hierarchy regarding the roles that the user is a member of. Actors/users higher in the hierarchy can inherit role permissions from users in lower levels of the hierarchy, whereas user membership is passed in a top-down approach. Figure 19 illustrates how elements interact with each other in this model and how role hierarchies can be designed for a project.



**Figure 19: Hierarchical RBAC – Elements and their interconnection**

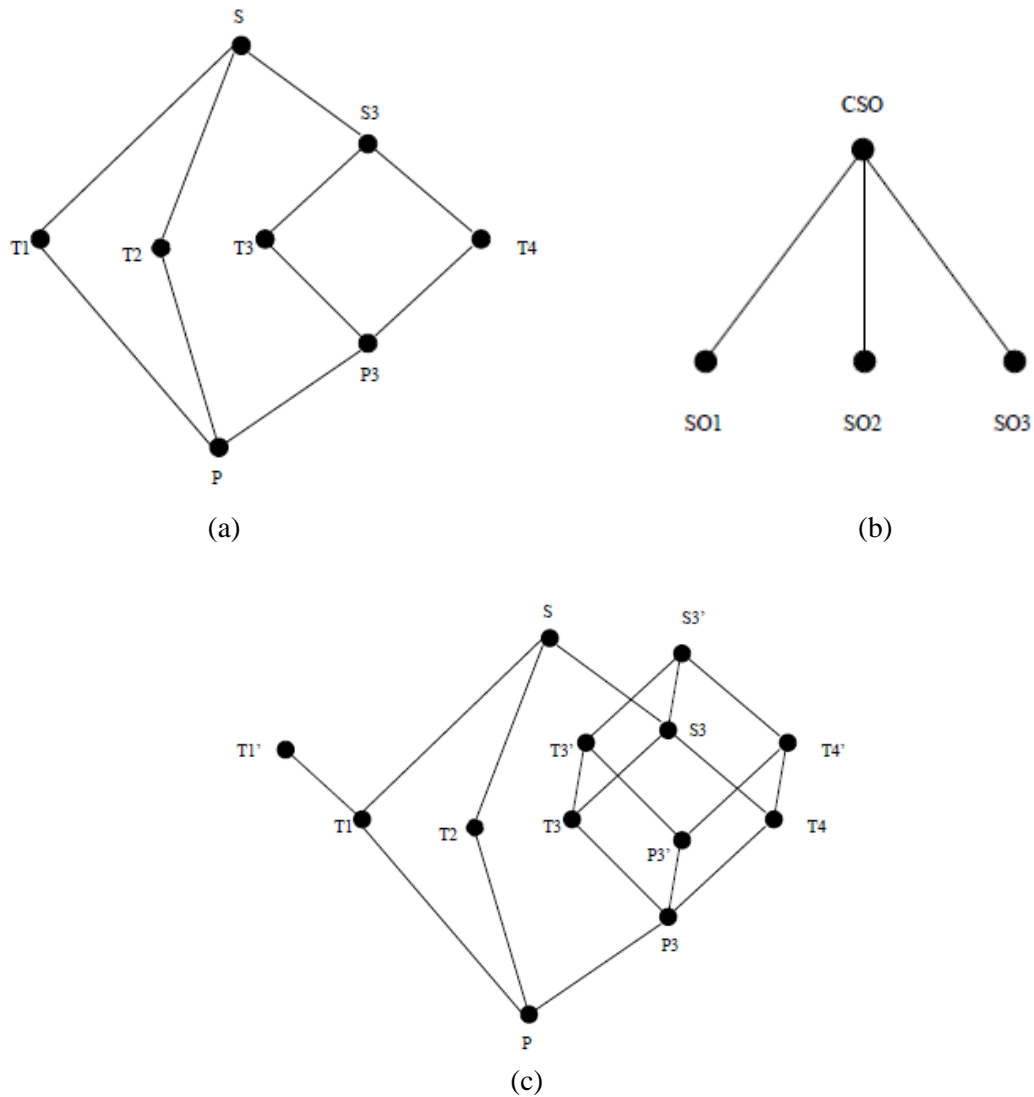
The functional specifications of hierarchical RBAC are: hierarchical administrative functions, supporting system functions and review functions. The first category includes all the administrative functions necessary for the RBAC core model. The functions that differentiate this model from RBAC core are the ones to create or delete inheritance relationships between existing roles. Thus, establishing or removing a new inheritance relationship between existing roles and creating a new role and setting it for a user/ actor in the hierarchy are part of the administrative functions of this model. Regarding the supporting system functions, this model employs the same functions as those of RBAC core. However, when creating a session or when setting a new role, special attention should be paid as to whether roles are automatically activated or should be activated in an explicit way. As in the other function categories, the review functions for RBAC entail the functions of RBAC core. In addition, however, the set of users that are attributed a specific role is defined along with the roles that inherited this specific role. The set of roles assigned to a given user is defined along with the roles that were inherited by this set of roles.

### 3.2.1.3 Constrained RBAC:

Constrained RBAC revolves around the separation of duties pertaining to an RBAC model. For instance, it is likely that the same person will not be allowed to have access to two different roles; therefore, separation of duties is enforced. Constraints can be applied to user and role functions. For instance, a limitation could be that a role has a maximum number of users. For constrained RBAC, two subcategories can be defined [17]:

- 1) static separation of duty relations and,
- 2) dynamic separation of duty relations.

The static separation of duty relations has to do with imposing constraints on the attribution of users to roles (Figure 20). An example can be the definition of non-connected user assignments linked to sets of roles. According to this model, the restrictions are posed on roles and in defining user assignment relations. A user cannot be assigned to two different roles. Figure 21 illustrates how different elements interact within this model.



**Figure 20: Role hierarchies for a project: a) Role hierarchy, b) Administrative Role Hierarchy and, c) Private and Scoped Roles**

Dynamic separation of duty relations aims to decrease the permissions that are assigned to a user. This model, defines constraints on the roles that can be activated within a user session. Permissions can be allowed only for a limited amount of time needed to accomplish a specific duty. This model enables the user to be assigned two different roles as long as these roles do not create conflict of interest when acted independently. Thus, this model imposes limitations on the roles of a user during a session. Figure 22 illustrates how different elements interact within this RBAC model.

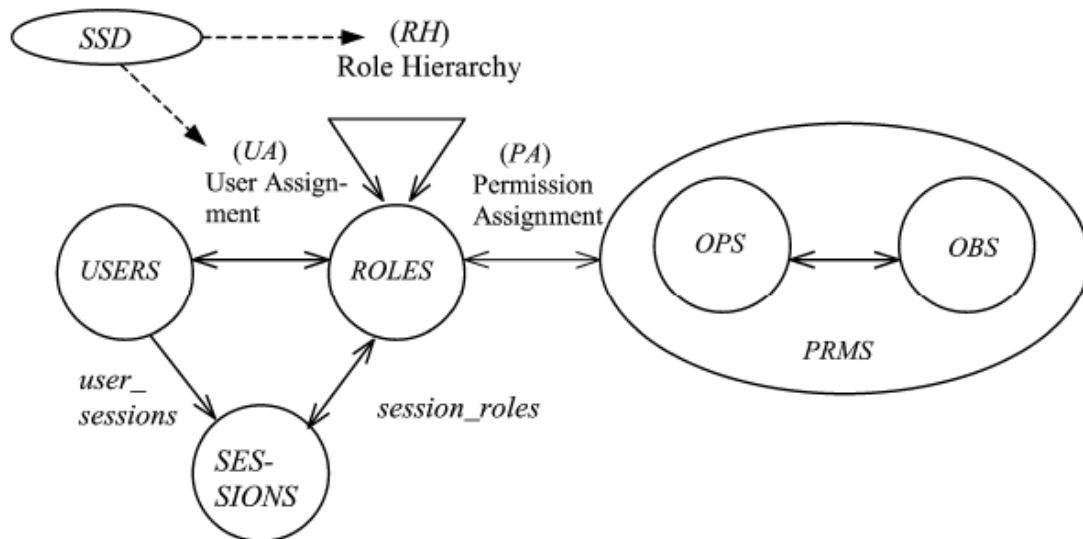


Figure 21: Static Separation of Duty relations, Constrained RBAC – elements and their interconnection.

