Ecosystem for COllaborative Manufacturing PrOceSses – Intra- and Interfactory Integration and AutomaTION
(Grant Agreement No 723145)

D4.2 Design of Security Framework II

Date: 2018-02-27

Version 1.0

Published by the COMPOSITION Consortium

Dissemination Level: Public

Co-funded by the European Union's Horizon 2020 Framework Programme for Research and Innovation under Grant Agreement No 723145

Document control page

Document file: D4.2 Security Framework II-V010-FINAL.docx
Document version: 1.0
Document owner: ATOS

Work package: WP4 – Secure Data Management and Exchange in Manufacturing
Task: T4.1 – Security by design for cloud-based data exchange
T4.3 – Knowledge Protection, IPR Protection and Trust for Collaborative Manufacturing Environments
T4.4 – Cyber Security for Factories
Deliverable type: R

Document status: ☒ Approved by the document owner for internal review
☒ Approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Javier Romero (ATOS)	2018-02-16	Deliverable structure
0.2	Javier Romero (ATOS)	2018-02-21	Main content
0.3	Mario Faiella (ATOS)	2018-02-21	Reputation Model content
0.4	Javier Romero (ATOS)	2018-02-22	Integration
0.5	Javier Romero (ATOS)	2018-02-22	Final changes and updates
0.6	Javier Romero (ATOS)	2018-02-22	Version for internal review
0.9	Javier Romero (ATOS)	2018-02-26	Addressed internal review comments
1.0	Javier Romero (ATOS)	2018-02-27	Final version

Internal review history:

Reviewed by	Date	Summary of comments
Vagia Rousopoulou (CERTH)	2018-02-23	Approved with comments: The content of document is comprehensive. Correct template has been used. The structure is good and mandatory sections are included. Most of D4.1 next steps have been accomplished. The template references style should be used. The language used could be more formal. The tables could be styled as shown in the template.
Ifigeneia Metaxa (ATL)	2018-02-23	Approved with comments: Minor typos, please take care of subscripts and superscripts in equations and explanations of symbols. Address comments within the document. Good structure and quality, level of details allows the reader to understand the approach in COMPOSITION. Might be valuable to consider a paragraph in the introduction on why this is necessary from the user's point of view.

Index:

Abbreviations

Acronym	Meaning
AMQP	Advanced Message Queuing Protocol
DRPC	Distributed Remote Procedure Call
DSS	Decision Support System
EPL	Event Processing Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPR	Intellectual Property Rights
JSON	JavaScript Object Notation
JWS	JSON Web Signature
JWT	JSON Web Token
MB	Message Broker
OAuth	Open Authorization
OIDC	Open ID Connect
OSSIM	Open Source Security Information Management
PAP	Policy Administration Point
PEP	Policy Enforcement Point
PDP	Policy Decision Point
PIP	Policy Information Point
PRP	Policy Retrieval Point
REST	Representational State Transfer
SAML	Security Assertion Mark-up Language
SHA	Secure Hashing Algorithms
SIEM	Security Information and Event Management
SPI	Service Provider Interface
SQL	Structured Query Language
SSL	Secure Sockets Layer
STIX	Structured Threat Information eXpression
TLS	Transport Layer Security
UDP	User Datagram Protocol
XACML	eXtensible Access Control Mark-up Language
XML	eXtensible Markup Language

1 Executive Summary

The aim of this deliverable is to update and complement what was reported on D4.1 Design of Security framework I due on M12. This deliverable takes as starting point the information contained in the aforementioned deliverable D4.1.

This deliverable reports the outcome of the following tasks: T4.1 – Security by design for cloud-based data exchange, T4.3 – Knowledge Protection, IPR Protection and Trust for Collaborative Manufacturing Environments and T4.4 – Cyber Security for Factories from M12 until M18. The purpose of these tasks is to define, propose a design and develop a core set of security measures that will be part of the COMPOSITION Security Framework, whose task will be to guarantee security, confidentiality, integrity and availability of managed information for all authorized stakeholders in the supply chain.

Some of the components and technologies reported in this deliverable ensure trusted and secure collaboration; and at the same time they guarantee confidentiality and integrity of the information transmitted by addressing end-to-end security across all layers of the system integrating in a seamless manner three major groups of security mechanisms: authentication, access control and transport security. Other components ensure protection against cyber-attacks and provide security monitoring.

The architecture is based on well established guidelines and best practices, as well as proven technologies; but also includes innovative and experimental solutions that will guard the COMPOSITION system against unknown threats.

The first prototype of the COMPOSITION Security Framework will be based on the architecture, components and technologies proposed on this deliverable and will be reported in D4.4 Prototype of the Security Framework I due on M20.

2 Introduction

Deliverable D4.2 Design of Security Framework II reports the results in the context of tasks T4.1 – Security by design for cloud-based data exchange, T4.3 – Knowledge Protection, IPR Protection and Trust for Collaborative Manufacturing Environments and T4.4 – Cyber Security for Factories from M12 until M18. It updates and complements the results reported on D4.1 Design of Security Framework I due on M12.

This deliverable describes the architecture design of the COMPOSITION Security Framework as well as the components and technologies that are part of it. It also reports on the developments that have taken place in this period of time. Some descriptions of components that make use of the blockchain technology are also given, although this subject is out of the scope of this deliverable and will be reported on D4.3 The Composition Blockchain due on M30. The proposal for a Reputation Model to be implemented in the COMPOSITION platform it is also provided.

This deliverable gives detailed information on some technologies proposed to be part of the COMPOSITION Security Framework, as these technologies will be an indispensable part of the platform and of mandatory use by most COMPOSITION components to be able to offer a high level of security, integrity of data and trust to the users of COMPOSITION platform.

2.1 Purpose, context and scope of this deliverable

The purpose of this deliverable is to update the proposal done on D4.1 Design of Security Framework I due on M12 for a design of a security framework that will ensure trusted and secure cooperation providing protection and monitoring against cyber-attacks. The set of components proposed are based on the following needs and requirements:

- Well-established authentication mechanism along with a multi-stakeholder attribute based access control mechanism. This combination should provide fine-grained access control to the data, based on a security token included within a submitted request and the evaluation of security policies.

- Guarantee the confidentiality and integrity of data in motion with the use of cryptographic mechanisms at transport layer

- Ensure the security monitoring and protection against potential threats identified in collaborative manufacturing and logistics ecosystems

2.2 Content and structure of this deliverable

This deliverable is composed of the following sections:

Section 2 - Introduction: serves as introduction and identifies the purpose, scope and context of this deliverable.

Section 3 - Security Framework Architecture: focuses on the architecture general overview as well as going in detail with different alternatives to the default architecture for some components. It also gives an overview of the components that take part in the architecture.

Section 4 - Security Framework Components – Configuration, Development, Integration: provides a view on the work done related to the components of the framework regarding to the development, integration, deployment and customization of components.

Section 5 - Integrity and trust of information: focuses on technologies proposed to bring integrity and trust on information to COMPOSITION

Section 6 - Transport security: provides information on the technology used to secure communication in COMPOSITION platform

Section 7 - Next Steps: provides an overview of the future work.

Section 8 - Summary: offers an overview of all reported in this deliverable

3 Security Framework Architecture

The purpose of the COMPOSITION Security Framework will be to guarantee security, confidentiality, integrity and availability of managed information for all authorized stakeholders within COMPOSITION platform. In Section 3.1, a brief description is given of each of the components that make up the Security Framework.

The following diagram shown in Figure 1 presents a very general overview of the COMPOSITION Security Framework and briefly describes the interactions with some other components in the COMPOSITION platform architecture.

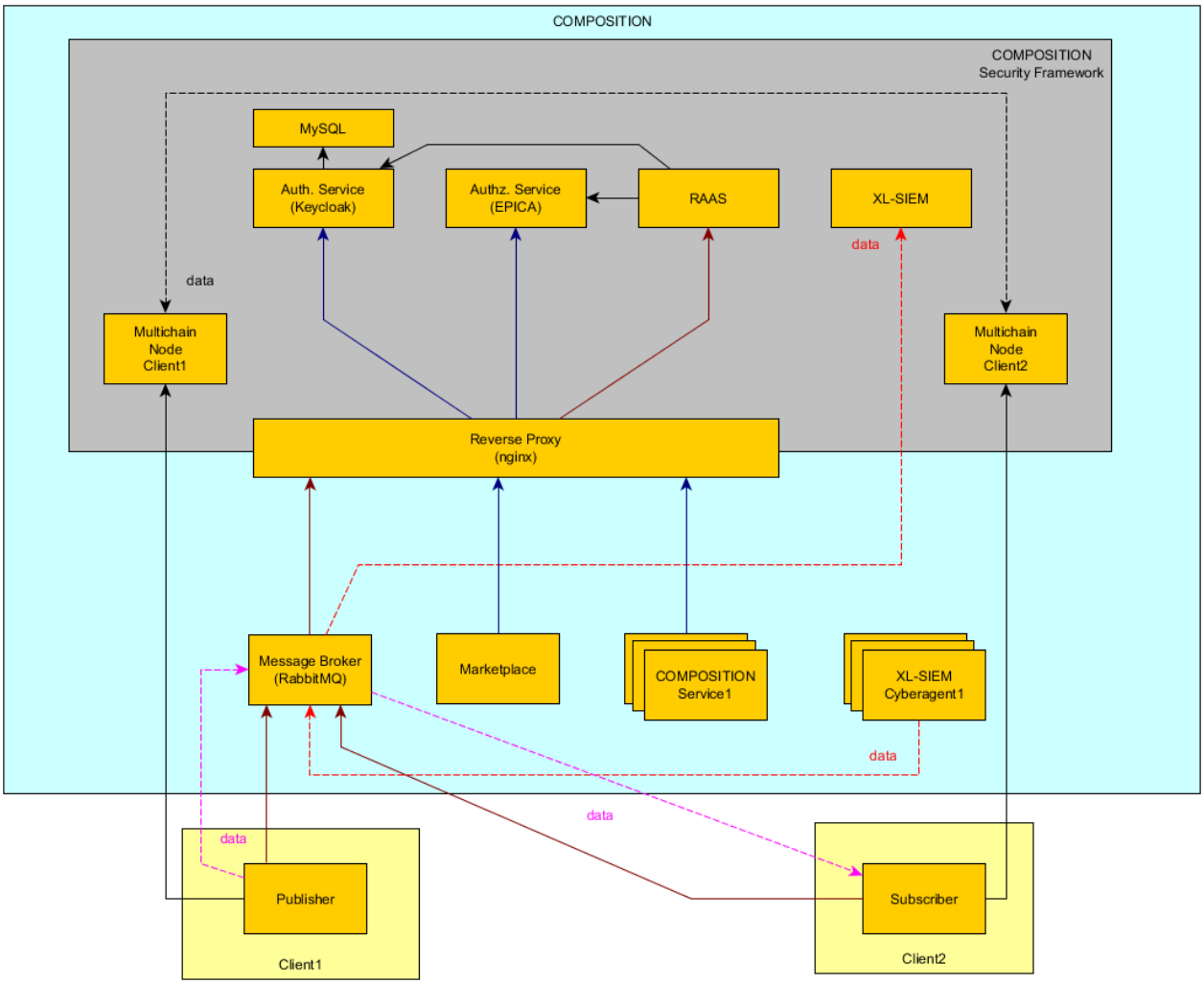

Figure 1 - Security Framework general architecture overview

COMPOSITION Security Framework will be composed of the following components to cover Inter-Factory and Intra-Factory scenarios:

- One Authentication Service (Keycloak[1])
- One Authorization Service (EPICA)
- Two RAAS services: One for Intra-Factory scenarios and one for Inter-Factory scenarios
- One XL-SIEM
- One or more cyber-agents
- Multiple blockchain nodes

[1] http://www.keycloak.org/

Due to the versatility on the platform proposed and focusing mainly on the deployment of RAAS services Section 3.2 will cover two possible deployments for these components and the interaction with the COMPOSITION Message Brokers (RabbitMQ) as well as with the rest of the Security Framework components involved.

3.1 Security Framework Components

COMPOSITION Security Framework consist of the following main components, each with a task to fulfil: an Authentication service (Keycloak), an Authorization service (EPICA), an Authentication and Authorization service for COMPOSITION message broker (RAAS), XL-SIEM which is a Security Information and Event Management system (SIEM) with additional functionalities and a Reverse proxy (Nginx). Each component is briefly described below and a more detailed description of each of them can be found in D4.1 Security Framework I due on M12.

3.1.1 Authentication service – Keycloak

The main task of this service is providing the authentication mechanisms for users, applications, services and devices. The following standard authentication protocols are supported by Keycloak:

- OAuth 2.0: Industry-standard protocol for authorization. Makes heavy use of the JSON Web Token (JWT) set of standards.
- Open ID Connect (OIDC): Authentication protocol based on OAuth 2.0. Unlike OAuth 2.0 OIDC is an authentication and authorization protocol.
- SAML 2.0: Authentication protocol similar to OIDC. It relies on the exchange of XML documents between the authentication server and the application.

From the available authentication protocols described above COMPOSITION makes use of the default one in Keycloak, which is OIDC (Open ID Connect). [1]

Custom mapper is in development to extend Keycloak´s capabilities, by enabling the possibility to add custom external information to the tokens provided by Keycloak. More information on this topic on Section 4.1

For more detailed information related to this component on Section 4.1 of D4.1 Design of the Security Framework I due on M12

3.1.2 Authorization service – EPICA

This component is responsible for providing authorization mechanisms to other COMPOSITION components. It is based on XACML v3.0 which provides an attribute-based access control mechanism and provides the means to define authorization policies used to protect resources. Any request to access a protected resource will first be evaluated against the defined policies and the evaluation result will be enforced depending on the outcome. EPICA is divided into two main subcomponents: the Authorization engine and the Policy Administration Point (PAP). [1]

Detailed information about this component can be found Section 4.2 of D4.1 Design of the Security Framework I due on M12.

3.1.3 RAAS (RabbitMQ[2] authentication and authorization service)

This component in development is an http service whose main task is enabling the use of the Authentication (Keycloak) and Authorization (EPICA) services by the Message Broker (RabbitMQ).

RAAS will be able to work in two modes:

1. RAAS will be the responsible to request and manage tokens from Authentication service (Keycloak) and perform authorization request to Authorization service (EPICA) with the obtained tokens. The clients make login in the message broker with username and password.
2. RAAS will be only responsible to verify the validity of tokens from Authentication service (Keycloak) and perform authorization request to Authorization service (EPICA) with the provided tokens. The

[2] https://www.rabbitmq.com/

clients are responsible to obtain and manage the authentication tokens and provide them to RAAS. The clients make login in the message broker with the token from Authentication service, no password involved in this mode.

Detailed information on this component can be found on Section 4.1 of this deliverable and in Section 4.3 of D4.1 Design of Security Framework I due on M12.

3.1.4 XL-SIEM

This component, with the help of the SIEM Agents responsible for data collection and deployed within the monitored infrastructure, provides capabilities of a SIEM solution with the advantage of being able to handle large volumes of data and raise security alerts from a business perspective, thanks to analysis and event processing in Storm cluster. The main functionalities of the XL-SIEM can be summarized in the next points:

- Real-time collection and analysis of security events.
- Prioritization, filtering and normalization of the data gathered from different sources.
- Consolidation and correlation of security events to carry out a risk assessment and generation of alarms and reports [1].

Detailed information about XL-SIEM can be found in Section 4.4 of D4.1 Design of Security Framework I due on M12.

3.1.5 Reverse proxy – Nginx[3]

This component is responsible for directing client requests to the appropriate backend server and also securing communication by enabling the use of TLS[4] (Transport Layer Security) cryptographic protocol. TLS provides security over a computer network, and aims primarily to provide privacy and data integrity between two communicating applications. The use of a reverse proxy also provides an additional defence layer against security attacks by protecting identities of servers and services. [1]

A high level diagram on how a reverse proxy works can be seen in Figure 2 below.

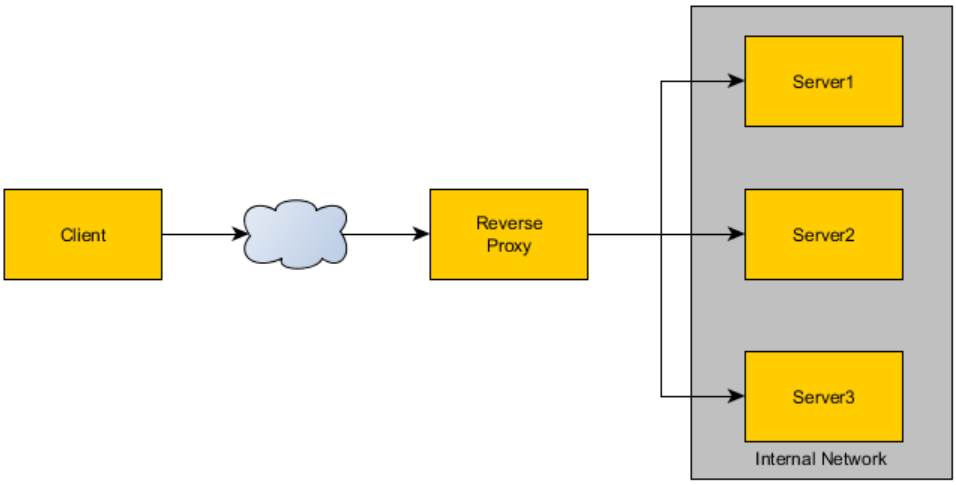

Figure 2 - Reverse proxy diagram

More information in deliverable Design of Security Framework I, Section 4.5, due on M12.

[3] https://nginx.org/en/
[4] https://tools.ietf.org/html/rfc5246

3.1.6 Blockchain – Multichain

It is not the purpose of this deliverable to cover in depth the blockchain components that form part of the COMPOSITION Security Framework, which will be done in the future deliverable D4.3 The COMPOSITION blockchain planned for M30; but just give a hint on some uses proposed for the blockchain technology within the Security Framework scope. The COMPOSITION blockchain is based on Multichain[5].

3.1.6.1 Public Key Infrastructure (PKI)

Since it is proposed that all messages flowing in the COMPOSITION platform through the COMPOSITION Message Broker (RabbitMQ[6]) must be signed using JWS[7] (JSON Web Signature) standard proposed by IETF[8] (see Section 5.2), there is the need to make available to the subscribers of messages the public keys so it is possible for them to verify the digital signature. Instead of using the common approach of publishing the public keys through a web site or a web service, in COMPOSITION we plan to use blockchain technology to make these public keys available.

The idea in the beginning is simple; the message publishers put in their blockchain node their public key while maintaining the private key locally and secret. The public keys published will be replicated on all blockchain nodes connected and keeping a copy of them making it accessible to all subscribers that have the rights to read them.

3.1.6.2 Message Logging

Along with message digital signature (see Section 3.1.6.1 and Section 5.2) COMPOSITION is going to log all the messages sent through the platform. To keep this log blockchain technology is going to be used too. The idea is that the publisher of a message should calculate the hash of the message using a hash cryptographic function (to be decided) and store the result hash value in the blockchain along with some metadata. Upon receiving a message, a subscriber can calculate the hash of the received data and can look for it in the blockchain, ensuring this way the integrity of the data received. This together with the digital signature of the message is going to give the subscriber security to trust on the message received. More information about hashing cryptographic functions on Section 5.3

The following diagram (Figure 3) gives a high-level overview on the signing and logging procedures and how data flow between the components involved.

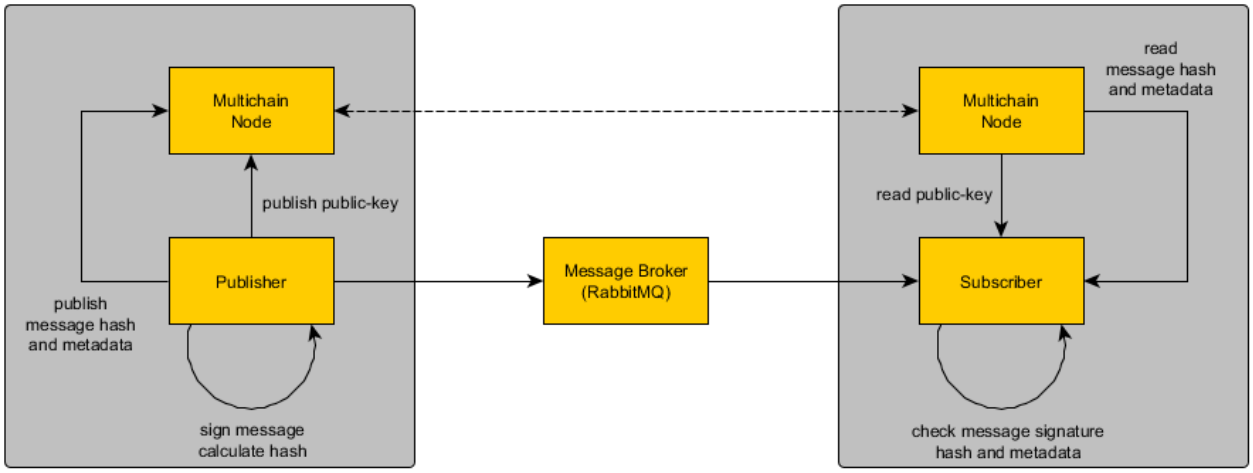

Figure 3 - Overview of signing and logging of messages

A more detailed view on the whole process of publishing and subscribing in COMPOSITION taking into account the use of the methods of signing messages and keep log of them, proposed in COMPOSITION

[5] https://www.multichain.com/
[6] https://www.rabbitmq.com/
[7] https://tools.ietf.org/html/rfc7515
[8] https://www.ietf.org/

Security Framework, as well as the steps to validate the signature and the content of the message can be seen in the flowchart diagram below (Figure 4).