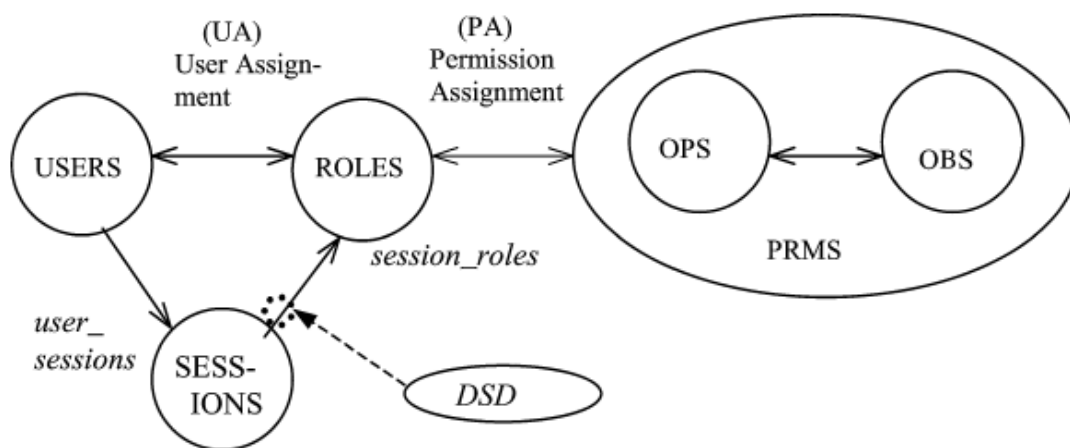


Figure 22: Dynamic Separation of Duty relations, Constrained RBAC – elements and their interconnection

### 3.3 Access Control Schemes: Considerations

Up to this point, we have presented a summary of the characteristics of the two main access control paradigms that are based either on the attributes of an entity (ABAC), or, its assigned roles (RBAC). Table 5 provides a summary of the advantages and disadvantages of each approach.

Table 5: RBAC vs ABAC characteristics

Characteristic	RBAC	ABAC
Flexibility	√ (for small and medium-sized organizations)	√
Scalability	-	√
Simplicity	Easy to establish roles and permissions for a small company, hard to maintain the system for a big	Hard to establish all the policies at the start, easy to



	company	maintain and support
Support for simple rules	√	√
Support for complex rules	√	√
Support for rules with dynamic parameters	-	√
Customizing user permissions	Customization entails the creation of new roles	√
Granularity	Low	High

Thus, from Table 5, we can deduce that the RBAC Access control policy does not suit the needs of DELTA. Indeed, RBAC is more suitable for small and medium scale organizations. Furthermore, its design process revolves around the notion of assigning the same access credentials to a group of users that share the same role. In DELTA's case, this would entail that, e.g., all FEIDs would have the same credentials and access to the same set of resources. Clearly, this would violate end-customer privacy and would not account for cases where, e.g., a FEID is uniquely assigned to a particular DVN.

The ABAC access policy also has its own limitations which, however, can be mitigated or even completely circumvented by carefully assigning attributes to entities/actors and by meticulously drafting the access control policies of the DELTA's exposed interfaces. Thus, DELTA will follow the ABAC approach. A concrete specification of the ABAC policies for all the interfaces exposed by the FEIDs, the DVNs and the DELTA Aggregator will be provided in D5.3. In this deliverable, we provide details in regards to access control revolving around DELTA's blockchain, in order to facilitate the works related to T5.2.

### 3.4 DELTA Blockchain: Access Control

The blockchain infrastructure is one of the most important subsystems in the DELTA platform as it facilitates the secure, fast and automatic deployment and settlement of various DR schemes between the Aggregator and customers. Strict access policy rules have to be in place for defining which subset of identities of the system can access each one of the distinct services provided by DELTA's blockchain network. The main entities interfacing with the DELTA blockchain are the Aggregator, the DVNs and the FEIDs. These entities are going to interact with smart contracts.

The DELTA blockchain will be developed on the Hyperledger Fabric platform (HLF)<sup>1</sup>. In HLF, networks are maintained and used by multiple organizations. Each organization participates in the system by providing at least one peer (blockchain node) and one CA that manages the identities of those related to the organization. Apart from the peers, an organization can allow its users to interact with smart contracts through client apps. In any case, every actor interacting with the blockchain infrastructure is required to be identified (through its organization's CA) in order to then apply specific predefined access rules regarding this interaction.

In HLF there is a basic notion, referred to as channel, which practically stands for a blockchain instance. In a single HLF network more than one channel may be concurrently active and each one of those is accessed by a defined subset of actors of the global network. In each one of these channels, one or more smart contracts are deployed. Smart contracts in an HLF environment implement the business logic of the applications to be served by the HLF network. In addition, HLF provides a set of system configuration smart contracts, which mainly define the rules for operating each individual channel, such as who can deploy smart contracts, who can commit transactions, or how those are validated. Specifically, there are multiple different system chaincodes:

<sup>1</sup> <https://www.hyperledger.org/projects/fabric>

- **Configuration system chaincode (CSCC):** Runs on all peers and handles changes to a channel's configuration, such as a policy update.
- **Query system chaincode (QSCC):** Runs on all peers and exposes ledger APIs which include block query, transaction query etc.
- **Endorsement system chaincode (ESCC):** Runs in endorsing peers to cryptographically sign a transaction response.
- **Validation system chaincode (VSCC):** Validates a transaction, including checking endorsement policy and read-write set versioning.

#### 3.4.1 Network Level Access Control

The main resources of a HLF network are (system) smart contracts and the event stream source, which may be triggered by smart contracts as their state changes. HLF applies explicit access control for all these upon two different policy configurations. **Signature Policies** identify specific users who must provide their signature in order for a policy to be satisfied. These policies can be composed by chaining logical predicates (AND, OR), or even arithmetic expressions, such as a minimum number of actors of a specific role. **Implicit Meta Policies** can combine simpler **Signature policies**. These policies use a simple syntax that takes as input simpler policies (e.g., <ALL|ANY|MAJORITY> <sub\_policy>)

According to the aforementioned policy syntax, a specific plan regarding the decision to allow to a specific actor to initiate a specific action can be set up. Specifically for DELTA, this level of access control could provide the basis to develop rules, such as:

- Only the Aggregator can deploy a smart contract related to a DR event that involves, e.g., the Aggregator and one or more DVNs.
- Only a DVN can instantiate a chaincode that relates to the settlement of a DR scheme between a DVN and a FEID.
- A transaction can only be validated if at least one Aggregator peer endorses it.

#### 3.4.2 Smart Contract Level Access Control

Apart from the access control defined in the previous section, an additional access control layer of higher granularity resides inside the smart contracts that will define the business logic of the applications. More specifically, when a smart contract is invoked, it is possible to check the identity of the caller and use this information to decide on the workflow to be followed. The **Client Identity Chaincode (CID)** library of HLF enables the retrieval of the identity of the invoking party. The client's identity can be checked with respect to its organization's Membership Service Provider (MSP), in addition to specific attributes of the client, which are embedded in its certificate and can also be retrieved and used to take access control decisions.

In DELTA's context, by employing the CID library, different access rules can be set up, such as:

- Only the FEIDs that have been defined in the smart contract by the DVN to take part into a DR scheme can invoke the function through which they accept to participate.
- Only the FEIDs that participate in a DR scheme can report corresponding measurement values to the FEID.
- Only the Aggregator can invoke the function that finalizes a DR scheme and adds bonus points to the end-user accounts.

As it has been illustrated, the DELTA blockchain implementation is going to employ two levels of access control (network level and smart contract level). As the design and implementation of blockchain functionality is currently pending, currently there is no concrete access control scheme in place. Explicit rules are going to be defined in order to decide at each point in the workflow if an identity is able or not to interact with the DELTA blockchain. These rules are going to make up a concrete DELTA blockchain access control policy that will be strongly coupled and compatible with the access control policy that has been described for the rest of the DELTA system. The blockchain access control policy, along with the technical mechanisms, that are going to be employed, in order to implement it, are going to be described in more detail in D5.2.

### 3.5 Privacy

---

Privacy is a fundamental right established by much domestic legislation in contemporary societies. The concept of privacy has altered over time, driven mainly by the impact of new technologies. Concerning privacy in the DELTA project, the goal is to protect the privacy of individual customers, as well as, the Aggregator. In the following, we illustrate our approach in guaranteeing privacy within the DELTA platform.

The project's requirements, as documented in WP1, mandate that each individual customer does not have access to data related to other customers. This includes data that may either directly or indirectly identify another customer, as well as, data pertaining to the measurements reported by FEIDs that are installed at the premises of customers. As defined in DELTA's architecture, FEIDs do not communicate with each other. Thus, there is no direct transmission of data between FEIDs that can be used to identify customers or violate their privacy. Furthermore, customers have severely limited access to the capabilities of their FEIDs, e.g., they do not have the means to change the FEIDs configuration, apart from schedules revolving to DR events. Moreover, FEIDs are equipped with various hardware security features that strengthen even more their security properties.

FEIDs exchange data and communicate with DVNs using DELTA's P2P network. The architecture of DELTA's P2P network emphasizes on privacy and anonymity, ensuring that the data exchanged is hidden from eavesdroppers and spoofers and that the customers' personal data is secured during transmission. We stress that the Aggregator has access to its customers' personal/private data through DVNs, because both sides have signed an appropriate contract, i.e., the customer has provided his consent with a real-world contractual agreement and, thus, privacy his is not violated. Although Aggregators have full access to customers' personal data, other entities, such as the DSO, the TSO or other BRPs, cannot access such information. In cases where such parties need to process customer data, the customer's consent will be requested and once and if it is provided, the requested data will be provided to these parties by the Aggregator's infrastructure.

Lastly, it is imperative that customers do not have access to data that are relevant to the Aggregator's operations. Recall that customers, via their FEIDs or their mobile app, do not, in any way, pull data from the services of the Aggregator, or even the DVNs to which they are assigned to, that may leak any information in regards to the internals of the Aggregator's operations. Thus, the privacy of the Aggregator's business logic is also preserved.

## 4. Auxiliary Security Infrastructures

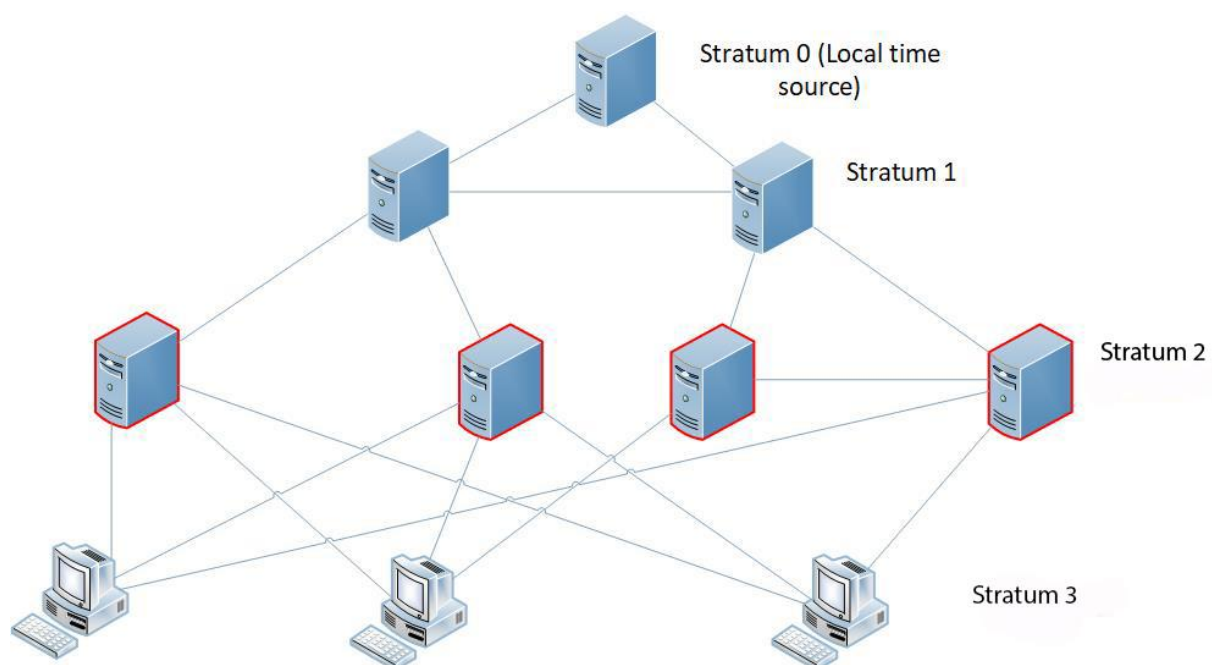
The time synchronization process is essential in the development of systems that involve the interaction of multiple components that are distributed over an asynchronous networking infrastructure, such as the Internet. Due to monetary costs or geographical problems (e.g., large distances), it is not always possible to adopt dedicated hardware-based solutions to guarantee clock synchronization. For such cases and others, timing protocols have been developed that manage to overcome the aforementioned limitations.

In DELTA, stemming from its DR-oriented functionalities, as well as, the non-negligible distance that separates the assets that are part of the Aggregator's portfolio, clock synchronization services are, not only relevant, but mandated to, e.g., guarantee the correctness of the certificates generated/revoked by the Aggregator's CA.

There are a couple of different temporal synchronization protocols where each has its own unique characteristics in terms of performance and, more importantly, security properties. Here, we analyze the most widely deployed temporal synchronization protocols, illustrate their induced safety risks and present our choice for DELTA.

### 4.1 NTP

The Network Time Protocol (NTP) is used to synchronize the clocks of components over the Internet. The first edition of the NTP protocol was published in RFC958 in 1988 [18], while it's the latest version, NTPV4, was updated in 2010 [19]. The NTP subnet model includes several widely accessible primary time servers, which are synchronized with each other. The NTP protocol conveys timekeeping information from these primary servers to secondary time servers and clients. NTP has a hierarchical structure divided into 16 levels or strata in which the first stratum, stratum zero, provides a high-precision time source. Clients can query a specific stratum depending on the timing precision required.



**Figure 23: An NTP Application Example with 4 Stratums**

An NTP primary server is synchronized to a reference clock directly related to the UTC time zone (Figure 23). A client synchronizes to one or more upstream servers, but does not provide

synchronization to dependent clients. A secondary server has one or more upstream servers and one or more downstream servers or clients. In order to maintain stability in large NTP subnets, secondary servers should be fully NTPv4-compliant.

#### **4.1.1 Security Considerations**

NTP has been extended multiple times to enhance its security properties. It supports both symmetric and asymmetric cryptographic primitives for authentication. However, support for these mechanisms varies highly as implementations do not provide these extensions in a consistent manner. An NTP client can claim to have a validly synchronized clock to dependent applications only if all servers on the path to the primary servers are properly authenticated. This implies that each NTP server authenticates to lower stratum server, a process which is assumed to be performed recursively.

We stress that NTP authentication does not imply that the conveyed timing information is correct. The NTP protocol specification takes into account attacks, such as false injection NTP packets, clogging of the network, Denial-of-Service (DoS), replay attacks and others. We provide a quick overview of the defense mechanisms suggested by the NTP specification.

The on-wire timestamp exchange scheme is inherently resistant to spoofing, packet-loss, and replay attacks. The engineered clock filter, selection and clustering algorithms are designed to defend against evil cliques of Byzantine actors. While not necessarily designed to defend against persistent intruders, these algorithms and their accompanying sanity checks, have functioned well over the years to. However, these mechanisms do not securely identify and authenticate servers to clients. Furthermore, NTP appears to be susceptible to wiretap-like attacks, interception attacks, NTP packet storage and resource saturation attacks. RFC 7384 [20] defines a set of security requirements for timing protocols, focusing on the PTP (Precision Time Protocol) and the NTP (Network Time Protocol). Furthermore, it discusses security impacts of timing protocols, the performance implications of external security practices and the dependencies between other security services and clock synchronization.

## **4.2 PTP**

PTP, which was originally defined the IEEE 1588 standard [21], is a packet-based technology that enables the operator to deliver synchronization services on packet-based mobile networks. Version 2 of the protocol was released on 2008 and provides a highly precise protocol for time synchronization that synchronizes clocks in a distributed system, which communicates through Ethernet networks. Depending on the requirements, PTP can not only perform time synchronization, but can also deliver phase and frequency synchronization among devices that are connected to the same time source. Time synchronization is achieved through packets that are transmitted and received in a session between a master and a slave clock.

PTP is an application layer protocol provides that employs UDP/IP as its transport layer. PTP operates in a hierarchical manner, where the node that has the most precise clock becomes the time source master. Hence, PTP, instead of establishing a rigid hierarchy of timing information propagation across its multiple layers, tends to have a more dynamic organization where dedicated operational processes elect the most reliable node as the time source master. PTP provides an algorithm that calculates the time difference between nodes and updates the clocks of slave nodes to that of the master node. The time synchronization mechanism works as follows:

1. Transmit are transmitted over the Ethernet network.
2. The transmission and receipt time are recorded.
3. Calculate the time difference.
4. Adjust the slave's clock to match that of the master.
5. Repeat this process periodically.

The system clocks can be categorized based on the role of the node in the network. They are broadly categorized into ordinary and boundary clocks. The master and slave clocks are ordinary clocks. The boundary clock can operate as either a master or a slave clock. In the following, we elaborate on the clock categories.

**Master clock** — The master clock transmits messages to the PTP clients. This allows the clients to establish their relative time distance and offset from the master clock, which acts as a reference point. Messages are delivered to the clients either via unicast, or multicast over Ethernet or UDP/IP.

**Slave clock** — The slave clock performs clock and time recovery operations based on the received and requested timestamps from the master clock.

**Boundary clock** — The boundary clock operates as a combination of master and slave clocks. The boundary clock endpoint acts as a slave clock to the master clock and as a master to all the slaves reporting to the boundary endpoint.

The temporal precision that can be obtained through PTP is closely tied to its implementation. Table 6 provides concise descriptions of the four different implementation of PTP, which range from pure software to pure hardware implementations.