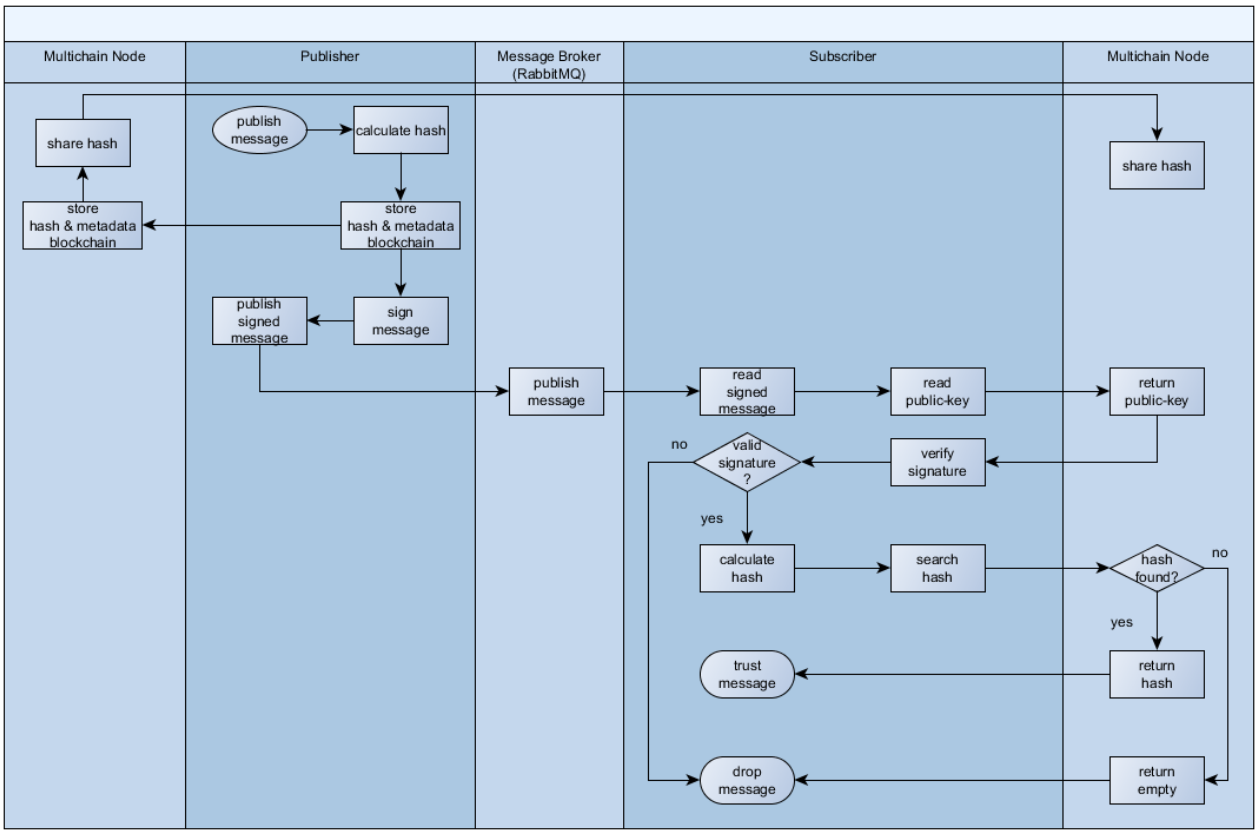

Figure 4 - Flowchart diagram publish-subscribe procedure in COMPOSITION

3.1.6.3 Reputation Model

COMPOSITION is going to define a reputation model adding another level on trust. The reputation model definition itself is not covered in this section but in Section 5.1. Blockchain is also the technology to be used to store the reputation of the stakeholders and the way to share it with other stakeholders. It will also keep track of the reputation over time due to the immutability nature of the blockchain technology. For more information about Reputation Model and blockchain refer to Section 5.1.2.

3.2 RAAS Deployments

Since COMPOSITION will have at least two Message Brokers (RabbitMQ[9]), one for the Inter-Factory scenarios and another for the Intra-Factory scenarios, the same number of RAAS services need to be deployed. The following sections will cover two recommended ways to deploy the RAAS services, one with the RAAS services in the same premises as the rest of the Security Framework components (Section 3.2.1) and another with the RAAS services deployed along with the COMPOSITION Message Brokers in the same premises (Section 0).

3.2.1 Default

The default architecture requires the RAAS services deployed along with the Authentication Service (Keycloak[10]) and the Authorization Service (EPICA) and all of them behind a reverse proxy in our case Nginx[11]. The default architecture can be seen in the diagram below (Figure 5) and shows apart how the components involved interact but also if the communications are encrypted using TLS[12] or not. There is no

[9] https://www.rabbitmq.com/
[10] http://www.keycloak.org/
[11] https://nginx.org/en/
[12] https://tools.ietf.org/html/rfc5246

need to encrypt all communication using TLS for services and components that are not exposed directly to Internet, as TLS encrypted communication comes with an overhead on the network traffic.

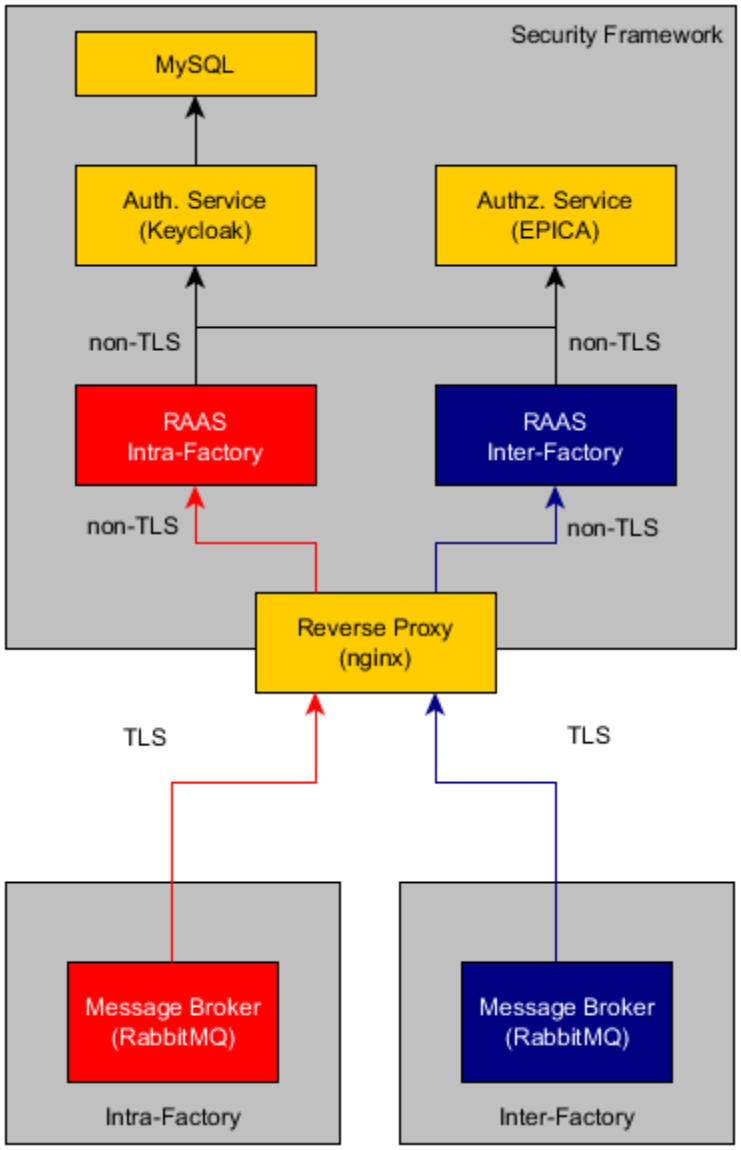

Figure 5 - Security Framework default architecture

The following diagram (Figure 6) shows a real-life deployment using Docker[13] containers of the default architecture shown before. The only port exposed to Internet it´s the one used by the reverse proxy to enable the encrypted communication; and it´s the default for TLS[14] communication, 443.

[13] https://www.docker.com/what-docker
[14] https://tools.ietf.org/html/rfc5246

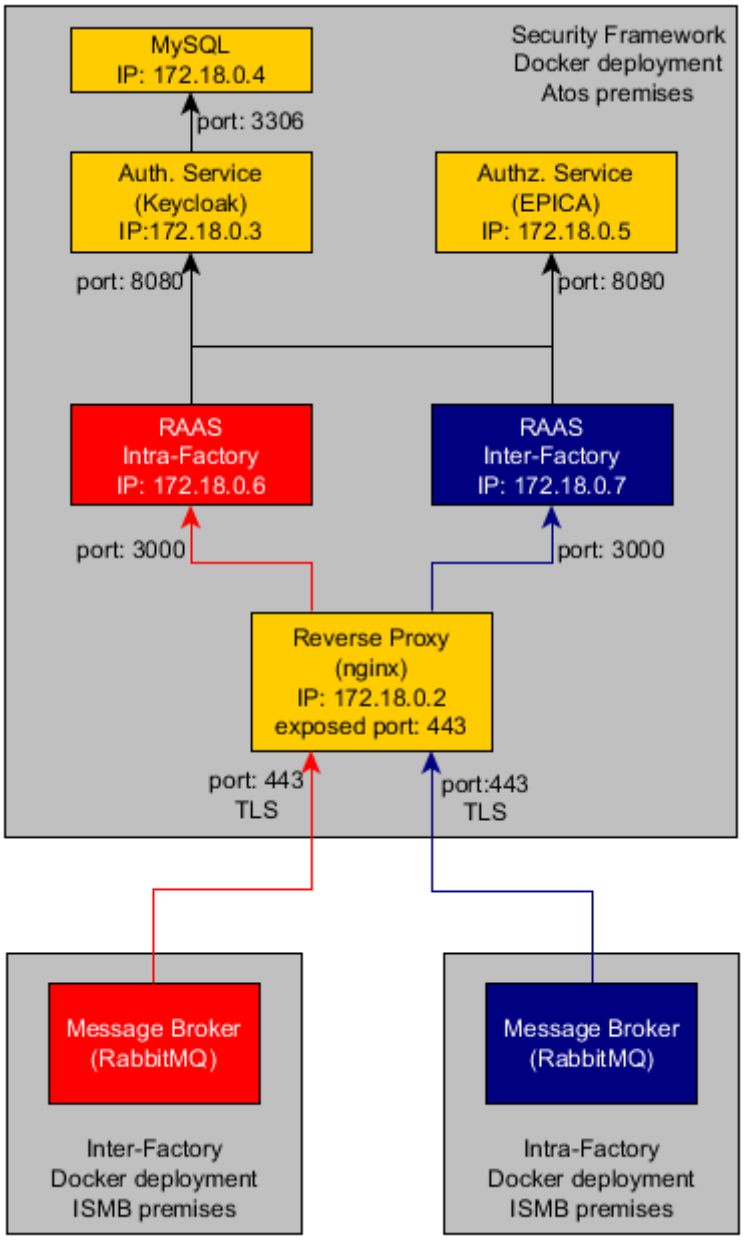

Figure 6 - Docker deployment for default architecture

3.2.2 Alternative

There is an alternative architecture to use RAAS services. This alternative requires RAAS services to be deployed along with the message brokers in the same local network. This architecture can be seen in the following diagram (Figure 7).

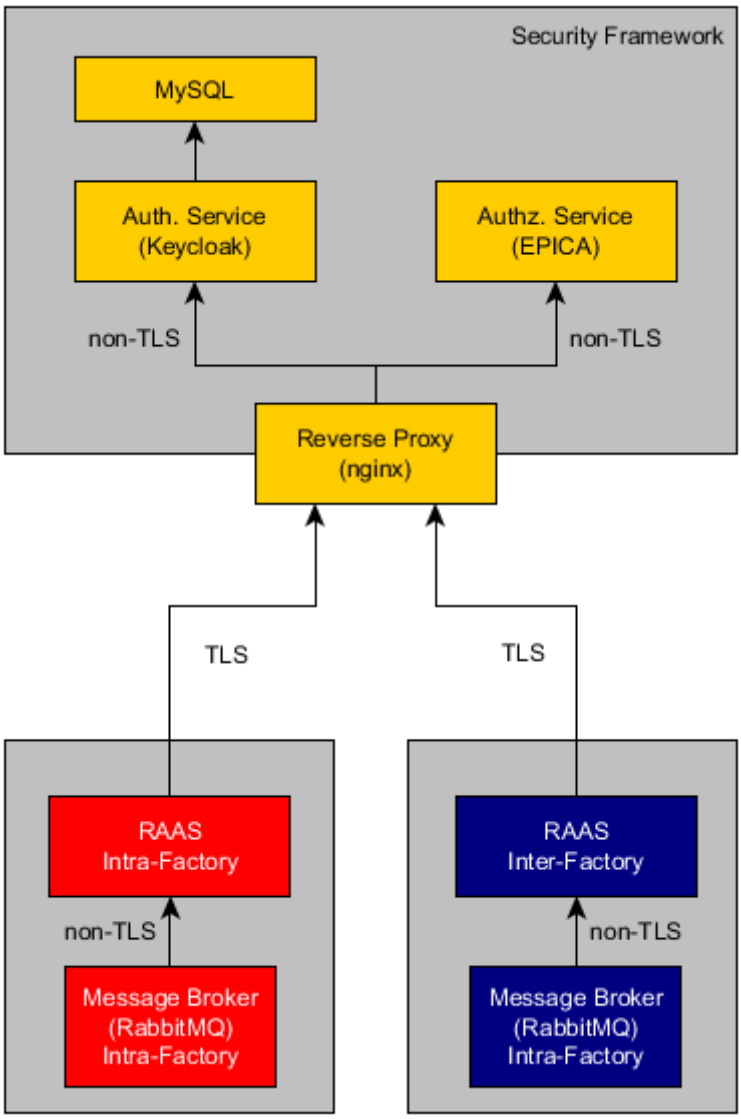

Figure 7 - Security Framework alternative architecture

The following diagram (Figure 8) shows a real-life deployment using Docker[15] containers of the alternative architecture shown before. As with the default architecture the only port exposed to Internet is the one used by the reverse proxy to enable the encrypted communication; and it is the default for TLS communication, 443. In this case, the communication of RAAS with the Authentication Service and the Authorization Service is encrypted, since it happens through Internet and not in local network as the default architecture.