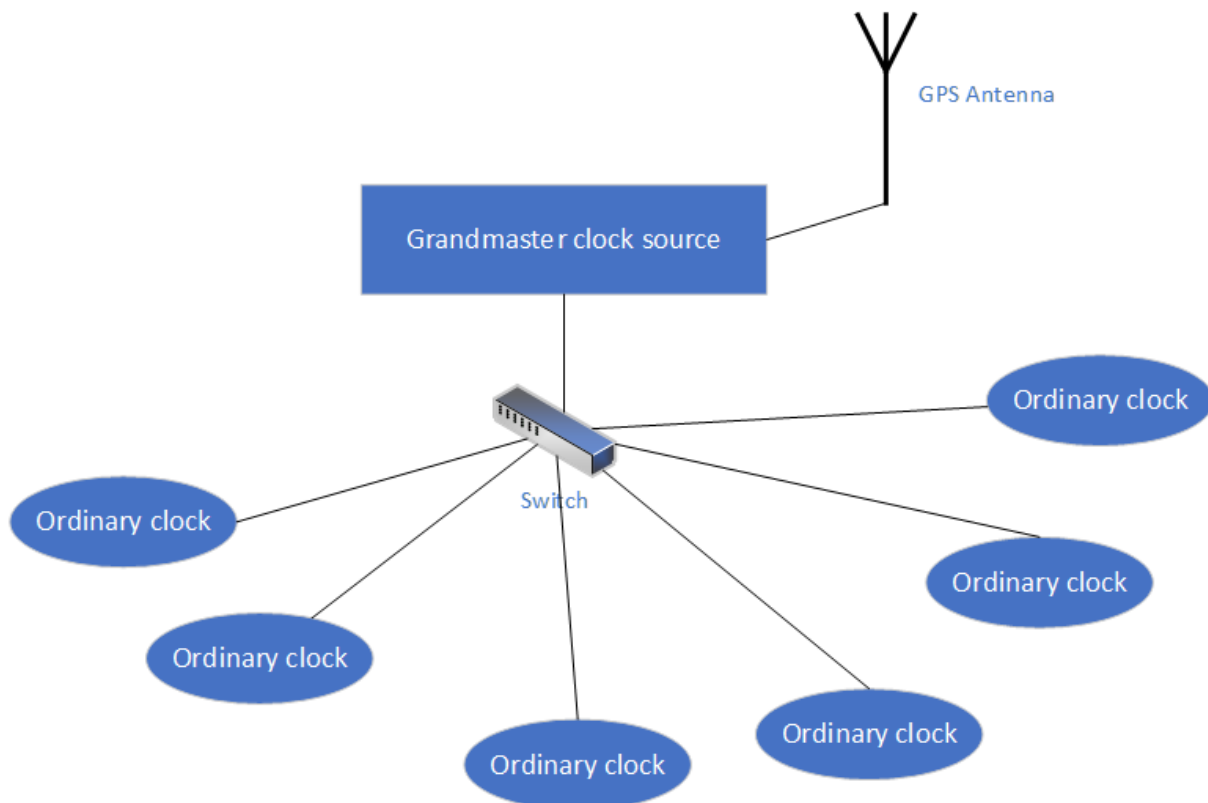
Approach	Development Consideration	Precision Performance
Software only	Requires software development	Lowest precision > 10 microseconds on a single link
Hardware assisted on FPGA	Requires significant hardware and software investments. FPGA development required	Typical > 30 nanoseconds on single link
Hardware assisted in Microcontroller	Requires changes to the microcontroller, Existing software application may need to be customized	Typical > 30 nanoseconds on single link
Hardware assisted in Ethernet PHY (available on DP83640)	Simple hardware implementation that involves software customization. Provide the tightest time synchronization available.	Typical < 30 nanoseconds on single link

**Table 6: Allowed PTP implementations and relative performances**

The PTP software stack resides at application layer. As a result, PTP packets can be affected by delays while traversing the hierarchy, i.e., from Ethernet PHY to the OS. Pure PTP software implementations provide the least precise timing synchronization, which is acceptable for applications that do not require accuracy in the order of nanoseconds (performance comparable to NTP).

In contrast, PTP configurations that require hardware components handle clock timestamps at the hardware level, which limits, to a minimum, the amount of incurred delays. A time signal source originates either from a local oscillator (e.g., rubidium gas oscillator), or from a trusted source that acts as leader in the network. The grandmaster clock, as illustrated in Figure 24, creates a master/slave hierarchy of clocks in the network, using a time signal that originates from a GPS antenna. All of the packets sent by the grandmaster clock traverse the switch and arrive at ordinary clocks.





**Figure 24: Simplified PTP communication/sync scheme**

#### **4.2.1 Security Considerations**

PTP was originally published in 2002 and it mainly focused on precise synchronization for instrumentation, industrial automation and military applications. The second version was finalized in 2008 [21] and introduced additional use cases, such as telecom and enterprise environments. While the first version of PTP contained no security mechanisms, the second version contained an experimental annex (Annex K). Annex K specifies a security solution that provides group source authentication, message integrity and replay attack protection. However, Annex K is neither well adopted nor implemented. Its security features can be summarized as follows:

1. Provides:
  - a. Group source authentication.
  - b. Message integrity.
  - c. Replay attack protection.
1. Does not provide:
  - a. Non-repudiation.
  - b. Data confidentiality.

PTP's secure implementation involves an integrity protection mechanism, which uses a message authentication code to verify that a received message was transmitted by an authenticated source, was not modified in transit and is fresh. Replay protection is implemented via a simple counter-based scheme. Furthermore, a challenge-response mechanism is used to affirm the authenticity of new sources and to maintain the freshness of trust relations.

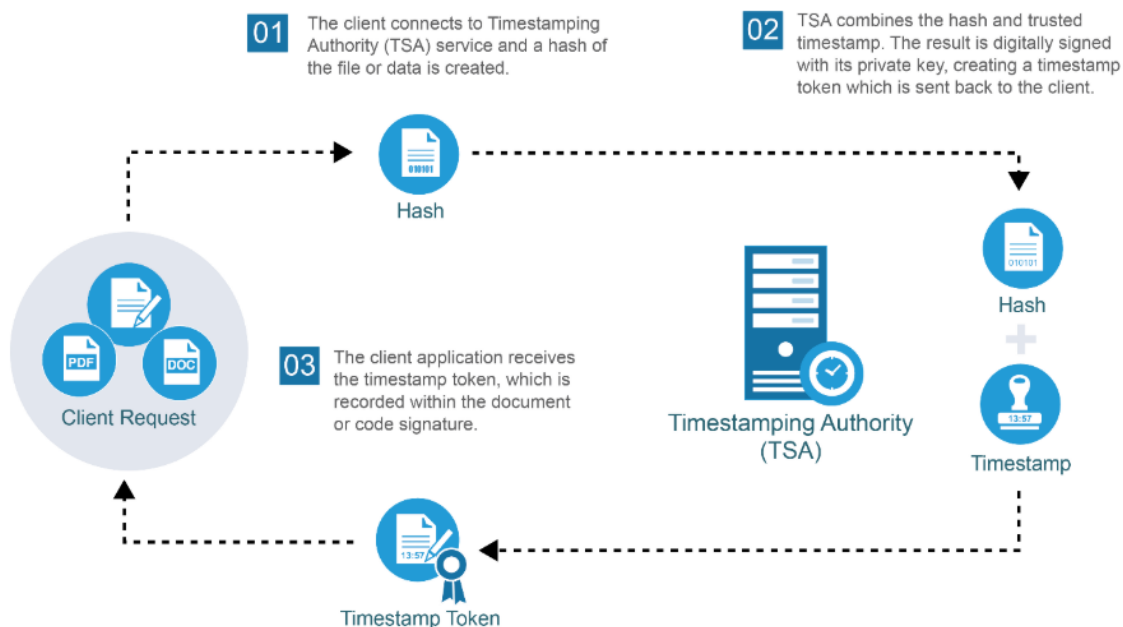
#### **4.3 RFC 3161 – Timestamp Authority**

A timestamp authority (TSA) is an entity whose main function is to associate arbitrary data with a particular time source. This trusted third-party certifies the existence of data at specific time points

while also certifying the authenticity of data [22]. TSAs employ PKIs and an overview of the timestamping procedure is as follows:

1. The client application creates a cryptographic hash of the data, which acts as a unique identifier of the data, and sends it to the TSA.
2. The TSA combines this hash with other relevant metadata information, such as the authoritative time. The result is digitally signed with the TSA's private key, creating a timestamp token which is sent back to the client. The timestamp token contains the information required by a client application to verify the timestamp of the data.
3. The timestamp token is received by the client application and appended to the actual data.

The client application uses the TSA's public key to authenticate the TSA (i.e., validate that the timestamp came from a trusted TSA) and re-calculate a hash of the original data. This new hash is compared to the originally created hash (Step 1). If any changes have been made to the data since the timestamp was generated, this hash check will fail, indicating that the data has been tampered with and, thus, it should not be trusted.