[15] https://www.docker.com/what-docker

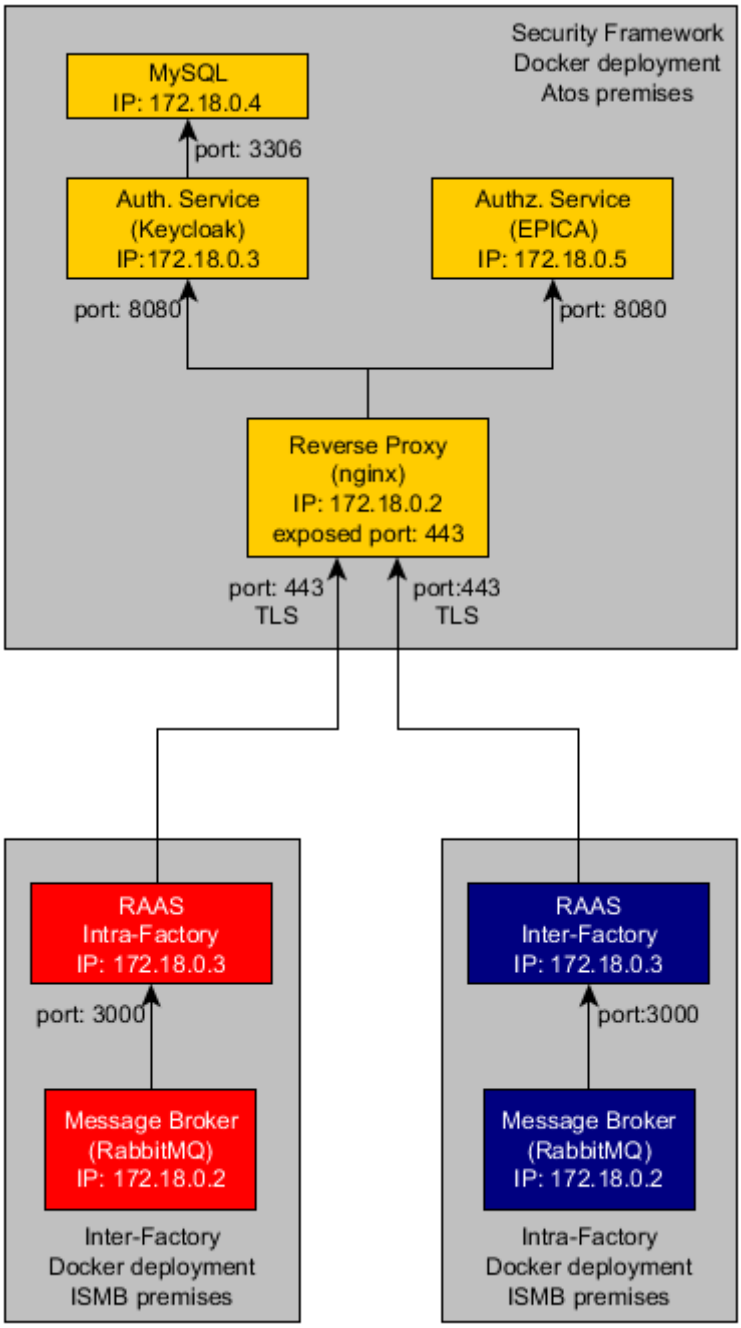

Figure 8 - Docker deployment for alternative architecture

4 Security Framework Components – Configuration, Development, Integration

This section will cover the work done since M12 in the Security Framework related with the configuration of services, development of components, and customization and integration of services. A brief list of the work done below and detailed information on each of the following sub-sections:

- Development of RAAS (see Section 4.1)
- Deployment, configuration and customization of Keycloak[16] (see Section 4.2)
- Deployment and integration of EPICA with Keycloak (see Section 4.3)
- Development of cyber-agent for XL-SIEM (see Section 4.4)
- Deployment and configuration of Nginx[17] reverse proxy (see Section 4.5)

4.1 RAAS (RabbitMQ[18] Authentication/Authorization Service)

RAAS is an http service in development and part of the Security Framework that enables the use of COMPOSITION Authentication Service (Keycloak) and Authorization Service (EPICA) with COMPOSiTION Message Broker (RabbitMQ). Itis not the scope of this deliverable to describe the configuration of the message broker to use RAAS, information on this topic can be found on Section 6 of D5.9 Intrafactory interoperability layer I due on M18.

As RAAS is in development the information detailed in this deliverable may change as the development progresses, the information about this component will be updated accordingly on the upcoming deliverables.

The endpoints RAAS shall expose to communicate with COMPOSITION Message Broker (RabbitMQ) are described below on Table 1

Table 1 - RAAS exposed endpoints

Path	Method	Parameters	Response
/auth/user	POST	username password	allow [list of tags], deny
/auth/vhost	POST	username vhost: name of the virtual host being accessed ip: client ip address	allow, deny
/auth/resource	POST	username vhost: name of the virtual host containing the resource resource: type of resource (exchange, queue, topic) name: name of the resource permission: access level to the resource (configure, write, read)	allow, deny
/auth/topic	POST	username vhost: the name of the virtual host containing the resource resource: the type of resource (topic in this case) name: name of the exchange permission: access level to the resource (write or read) routing_key: routing key of a published message (when the permission is write) or routing key of the queue binding	allow, deny

[16] http://www.keycloak.org/
[17] https://nginx.org/en/
[18] https://www.rabbitmq.com/

		(when the permission is read)	

The Authentication Service (Keycloak) endpoints RAAS shall use to perform authentication actions are described on Table 2 below:

Table 2 - Authentication Service (Keycloak) endpoints used by RAAS

/auth/realms/composition/protocol/openid-connect/token	
Action	**Parameters**
login (authenticate user and get set of tokens)	grant_type=password client_id=rabbitmq username=xxx password=xxx client_secret=xxx response_type=token
refresh-token (obtain new set of tokens when current expired)	grant_type=refresh_token client_id=rabbitmq client_secret=xxx refresh_token=xxx
/auth/realms/composition/protocol/openid-connect/logout	
logout (close session)	client_id=rabbitmq client_secret=xxx refresh_token=xxx

RAAS will be able to work in two modes explained in detail in the next sections Username and Password Mode (Section 4.1.1) and Token Mode (Section 4.1.2).

4.1.1 RAAS – Mode: Username and Password

In this mode RAAS will be the responsible to request and manage tokens from Authentication Service (Keycloak) and perform authorization request to Authorization Service (EPICA) with the obtained tokens. The clients make login in the message broker with username and password.

On a high level view the authentication process on this mode follows the following steps:

1. Credentials, username and password, are entered by user (or message broker client, publisher or subscriber)
2. Credentials are passed from message broker to RAAS
3. RAAS performs authentication against Authentication Service (Keycloak)
4. RAAS allow or deny the authentication request

The diagram below (Figure 9) describes on a very high level the authentication process and the components involved.

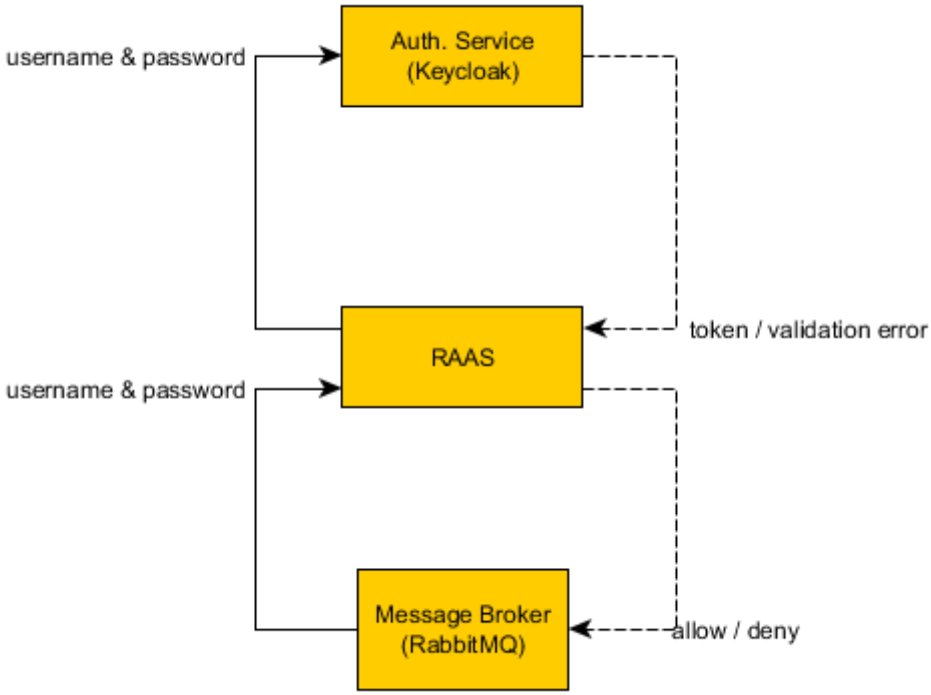

Figure 9 - RAAS: Authentication in mode Username and Password

A detailed description of the authentication procedure in this mode can be seen in the flowchart diagram below (Figure 10)