**Figure 25: TSA timestamping procedure**

The following requirements and assumptions are applicable in this setting regarding the TSA:

1. A trustworthy and accurate source of time is employed.
2. A unique integer is incorporated to each newly generated timestamp token.
3. A timestamp token is produced only upon receiving a valid request.
4. A unique identifier that indicates the security policy under which the token was generated is included in the token.
5. A collision resistant hash function is employed to generate a digest of the signed data.
6. The entity requesting the timestamp should not be identifiable by the timestamp token.

However, the standard does not specify the security requirements, nor the procedures required to ensure the safety of the TSA. RFC 3161 [22] does not mandate a specific transport mechanism for the transmission of TSA messages. Instead, it provides a description of the payload's structure for a variety transport protocols, which we enlist below:



1. SMTP.
2. FTP.
3. Socket-based protocol.
4. HTTP.

As we have already illustrated, timestamps in DELTA are necessary. However, developing and testing a timestamping service that requires dedicated hardware and software for its functionality is out of the project's scope. Typically, the TSA service is offered to third-party applicants as a service. However, in DELTA, we advise against third party services that may have detrimental impact on the platform's security.

## 5. DELTA Peer-to-Peer Network: Architecture & Security

A peer-to-peer (P2P) network allows the direct exchange of information between the nodes comprise the network using arbitrary data formats. According to their degree of centralization, P2P networks can be classified as follows:

- **Centralized;** All information exchange is done through a single, centralized server. This type of network is limited in terms of user privacy and scalability. The P2P server constitutes a single point of failure and has high bandwidth requirements.
- **Distributed;** This is the most common approach where nodes act both as clients and servers, thus, there is no central server that handles network connections. All involved communications are point-to-point between nodes and other nodes of the network offer their link to facilitate message relay. This approach is scalable, does not introduce a single point of failure and bandwidth requirements are distributed over all nodes of the network. The level of privacy that they offer depends on the peculiarities of the deployment.
- **Hybrid;** This approach, much like the centralized one, employs one or more centralized servers that, however, collectively manage network resources and, via standard redundancy techniques, do not introduce a single point of failure. Thus, the network is operational even in the presence of server failures. This approach combines the benefits of both decentralized and centralized networks.

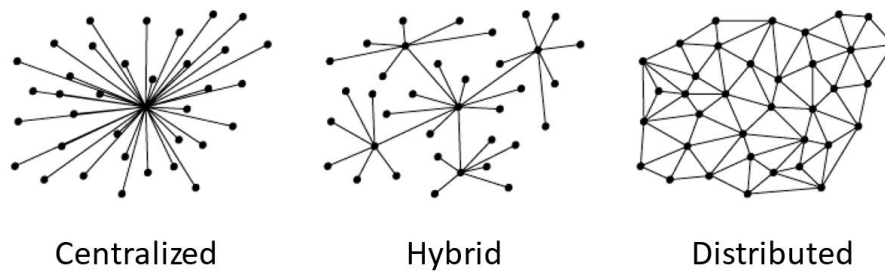


Figure 26: P2P network classification and node interconnections.

In the DELTA platform, a Hybrid P2P network has been implemented, since P2P is one of the OpenADR 2.0b requirements for the data exchange. The DELTA P2P network has the following features:

- **Scalability and decentralization;** in the P2P network, brokers (servers) and nodes (clients) can be added, as the system requires. This feature allows including servers that will behave as just one, i.e., transparently for the nodes, at the same time that the number of clients grows. The growth of the servers prevents the network to be flooded with the data exchange among clients, and at the same time, the possibility of adding more servers allows to include an undefined number of clients. Thanks to this scalability and decentralization, a single point of failure is avoided, and clustering allows the broker to scale with more brokers, so the information is distributed among all brokers.
- **Privacy;** to protect the privacy of the data that is stored on the broker, data needs to be encrypted. This feature allows customers' data not to be read by unauthorized devices. In the case of a computer attack, this information would remain secure, since by encrypting private data, a hacker would have more difficult access to such data.
- **Authentication;** to prevent and avoid possible malicious nodes, nodes cannot be anonymous. Establishing an identity verification makes it more difficult for an unauthorized person to access a device. With certificates, impersonation is even more difficult, so information theft will be avoided with this feature, helping to comply with security and privacy measures.

From a technological point of view, the DELTA platform uses an OpenFire [23] server as communication broker. This server keeps record of all the users that can join the network, and is the authority that enables or disables nodes to exchange data. The OpenFire server is not a centralised authority since, as we will explain later, it was designed to work with several deployments. In other words, several OpenFire brokers can be deployed and behave as just one providing fault tolerance mechanisms, scalability in the number of nodes that can join their P2P network, and decentralization in the deployment.

## 5.1 OpenFire

---

OpenFire is a cross-platform real-time collaboration server based on the XMPP protocol. This protocol is the leading open standard for real-time messaging, in fact it is one of the most used in real-world applications, for instance, it has been used for private chatrooms in gaming applications for platforms such as Origin, PlayStation, etc.

### 5.1.1 Configuration

The OpenFire server has to be configured with a XMPP domain name (each node in the cluster must have the same XMPP domain name), the server host domain (each node in the cluster must have a unique server host domain), the admin console port (http web-based admin console), the secure admin console port (SSL web-based admin console), the encryption type (blowfish or AES, blowfish is faster and AES is more secure) and a custom encryption key (for additional security).

The OpenFire server enables the login of clients in the network, as well as, some other features for the client. Allowing such features requires OpenFire to store data, for which two types of databases are available in the OpenFire configuration: an external database (MySQL, PostgreSQL, or Microsoft SQL server), or an embedded database. Regarding the profile settings, a directory server (LDAP) can be deployed for store users and groups or store them in the server database (allowing hashed passwords). Finally, in the last step of the configuration, we must configure an admin email address, which will be a username, and the password. This administrator user will have permissions to configure the entire system, add new users, add certificate authorities, add new administrators, etc.

### 5.1.2 Setup

In this section, a best-practices overview to define the optimal setup options for the DELTA project is provided, bearing in mind the security requirements that OpenADR establishes and others decided in the consortium.

#### 5.1.2.1 Scalability & decentralization

The scalability is one of the bottlenecks of the systems that accept new clients in their networks. With OpenFire this issue is solved by allowing one broker join a cluster of other brokers of the same nature. This cluster behaves as one for the clients, but leverages all the communications and data exchange by distributing the loads among them. For a broker to join an existing cluster, the domain name and subdomain must be placed in the install configuration. The communication process between server-to-server have two types of connections, the plain-text connection (using the STARTTLS protocol) port is the 5269, and the encrypted connections (as opposed to using STARTTLS) use the 5270 port. The client connections have two types of connections too, the plain-text connection (using the STARTTLS protocol) use port is the 5222, and the encrypted connections (as opposed to using STARTTLS) use the 5223 port.

The difference between the plain text connection and the encrypted connection is to use or not STARTTLS, which offers a way to improve a plain text connection to an encrypted connection instead of using a different port, so you can use any of the two connections

For decentralization, OpenFire have the function “Clustering”. This option can only be activated if a database is not embedded, since such database will established the grounded knowledge shared among all the brokers, and installing the plugin “Hazelcast Plugin”. Following an insight for the setup is provided:

### Clustering

Clustering allows the server to scale a lot more and at the same time avoid a single point of failure. Use the form to below to enable or disable clustering for this system. After disabling clustering this system will leave the cluster but the cluster will remain active with the remaining cluster nodes. When clustering is enabled this page will show information about the load each cluster node is having.

**Clustering Enabled**

☐ Disabled - This system is not running in a cluster.

☒ Enabled - This system is part of a cluster. **Note: enabling clustering may take up to 30 seconds.**

**Cluster Overview**

Below is an overview of your cluster. You have 1 node(s) running and you are licensed to 10,000 node(s) in this cluster. To see more information, click each node. The row in yellow indicates the local node.



Nodes	Joined	Clients	Incoming Servers	Outgoing Servers	Memory
 <b>desktop-hrt78c7 (192.168.1.142)</b> 5bb24a43-c3c1-43b4-83d6-3e34e91ac44c	Oct 16, 2019 1:28:16 AM	0 (0%)	0 (0%)	0 (0%)	 64,30 MB of 1979,75 MB used

Figure 27 OpenFire “Clustering” function

### 5.1.3 Privacy

In the connections client-to-client and server-to-server, the encryption protocol must be aligned with the OpenADR standard. This forces the DELTA OpenFire server require the use of TLSv1.2 or superior. OpenFire allows the use of several TLS, one of them being TLSv1.2, to configure that only this type of TLS is used, in the advanced configuration of Plain-text connections and encrypted connections, both in connections client-to-client and server-to-server, it must only permit TLSv1.2 or superior.

**Encryption Protocols**

These are all encryption protocols that this instance of Openfire supports. Those with a checked box are enabled, and can be used to establish an encrypted connection. Deselecting all values will cause a default to be restored.