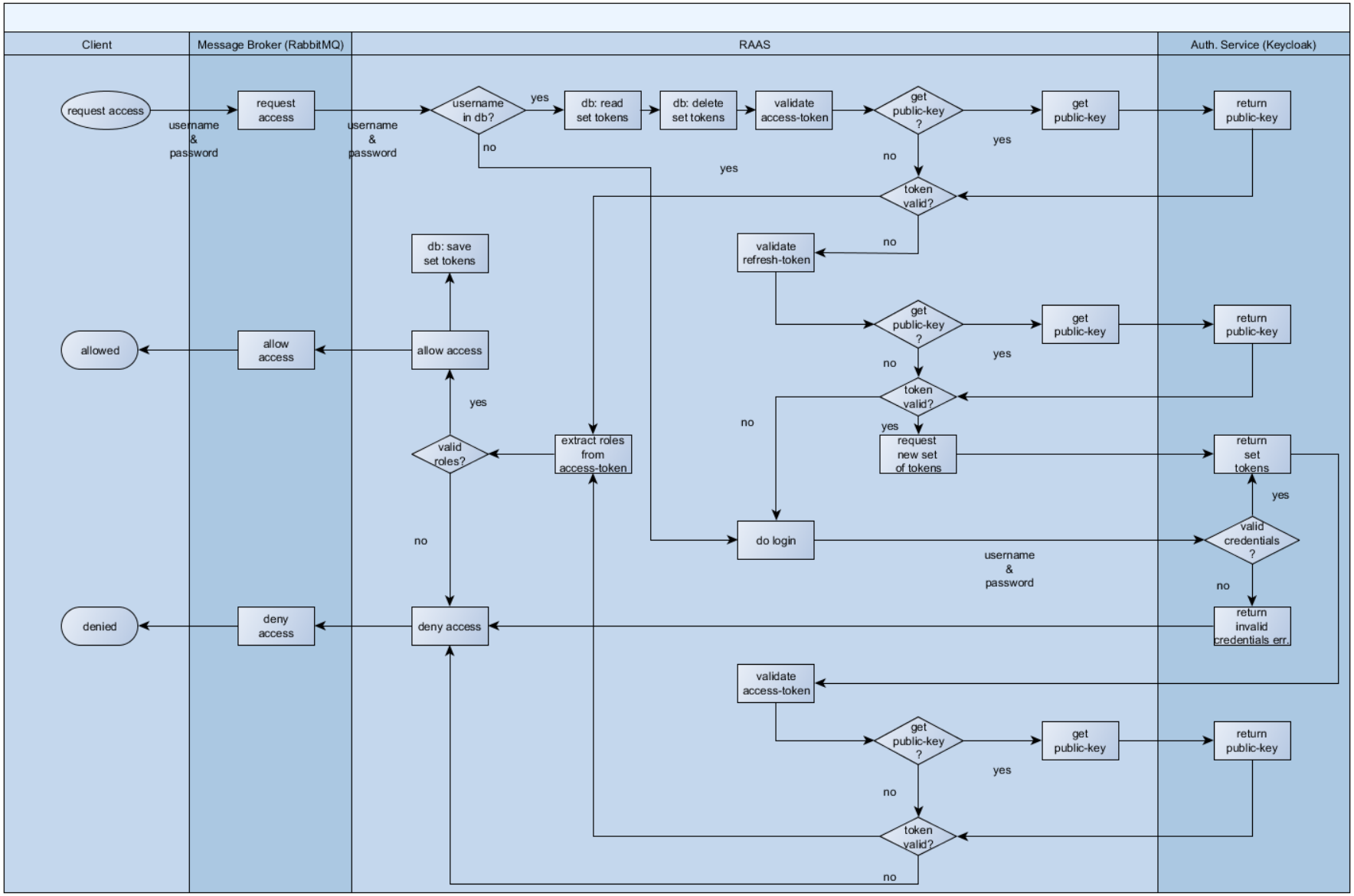

Figure 10 - Flowchart RAAS Authentication and Authorization in mode Username and Password

The process of authorization in this mode -unlike the authentication one- does not involve the password by the resource information to be accessed.
In this case the following steps take place:

1. Message broker request access to a resource
2. Request is passed to RAAS
3. RAAS look for token of already authenticated user
4. RAAS request Authorization Service (EPICA) access to the resource using token and resource info.
5. RAAS allow or deny based on the response from Authorization Service (EPICA)

The diagram below (Figure 11) describes on a very high level the authorization process:

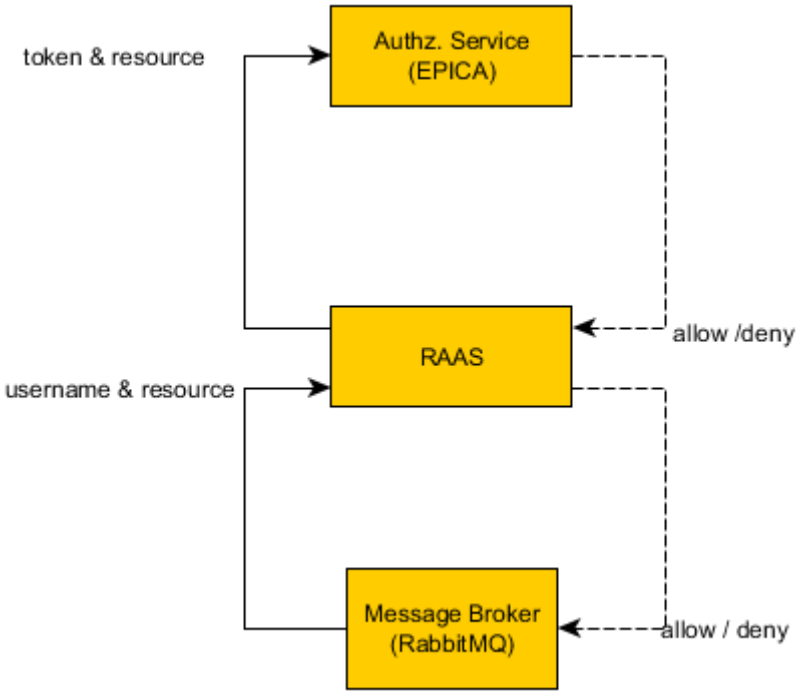

Figure 11 - RAAS Authorization in mode Username and Password

4.1.2 RAAS – Mode: Token

In this mode RAAS will be only responsible to verify the validity of tokens from Authentication Service (Keycloak[19]) and perform authorization request to Authorization Service (EPICA) with the provided tokens. The clients are responsible to obtain and manage the authentication tokens and provide them to RAAS. The clients make login in the message broker with the token from Authentication Service (Keycloak), no password involved in this mode.

On a high level view the authentication process on this mode follows the following steps:

1. Client (publisher or subscriber) authenticate against COMPOSITION Authentication Service (Keycloak).
2. Token obtained from Authentication Service (Keycloak) is used to authenticate against message broker. No password involved, the token is passed as username.
3. Token is passed from message broker to RAAS
4. RAAS verify the token
5. RAAS allow or deny the authentication request based on the token verification

[19] http://www.keycloak.org/

The diagram below (Figure 12) describes on a very high level the process of authentication and the components involved.

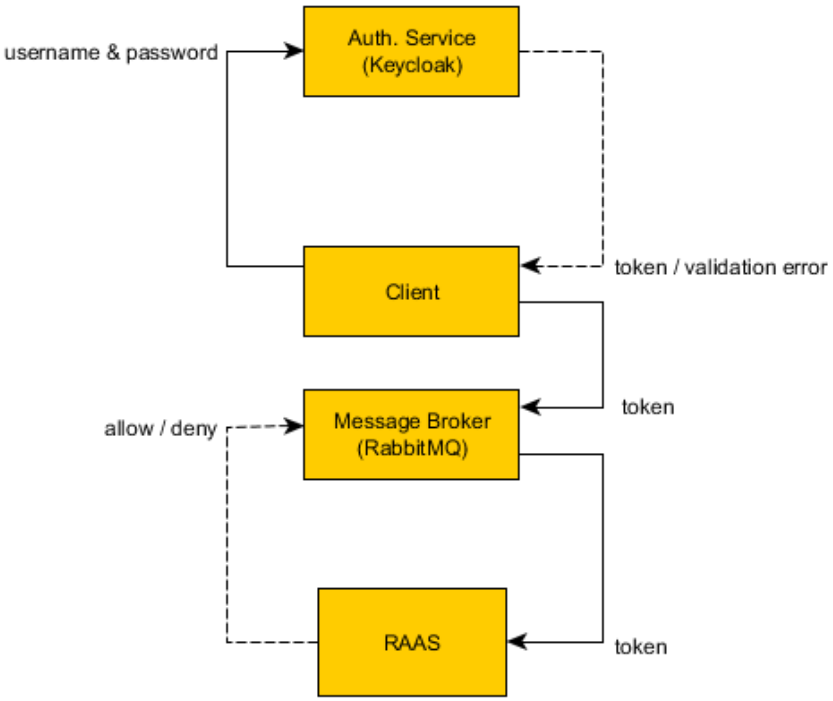

Figure 12 - RAAS Authentication in mode Token

A detailed description of the authentication procedure in this mode can be seen in the flowchart diagram below (Figure 13).

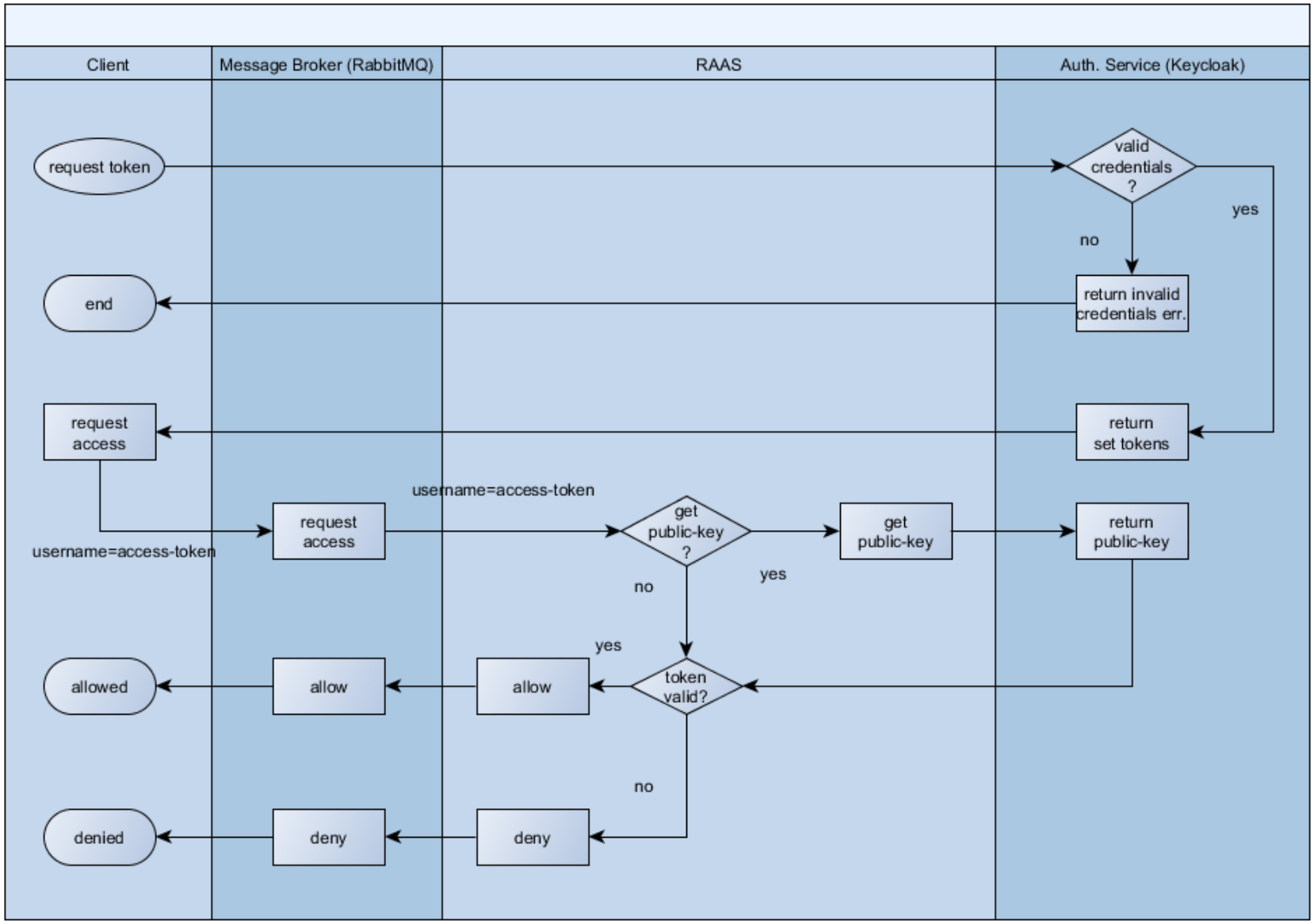

Figure 13 - Flowchart RAAS Authentication and Authorization in mode Token

The process of authorization in this mode has the following steps:

1. Message broker request access to a resource but unlike the previous mode where username was used in this case the username is the token obtained by the client when authenticated directly to the Authentication Service (Keycloak).
2. Request is passed to RAAS
3. RAAS verify token.
4. RAAS request Authorization Service (EPICA) access to the resource using token and resource info.
5. RAAS allow or deny based on the response from Authorization Service (EPICA)

The diagram below (Figure 11) describes on a very high level the authorization process:

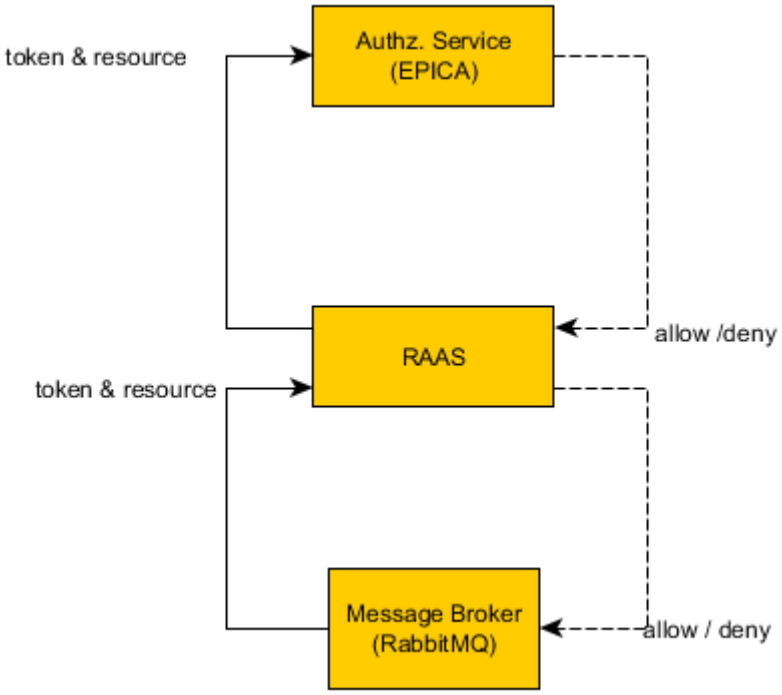

Figure 14 - RAAS Authorization in mode Token

4.2 Authentication Service – Keycloak[20]

This section will cover the work done in Keycloak related to deployment, configuration and customization.

4.2.1 Deployment and Configuration

Keycloak has been deployed as a docker container in Atos premises. Details of the docker container on the screenshot below (Figure 15)

[20] http://www.keycloak.org/

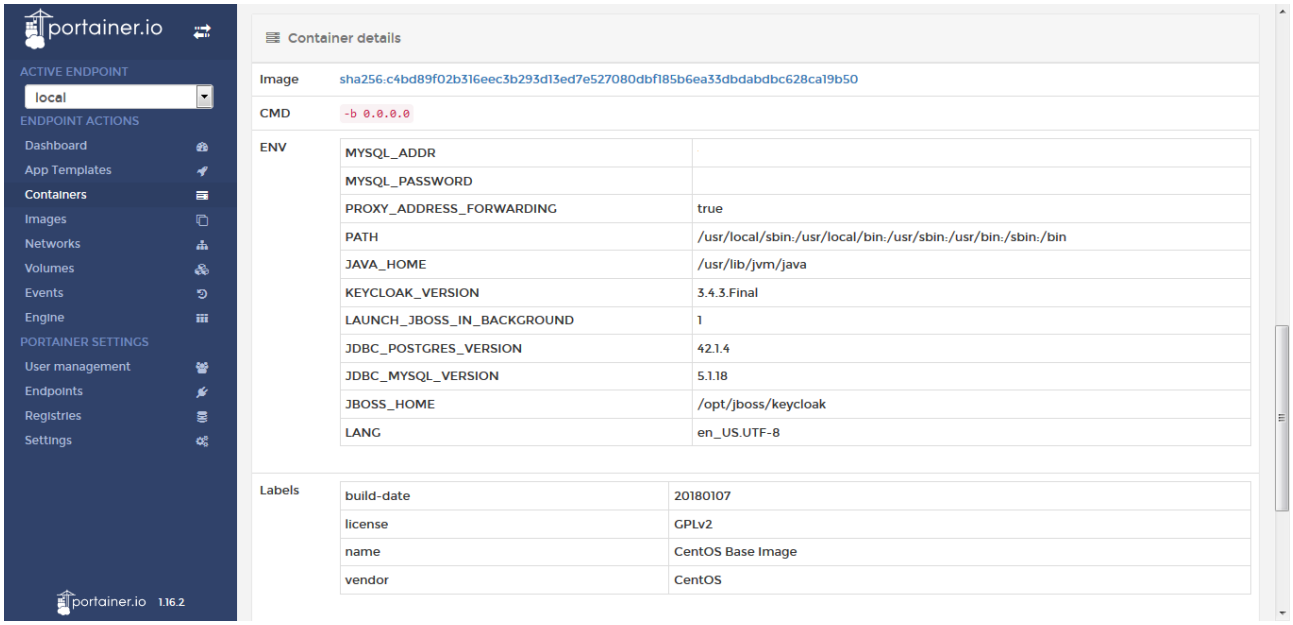

Figure 15 - Keycloak docker container details

To give support to both Inter-Factory and Intra-Factory scenarios, two different realms have been created in Keycloak one realm for Inter-Factory named *composition-inter* and one for Intra-Factory named *composition-intra* (see Figure 16). The use of two different realms will allow the management of clients and users independently for each Inter-Factory and Intra-Factory scenarios.

Figure 16 - Keycloak realms

Into each realm a client named *rabbitmq* has been created (see Figure 17) as well as a role named *rabbitmq*. Into each client the following roles have been created: administrator, management, monitoring and policymaker (see Figure 18). These roles will be used to authorize users and clients accessing RabbitMQ message broker.

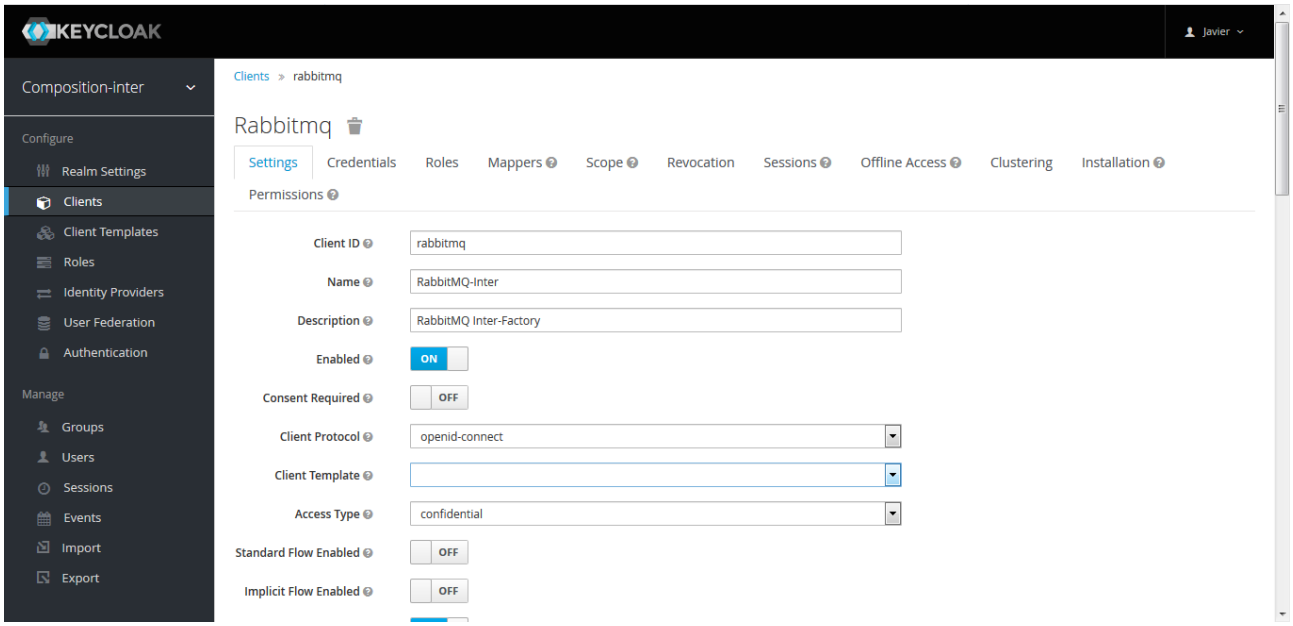

Figure 17 - Keycloak *rabbitmq* client

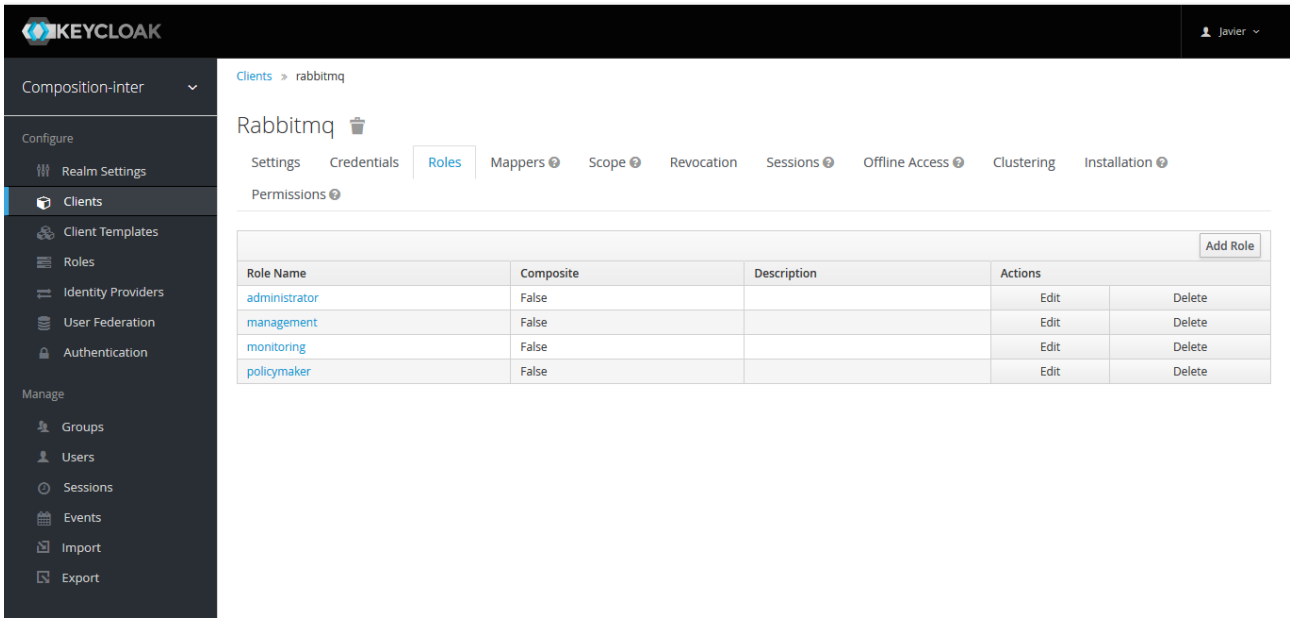

Figure 18 - Keycloak *rabbitmq* client roles

4.2.2 Customization

Although Keycloak is designed to cover most use-cases without requiring custom code, it has a number of Service Provider Interfaces (SPI) for which own providers can be implemented (Keycloak Service Provider Interfaces (SPI), n.d.). From the available list of SPI, COMPOSITION is implementing the Protocol-Mapper SPI to create a Custom-Mapper which will enable the ability to add into tokens additional information from external sources, like databases.