☐ SSLv3

☐ TLSv1

☐ TLSv1.1

☒ TLSv1.2

When setting up a new encrypted connection some encryption protocols allow you to have part of the handshake (the 'hello') encapsulated in an SSLv2 format. The SSLv2Hello option below controls this encapsulation. When enabled, incoming data may use the SSLv2 handshake format (but SSLv2 itself will never be allowed). When disabled, all incoming data must conform to the SSLv3/TLSv1 handshake format. All outgoing data (which applies to outbound server-to-server connections) will always conform to the SSLv3/TLSv1 format irrespective of this setting.

☐ SSLv2Hello

Figure 28 OpenFire Encryption Protocols

For the encryption cipher suites, based on the requirements of OpenFire, the following options must be selected, depending on the system requirements:

- ECC - TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256
- RSA - TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256

#### 5.1.4 Authentication

The OpenFire allows the brokers to have control over who joins the P2P network. The brokers are the ones responsible of creating new credentials for those clients that are joining the P2p network. The anonymous connections are supported by OpenFire, nevertheless in the DELTA platform are discarded for two reasons, i.e., OpenADR does not support anonymous connections and the consortium of DELTA considers that it would turn in a weakness of the system not knowing who is creating traffic (bear in mind the nature of the data in DELTA).

Inband Account Registration
Inband account registration allows users to create accounts on the server automatically using most clients. It does not affect the ability to create new accounts through this web administration interface. Administrators may want to disable this option so users are required to register by other means (e.g. sending requests to the server administrator or through your own custom web interface).
<input checked="" type="radio"/> <b>Enabled</b> -Users can automatically create new accounts.
<input type="radio"/> <b>Disabled</b> - Users can not automatically create new accounts.

Change Password
You can choose whether users are allowed to change their password. Password changing is independent from inband account registration. However, you may only want to disable this feature when disabling inband account registration.
<input type="radio"/> <b>Enabled</b> - Users can change their password.
<input checked="" type="radio"/> <b>Disabled</b> - Users are not allowed to change their password.

Anonymous Login
You can choose to enable or disable anonymous user login. If it is enabled, anyone can connect to the server and create a new session. If it is disabled only users who have accounts will be able to connect.
<input type="radio"/> <b>Enabled</b> - Anyone may login to the server.
<input checked="" type="radio"/> <b>Disabled</b> - Only registered users may login.

**Figure 29 OpenFire Authentication Functions**

OpenFire uses the well-known jabber identification system. Users have a username, a name, an email and a password, and can be an administrator (grants admin access to OpenFire). The username and email are unique and help to allow the login to the system, with its corresponding password. All this, together with the name, can identify users in the system. Authentication in OpenFire can also be shielded by using certificates, which will be discussed in a future section.

#### 5.1.5 Fault Tolerance: XMPP Server Redundancy

Fault tolerance is the property that allows a system to continue to function properly in case of failure of one or more devices. In a hybrid P2P network, we will need more than one server to avoid failures, this problem is solved by adding more brokers (servers) what is called clustering. In addition, we find another point of failure, which is that OpenFire allows multiple logins with the same account, so this functionality conflicts with the operation of DELTA and must be corrected.

##### 5.1.5.1 Computer cluster

The term cluster is applied to a set of servers linked together and that behave as if they were a single server, controlled by specific software. The benefit of computer clustering is that it allows that overall system keeps working even when some specific node in the cluster fails. Other minor advantages is

that it allows to recovery data in the event of desynchronization, and provides parallel data processing and high processing capacity; which suits perfectly the decentralized scenario that DELTA brings.

## 5.2 Conflict policy

---

XMPP allows multiple logins to the same user account. If a connection makes a request for a resource that is already in use, the server must decide how to handle the conflict. OpenFire offers several options to handle these situations, like **always kick** if there is a resource conflict, **never kick** if there is a resource conflict, **allow one login attempt**, reporting an error but without kick the existing connection and **assigning kick value**, specifying the number of login attempts allowed.

## 5.3 Username Binding

---

OpenFire allows binding usernames with certificates to enhance the identification of requests in the network. Binding usernames in the P2P network requires valid certificates that have been signed by an authorized CA. To achieve this purpose the OpenFire server must be setup with the following options:

Property name	Property Value
xmpp.client.cert.policy	needed
xmpp.socket.ssl.certificate.accept-selfsigned	false
xmpp.socket.ssl.certificate.verify.validity	true
xmpp.socket.ssl.client.certificate.verify.validity	true
xmpp.socket.ssl.client.certificate.accept-selfsigned	false
xmpp.server.cert.policy	needed
xmpp.server.tls.policy	required
xmpp.socket.ssl.active	true

These options define that the connections are made securely using certificates, entailing that clients can trust the issuer of a request with such certificate attached, and establish a secured channel of communication. To ensure that the certificates have been legitimately granted, any of the OpenFire brokers may act as a certifying authority. The section to setup such feature is located in the TLS/SSL certificates section. Openfire uses the certificates in that list to verify the identity of remote clients and servers when encrypted connections are established.

**Certificate Stores**

These stores are used for all encrypted communication. Three stores are provided: one identity store, a trust store for server-based connections, and a trust store for client-based connections.

**Identity store**

File:

Password:

[Manage Store Contents](#)

**Trust store used for connections from other servers**

File:

Password:

[Manage Store Contents](#)

**Trust store used for connections from clients**

File:

Password:

[Manage Store Contents](#)

Figure 30: OpenFire Certificate Stores

#### Openfire Trust Certificate Store

✔ Certificate added or modified successfully.

Certificates in this list are used by Openfire to verify the identity of remote clients and servers when encrypted connections are being established. By default, Openfire ships with a number certificates from commonly trusted Certificate Authorities.

Certificates can be added to this list by using this [import form](#).

Organization (Alias)	Valid between	Algorithm	Delete
DELTA (delta)	Oct 16, 2019 - Oct 15, 2020	RSA	<input checked="" type="checkbox"/>

Figure 31: Open Fire Trust Certificate Store



## 6. Security-oriented aspects of FEID-related Use Cases

### 6.1 FEID

Based on the requirements that are documented in D1.1 (and in sequel revisions, i.e. D1.5) [24], FEIDs are actual devices which will be connected to smart meters to measure energy related data and through an intelligent lightweight toolkit they will compute real-time flexibility to provide them as input to the DVN. FEIDs provide aggregated metering from multiple IoT devices that are connected to customer assets, report issuance and interpretation of OpenADR-based DR request signals. In this subsection, an initial hardware security approach of the FEID will be given, presenting the TPM module. Furthermore, the FEID hardware installation will be described, together with the hardware setup and configuration. Finally, the process of registration of different FEID components will be analyzed. The image below (Figure 32) depicts a top view of the FEID board, which contains a number of interfaces, which will be mentioned below.

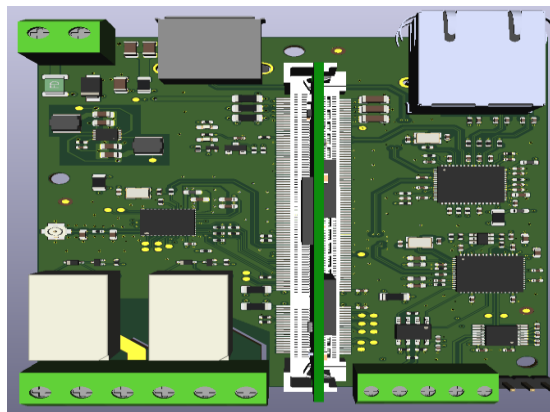


Figure 32: FEID board top view

Although all aspects around FEIDs will be thoroughly presented in the dedicated deliverable D3.4 - Fog-enabled Intelligent Devices, some high level information regarding hardware security will be presented. FEID will integrate on its board a chip called TPM (Trusted Platform Module) which was developed by the Trusted Computing Group, a group of computer industries and was standardized as ISO / IEC 11889 [25] in 2009 by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). The last revision of the TPM was performed in the first days of 2018, releasing TPM version 2.0 [26].

The latest TPM version includes attributes in order to support symmetric encryption in the platform the module is used and also provides the generation of high quality random numbers. Moreover, the module delivers a protected persistent store for small amounts of data, sticky bits, monotonic counters and extendable registers as well as an extensive choice of authorization methods to access protected keys and data. Except for the former characteristics of the TPM, it also allows the use of signing and verifying digital signatures and certifies the properties of the keys and data.

TPM v2.0 was created by TCG as a library in order to enable users to select the appropriate TPM design aspects for different levels of implementation and security. Furthermore, new features and functions have been introduced, to allow the implementation of new cryptographic algorithms when necessary. Detailed information will be presented in the corresponding deliverable regarding the FEID system. The figure below shows the differences of the two versions of the TPM (v1.2 and v2.0).