One scenario where Custom-Mapper can be very useful is the one where COMPOSITION users are able to assign roles to other COMPOSTION users; so the latter are granted access to resources from the former, without the need of administration rights in Keycloak.

The next screenshot (Figure 19) shows the list of installed mappers in Keycloak where is listed atos-custom-mapper, which is a prototype of the Custom-Mapper.

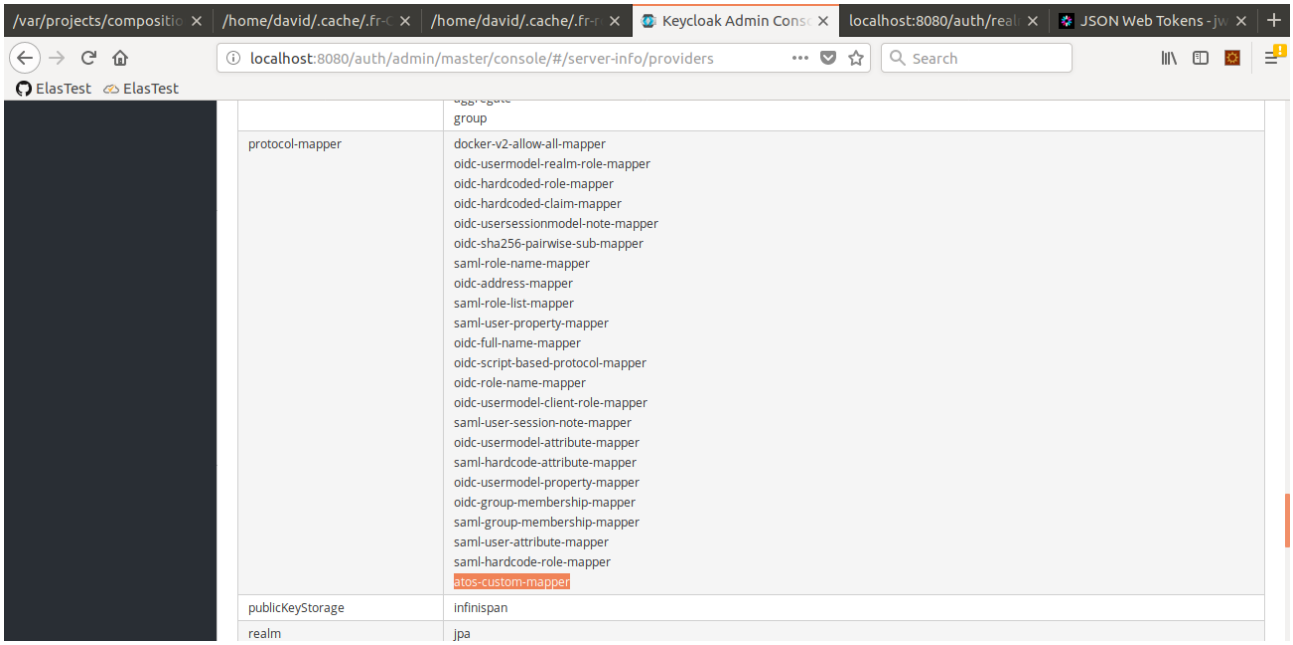

Figure 19 - Custom-mapper in protocol-mappers list installed

Screenshot below (Figure 20) shows atos-custom-mapper prototype assigned to rabbitmq client.

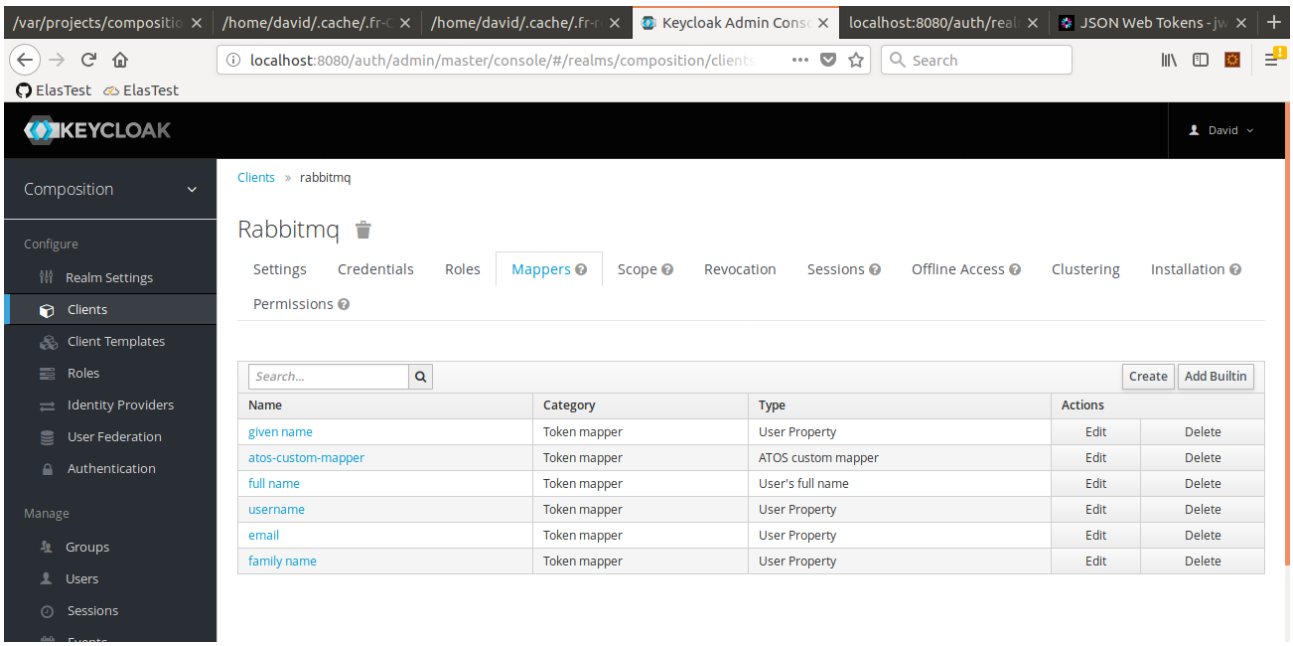

Figure 20 - Custom-mapper used in client

The next screenshot (Figure 21) shows the details of atos-custom-mapper prototype installed on Keycloak and used by rabbitmq client.

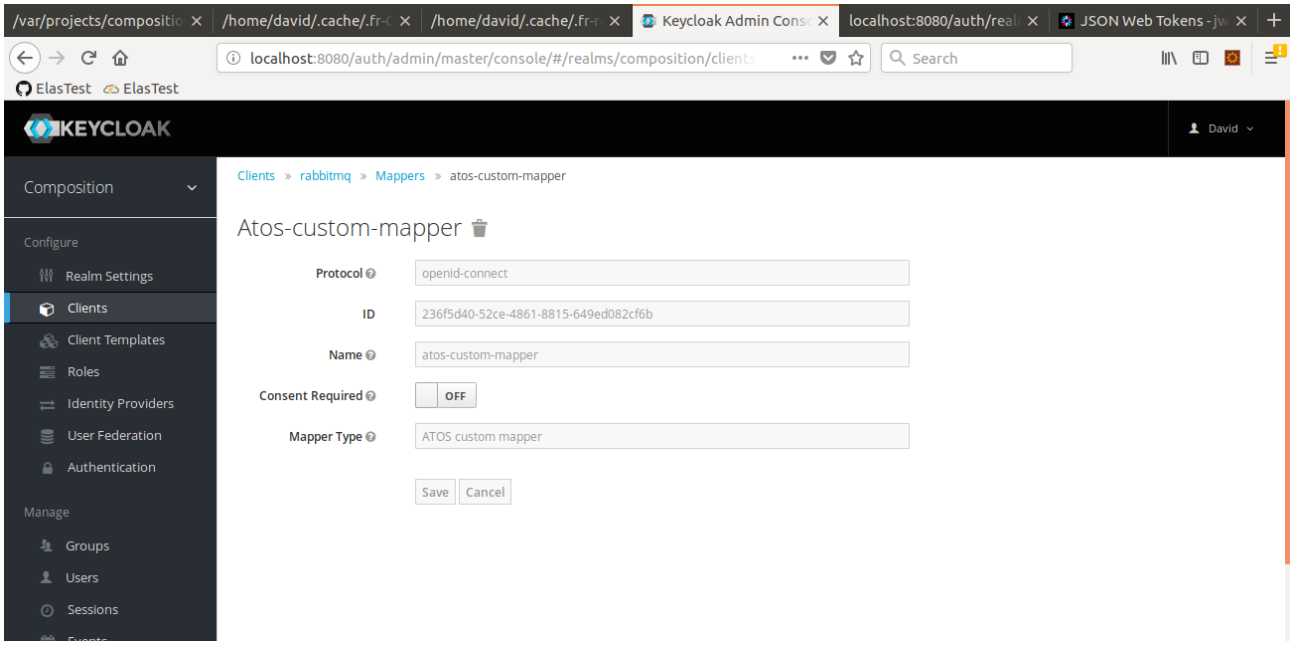

Figure 21 - Custom-mapper details

4.3 Authorization Service – EPICA

Authorization Service (EPICA) is currently being integrated with Authentication Service (Keycloak). The API exposed by EPICA is being modified to be able to deal with tokens from Keycloak. Once integration is completed and tested a first set of authorization policies will be created, involving COMPOSITION partners in this task.

4.4 XL-SIEM

A new cyberagent named *l-ads* is in development for XL-SIEM. This new agent makes use of neural networks to determine if an alert should be raised or not. The agent analyses the network traffic of the monitored interface and raises an alert based on the train done to the neural network.

Agent l-ads make use of NetFlow[21] network protocol to be used as source to analyse the network traffic and Softflowd[22] as the NetFlow exporter which aggregates packets into flows and exports flow records towards l-ads to be analysed.

- NetFlow is a network protocol developed by Cisco used in their routers for collecting IP traffic information and monitoring network traffic. NetFlow exports flow information in UDP[23] datagrams in one of the following formats: v1, v5, v7, v8 and v9.

  Agent l-ads supports v5 datagram format; each UDP datagram is composed of a header (see Table 3) and N flow records (see Table 4), being 1 <= N <= 30 and is specified by the *count* field in the header.

Table 3 - NetFlow v5 flow header format

bytes	content	description
0-1	version	NetFlow export format version number
2-3	count	Number of flows exported in this packet (1-30)
4-7	sys_uptime	Current time in milliseconds since the export device booted
8-11	unix_secs	Current count of seconds since 0000 UTC 1970
12-15	unix_nsecs	Residual nanoseconds since 0000 UTC 1970

[21] https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html
[22] https://www.mindrot.org/projects/softflowd/
[23] https://tools.ietf.org/html/rfc768

16-19	flow_sequence	Sequence counter of total flows seen
20	engine_type	Type of flow-switching engine
21	engine_id	Slot number of the flow-switching engine
22-23	sampling_interval	First two bits hold the sampling mode; remaining 14 bits hold value of sampling interval

Table 4 - NetFlow v5 flow record format

bytes	content	description
0-3	srcaddr	Source IP address
4-7	dstaddr	Destination IP address
8-11	nexthop	IP address of next hop router
12-13	input	SNMP index of input interface
14-15	output	SNMP index of output interface
16-19	dPkts	Packets in the flow
20-23	dOctets	Total number of Layer 3 bytes in the packets of the flow
24-27	first	SysUptime at start of flow
28-31	last	SysUptime at the time the last packet of the flow was received
32-33	srcport	TCP/UDP source port number or equivalent
34-35	dstport	TCP/UDP destination port number or equivalent
36	pad1	Unused (zero) bytes
37	tcp_flags	Cumulative OR of TCP flags
38	prot	IP protocol type (for example, TCP = 6; UDP = 17)
39	tos	IP type of service (ToS)
40-41	src_as	Autonomous system number of the source, either origin or peer
42-43	dst_as	Autonomous system number of the destination, either origin or peer
44	src_mask	Source address prefix mask bits
45	dst_mask	Destination address prefix mask bits
46-47	pad2	Unused (zero) bytes

- Softlowd is a flow-based network traffic analyser capable of Cisco Netflow data export. It aggregates packets into flows and exports flow records towards one or more flow collectors.