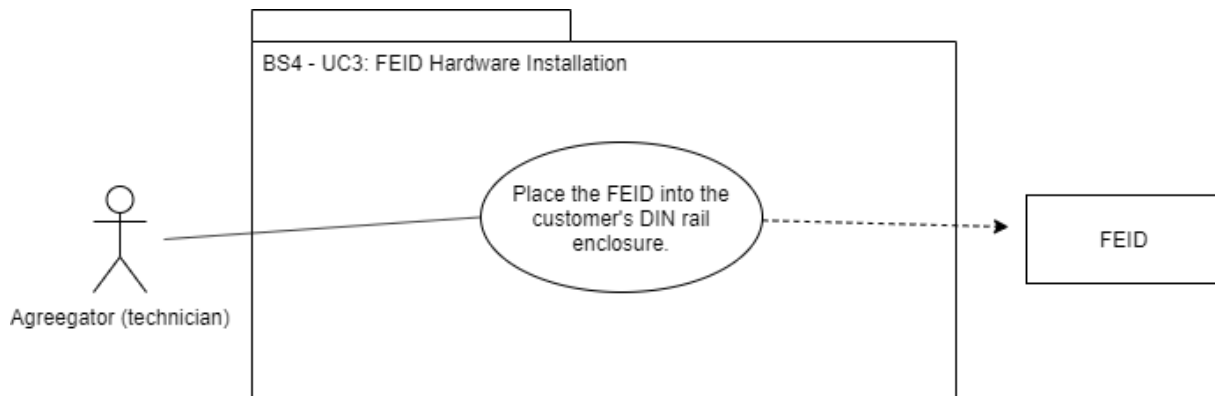
	TPM 1.2	TPM 2.0
Algorithms	SHA-1, RSA	Agile (such as SHA-1, SHA-256, RSA and Elliptic curve cryptography P256)
Crypto Primitives	RNG, SHA-1	RNG, RSA, SHA-1, SHA-256
Hierarchy	One (storage)	Three (platform, storage and endorsement)
Root Keys	One (SRK RSA-2048)	Multiple keys and algorithms per hierarchy
Authorization	HMAC, PCR, locality, physical presence	Password, HMAC, and policy (which covers HMAC, PRC, locality, and physical presence).
NV RAM	Unstructured data	Unstructured data, Counter, Bitmap, Extend

Figure 33: Differences between the two TPM versions

In the following, we present the description of use cases which are related to the management of FEIDs, from a security point of view, in DELTA.

### 6.1.1 FEID Hardware Installation

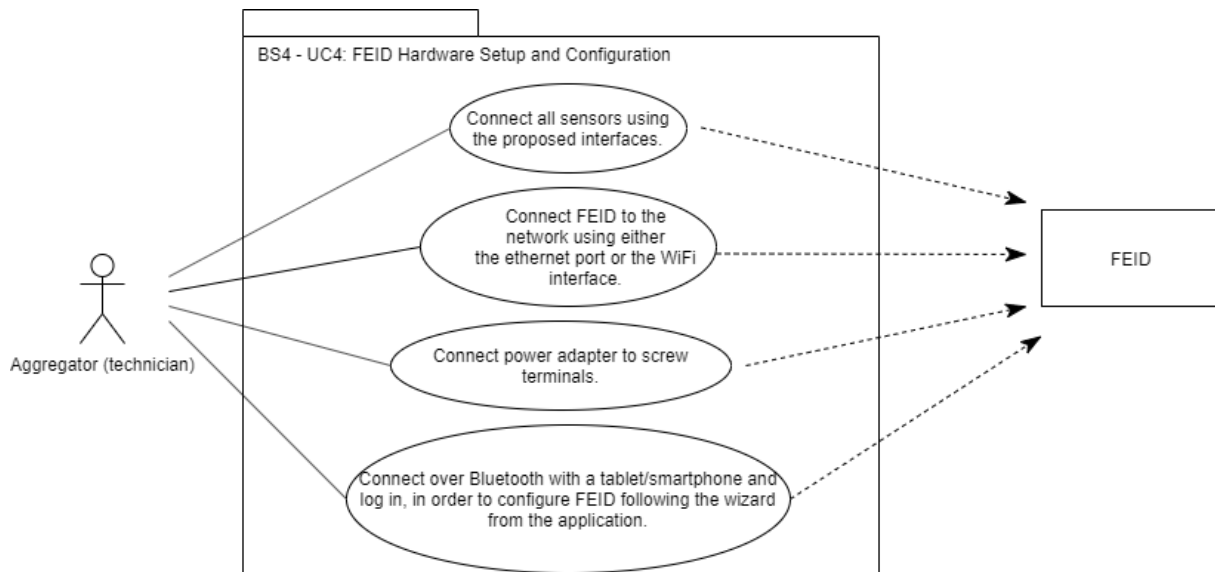
Use case name	BS4 – UC3: FEID Hardware Installation
Authors	CERTH/ITI
Brief Description	Installation of the Fog Enabled Intelligent Device in the customer's premises.
Assumptions & Preconditions	Connect FEID to smart technologies and/or to BMS or to power lines and sensors within the hosting infrastructure.
Objective	To install the FEID hardware in the customer's premises by an authorized technician.
Effects/Post Conditions	The Aggregator can ensure the proper installation of the FEID hardware and its successful inclusion in the platform.
Involved Actors	Technician (Aggregator), Prosumer, FEID
Use Case Initiation	The Aggregator and the customer agree on the most suitable time for both involved entities to install the hardware at the customer's premises.
Main course	<ul style="list-style-type: none"> <li>The technician will mount the FEID inside the customer's DIN rail enclosure, occupying 4 positions.</li> </ul>
Alternate course	-
Relationships with other Use Cases	-
Architectural Elements/Services Involved	FEID



**Figure 34: High-level Use Case Diagram**

### 6.1.2 FEID Hardware Setup and Configuration

Use case name	BS4 – UC4: FEID Hardware Setup and Configuration
Authors	CERTH/ITI
Brief Description	Setup and configuration of the Fog Enabled Intelligent Device in the customer's premises by the technician.
Assumptions & Preconditions	Wire up the FEID with the power adapter, connect it to power lines and sensors within the hosting infrastructure and configure it using an application.
Objective	To setup and configure the FEID hardware in the customer's premises by an authorized technician.
Effects/Post Conditions	The Aggregator can ensure the proper operation of the FEID hardware.
Involved Actors	Technician (Aggregator), Prosumer, FEID
Use Case Initiation	The Aggregator and the customer agree on the most suitable time for both involved entities to setup and configure the hardware at the customer's premises.
Main course	<ul style="list-style-type: none"> <li>• Connect all sensors to FEID using one of the interfaces (RS-232, RS-485, I<sup>2</sup>C, SPI, UART, Bluetooth v5.0).</li> <li>• Connect FEID to the network using either the Ethernet port or the Wi-Fi interface.</li> <li>• Connect the power cables to the two screw terminals and power up the FEID.</li> <li>• Connect over Bluetooth to the FEID to setup the device with a smartphone/tablet.</li> <li>• Log in with the credentials.</li> <li>• Follow the application's wizard in order to setup the device.</li> </ul>
Alternate course	-
Relationships with other Use Cases	BS4 – UC3
Architectural Elements/Services Involved	FEID

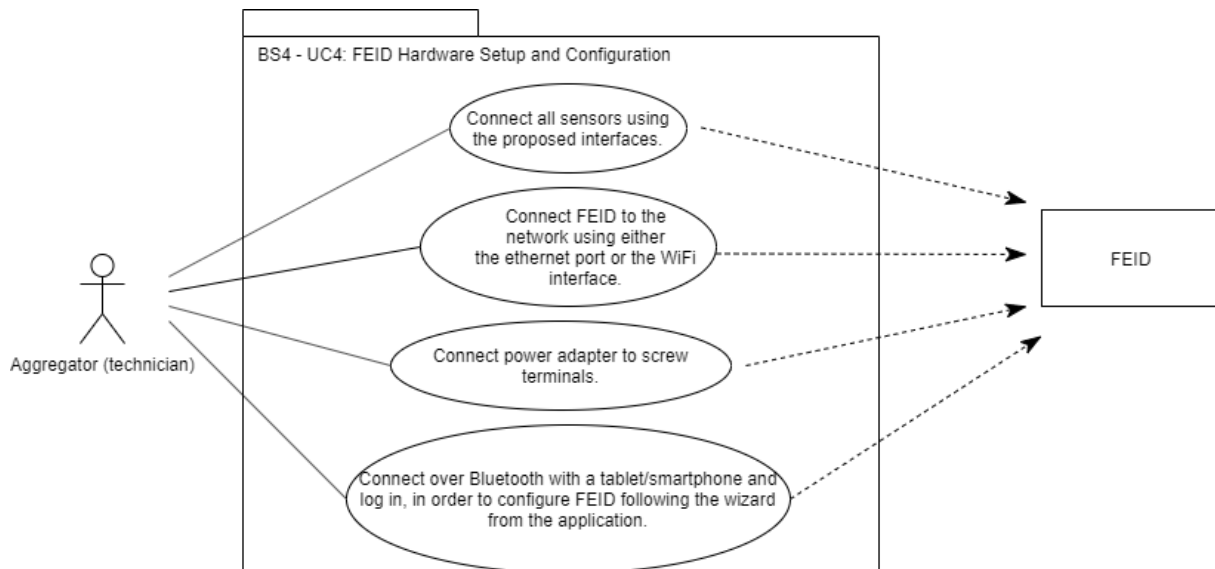


**Figure 35: High-level Use Case Diagram**

### 6.1.3 FEID Component Registration

Use case name	BS4 – UC4: FEID Hardware Setup and Configuration
Authors	CERTH/ITI
Brief Description	Setup and configuration of the Fog Enabled Intelligent Device in the customer's premises by the technician.
Assumptions & Preconditions	Wire up the FEID with the power adapter, connect it to power lines and sensors within the hosting infrastructure and configure it using an application.
Objective	To setup and configure the FEID hardware in the customer's premises by an authorized technician.
Effects/Post Conditions	The Aggregator can ensure the proper operation of the FEID hardware.
Involved Actors	Technician (Aggregator), Prosumer, FEID
Use Case Initiation	The Aggregator and the customer agree on the most suitable time for both involved entities to setup and configure the hardware at the customer's premises.
Main course	<ul style="list-style-type: none"> <li>• Connect all sensors to FEID using one of the interfaces (RS-232, RS-485, I<sup>2</sup>C, SPI, UART, Bluetooth v5.0).</li> <li>• Connect FEID to the network using either the Ethernet port or the Wi-Fi__33 interface.</li> <li>• Connect the power cables to the two screw terminals and power up the FEID.</li> <li>• Connect over Bluetooth to the FEID to setup the device with a smartphone/tablet.</li> <li>• Log in with the credentials.</li> <li>• Follow the application's wizard in order to</li> </ul>

	setup the device.
Alternate course	-
Relationships with other Use Cases	BS4 – UC3
Architectural Elements/Services Involved	FEID

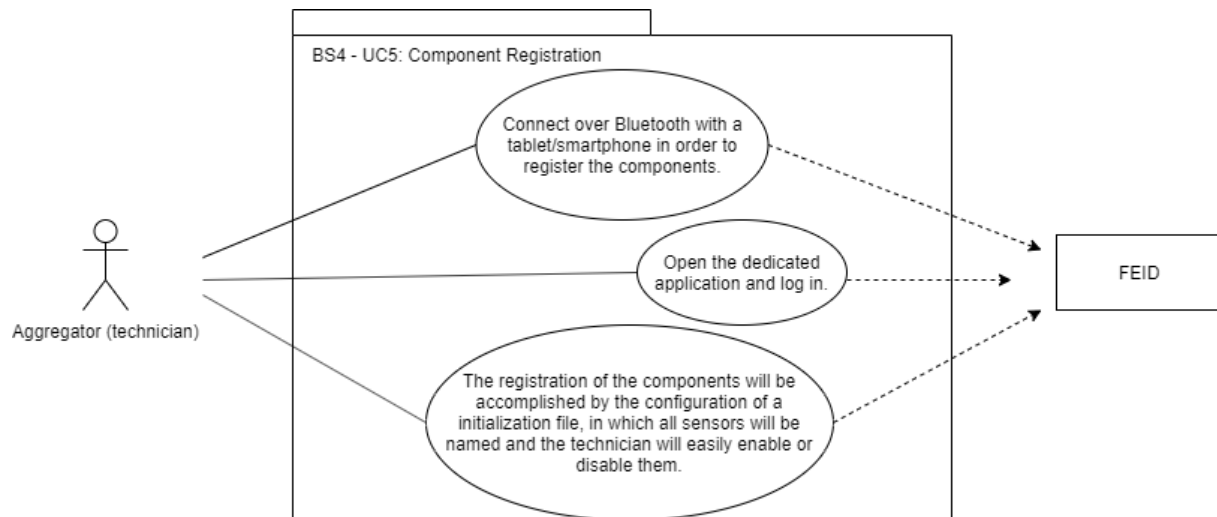


**Figure 36: High-level Use Case Diagram**

#### 6.1.4 FEID Component Registration

Use case name	BS5 – UC5: FEID Component Registration
Authors	CERTH/ITI
Brief Description	Register the components connected to the FEID in a customer's premises and register the FEID to the DVN by the technician.
Assumptions & Preconditions	Connect to the FEID via Bluetooth using a smartphone/tablet, log in with the necessary credentials in order to be able to configure and register the components to the system.
Objective	To register the sensors connected to the FEID hardware in a customer's premises by an authorized technician.
Effects/Post Conditions	The Aggregator can ensure the proper operation of the FEID hardware.
Involved Actors	Technician (Aggregator), Prosumer, FEID
Use Case Initiation	The Aggregator and the customer to agree on the most suitable time for both involved entities to register all components at the customer's premises.
Main course	<ul style="list-style-type: none"> <li>Connect over Bluetooth to the FEID using a smartphone/tablet.</li> <li>Open the dedicated application.</li> <li>Log in with the credentials.</li> <li>Configure the file that contains the names of all sensors to be registered to FEID.</li> </ul>

	<ul style="list-style-type: none"> <li>• Save the configured file.</li> </ul>
Alternate course	-
Relationships with other Use Cases	BS4 – UC3, BS4 – UC4
Architectural Elements/Services Involved	FEID



**Figure 37: High-level Use Case Diagram**

## 7. Conclusions

This deliverable presents all the required details and specifications that are involved in the design and implementation of an end-to-end secure, privacy preserving framework for all the actors involved in DELTA's ecosystem. A concrete documentation of the processes revolving around the management of digital identities is presented, as well as, the necessary adaptations required to support features of OpenADR 2.0b. Based on DELTA's identity services, we survey relevant and mature standards for access control and discuss their appropriateness in the project's context. We present our employed approach and introduce preliminary access control policies revolving one of DELTA's cybersecurity infrastructure services, i.e., DELTA's blockchain, pertaining to both the network, as well as, the smart contract level. These access control policies form the necessary basis to guarantee end-to-end privacy in DELTA. Furthermore, we introduce additional security infrastructures that resolve the issues that stem from the asynchronous nature of the underlying network, i.e., the Internet, which, if left unchecked, can have severe consequences, both in terms of security, as well as, privacy. To address the communication across the multiple layers that comprise DELTA's architecture, we provide concise details related to DELTA's P2P network architecture, security and configuration. Lastly, we conclude by presenting additional, security-oriented use cases pertaining to DELTA's FEIDs that are a result of the works involved in drafting this deliverable. The combination of all this information covers all aspects of DELTA's security.

## References

- [1] <https://tools.ietf.org/html/rfc5280>
- [2] <https://tools.ietf.org/html/rfc6187>
- [3] <https://tools.ietf.org/html/rfc2315>
- [4] <https://tools.ietf.org/html/rfc7292>
- [5] <https://www.oasis-open.org/committees/>
- [6] <https://www.sciencedirect.com/topics/computer-science/attribute-based-access-control>
- [7] <https://www.avatier.com/products/identity-management/resources/gartner-iam-2020-predictions/>
- [8] <https://www.axiomatics.com/blog/intro-to-attribute-based-access-control-abac/>
- [9] [https://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=xacml](https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml)
- [10] <https://medium.com/box-tech-blog/allowing-ada-to-view-her-files-boxs-attribute-based-access-control-solution-d83250216c1>
- [11] <https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf>
- [12] <https://auth0.com/docs/authorization/concepts/rbac>
- [13] Soni, K., & Kumar, S. (2019, February). Comparison of RBAC and ABAC Security Models for Private Cloud. In 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon) (pp. 584-587). IEEE.
- [14] <https://csrc.nist.gov/publications/detail/sp/800-178/final>
- [15] <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-178.pdf>
- [16] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, "Role-Based Access Control Models", *IEEE Computer*, vol. 29, no. 2, p. 38 – 47, Feb 1996.
- [17] D. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control", *ACM Transactions on Information and System Security*, vol. 4, no. 3, p. 224 – 274, Aug 2001.
- [18] <https://tools.ietf.org/html/rfc958>
- [19] Network Time Protocol Version 4: Protocol and Algorithms Specification, <https://tools.ietf.org/html/rfc5905#page-5>
- [20] <https://tools.ietf.org/html/rfc7384>
- [21] IEEE 1588-2008, IEEE Instrumentation and Measurement Society. TC9 Sensor Technology, "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems", 2008.
- [22] <https://tools.ietf.org/html/rfc3161>
- [23] OpenFire <https://www.igniterealtime.org/projects/openfire/>
- [24] D1.1 DELTA Requirements, Business Scenarios and Use Cases, March 2019
- [25] <https://www.iso.org/standard/50970.html>
- [26] W. Arthur, D. Challener, A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security Apress, 2015.