Instead of analysing all packets flowing across the monitored network interface the agent will instead analyse NetFlow v5 datagrams collected by Softflowd.on the monitored network interface and exported to an IP and a port as UDP datagrams.

The diagram below (Figure 22) gives an overview of l-ads architecture with the components involved and the interaction between them.

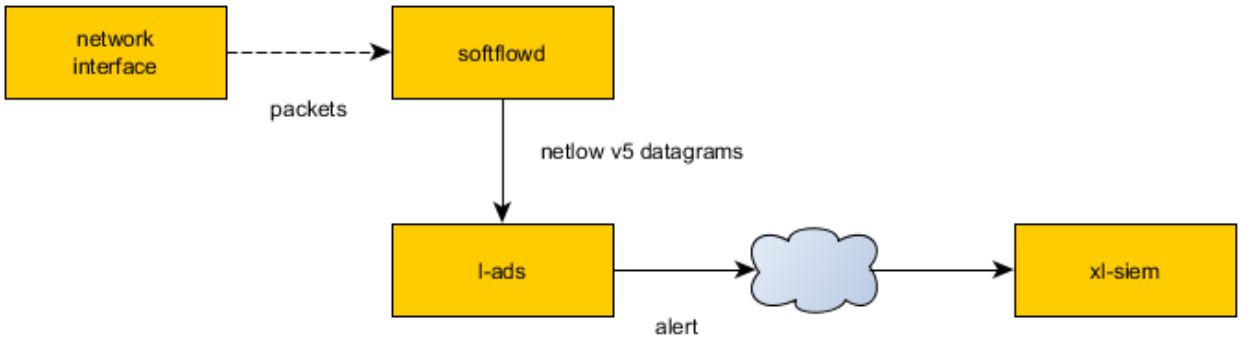

Figure 22 - l-ads architecture overview

4.5 Reverse proxy – Nginx[24]

Nginx has been deployed as a Docker[25] container in Atos premises. Nginx configuration files as well as certificates used for providing TLS support are stored outside the container for better management and maintenance. Current Nginx configuration enables only encrypted TLS connections through port 443.

The screenshot below (Figure 23) shows the details of the Docker container deployment.

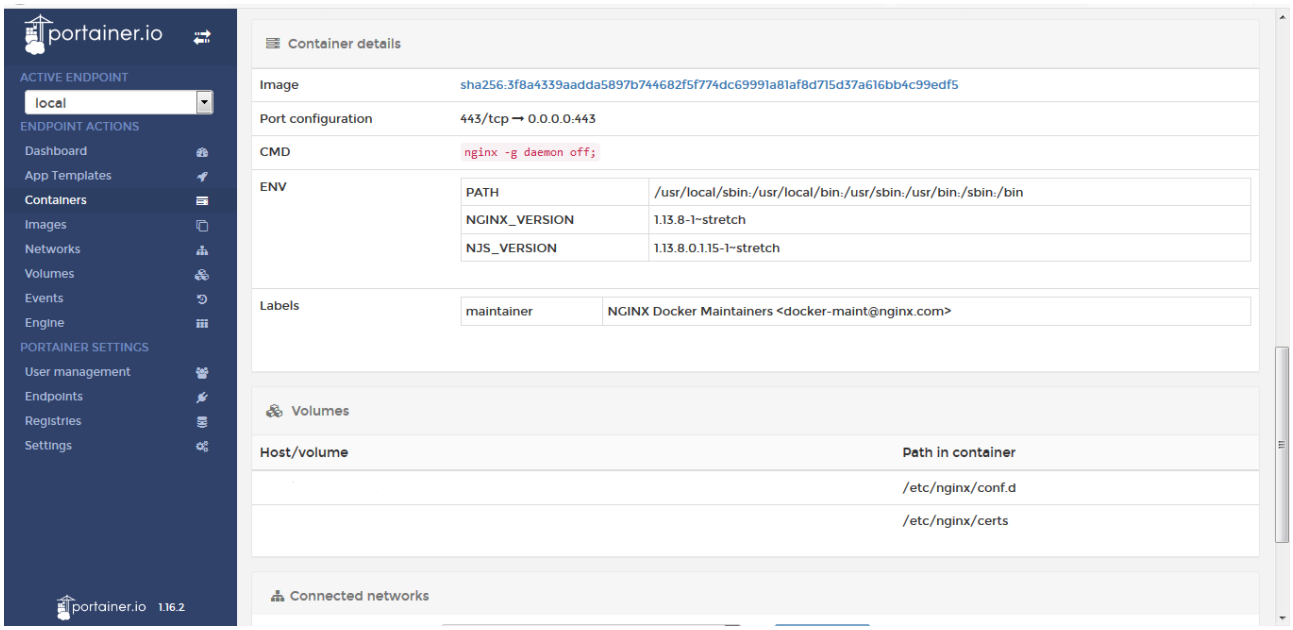

Figure 23 - Nginx docker container details

5 Integrity and trust of information

5.1 Reputation Model

Before talking about the COMPOSITION Reputation Model, it is necessary to point out some typical characteristics of these models. First of all, they are based on the concept of "reputation", often misunderstood with the one related to "trust", as explained in (Jaydip Sen, 2010) and (Ramana et al, 2010). In (Hoffman et al, 2009) and (Moyano et al, 2012), reputation is considered as a means for computing trust, together with other context-dependent factors. Always in (Moyano et al, 2012), as well as in (Artz et al, 2007), a more detailed explanation is provided, associating a completely objective nature to the concept of reputation, differently from trust.

Considering (Jøsang et al, 2007), reputation is defined as "what is generally said or believed about a person's or thing's character or standing". In this survey, also a very interesting connection between the two concepts is expressed, through the following statements: "I trust you because of your good reputation" and "I don't trust you despite your bad reputation". These two statements express clearly the different nature between the two concepts.

This differentiation is essential in order to design a Reputation Model. Reputation must be computed taking into account the specific scenario where the model is applied: considering an online marketplace scenario, for instance, every time an interaction takes place, a local reputation score must be computed by the trustor (the agent who makes the request) and aggregated with the other scores related to the previous interactions, with the same trustee: in this case the obtained score will be updated when a new interaction occurs, but, at the same time, it will represent also a global view of trustee's historical behaviour, from the trustor point of view. Each new value can be seen as a feedback representing the trustor "satisfaction" for the received service, in that specific interaction. Then, the updated global reputation could be used by the trustor or by

[24] https://nginx.org/en/
[25] https://www.docker.com/what-docker

other agents as well, for taking future trust-based decisions related to the same trustee, for possible future interactions.

This happens, for instance, in the main, and simple, scenario of e-commerce marketplaces, the eBay[26] Reputation Model (Resnick et al, 2002): in this case the trustor requests a service from a provider (the trustee), and gives a positive (+1) or negative (-1) feedback, after the service has been received. All the scores related to the same trustee are aggregated by a centralised unit, using a basic summation operator, which provides the global reputation score for that trustee. This schema is very basic, and suffers from several issues, such as the ballot stuffing (e.g., ratings repeated many times) and unfair ratings problems, as stated in (Jøsang et al, 2007).

Another common solution, related to online marketplace scenarios, is the REGRET model (Sabater et al, 2001), where reputation is defined as "the opinion or view of one about something". In this model, individual reputation inferred from direct interaction, is aggregated, locally by the trustor, with other social and ontological factors (e.g., social relationships among involved entities, combination of different reputation values related to different aspects), as well as with reputation scores provided by other entities about their past interactions with the same trustee, for obtaining the final trust value.

Depending on how reputation is evaluated, the model could be centralised (e.g., eBay model), if a single entity is in charge of computing it for every involved parties, or decentralised (e.g., REGRET model), where each entity compute their local reputation values referred to others, before disseminating them.

In (Vavilis et al, 2014), some guidelines for designing Reputation Models are provided, in terms of requirements and features that they should implement for fulfilling them, as well as a comparison among most known models.

For a more detailed summary on Reputation Models, many surveys can be consulted, such as (Ruohomaa et al, 2007) (Hoffman et al, 2009) (Sabater et al, 2005) (Jøsang et al, 2007) and (Gomez et al, 2011).

5.1.1 COMPOSITION Reputation Model

Regarding the COMPOSITION Reputation Model, the basic idea is to follow the reference model described in (Vavilis et al, 2014), in order to infer the basic requirements that should be satisfied, depending on the specific context of the project. Each agent of the marketplace must be able to provide a rating related to each single transaction, when they act as the requestor (trustor): these ratings could be integer values within a predefined interval, for expressing different level of "satisfaction", for example:

- Not satisfied → 1
- Partially satisfied → 2
- Satisfied → 3
- Very Satisfied → 4
- Completely Satisfied → 5

Each single rating will be aggregated with the previous ratings associated to the same provider (trustee), through an aggregator operator, and the result could be a real number belonging to the interval [1, 5] (if the above mentioned values would be used). Considering (Torra et al, 2007), (Ravana et al, 2009), (Torra, 2017), (Derakhshandeh et al, 2011) and (Cornelis et al, 2010), there are many of them that could be checked, such as:

- **Arithmetic Mean (AM)**: It consists in the sum of a certain number of values, which is then divided for the total number of values themselves (Torra et al, 2007), (Ravana et al, 2009). The formula is shown in Equation 1.

$$\mathbf{AM} = \frac{\sum_{i=1}^{t} \boldsymbol{Xi}}{\boldsymbol{t}} \quad (\mathbf{1})$$

[26] https://www.ebay.com/

- **Geometric Mean (GM)**: It indicates the central tendency, called also typical value, of a set of numbers by using their product. The geometric mean is defined as the t^{th} root product of the t numbers (Ravana et al, 2009) and the formula is expressed in Equation 2. It is more stable than the arithmetic mean.

$$\mathbf{GM} = \left(\prod_{i=1}^{t} \boldsymbol{Xi} \right)^{\frac{1}{t}} \quad (\mathbf{2})$$

- **Weighted Mean (WM)**: It has many similarities with the arithmetic mean. However each single value of the sum is weighted accordingly to its "importance" for the computation of the final result. The formula can be seen in Equation 3, where X_i is the i^{th} value, while W_i the i^{th} weight (Torra, 2017).

$$\mathbf{WM} = \sum_{i=1}^{t} \boldsymbol{Xi} \,.\, \boldsymbol{Pi} \,(\mathbf{3})$$

- **Ordered Weighted Mean (OWA)**: It is an operator similar to the previous one, with the difference that the set of values (a_1, … , a_n) is ordered decreasingly and, then each value is weighted considering its position, taking into account a weighting vector (w_1, … , w_n), as can be seen in Equation 4. In this way, weights allow expressing whether the importance is given to low, high or central data (Torra et al, 2007), (Derakhshandeh et al, 2011), (Cornelis et al, 2010).

$$\mathbf{OWA}\,(\mathbf{a1}, \ldots, \mathbf{an}) = \sum_{i=1}^{n} \boldsymbol{Pi} \,.\, \boldsymbol{Bi} \quad (\mathbf{4})$$

  Where:
  $\boldsymbol{P_i}$ is the weight associated to the i^{th} data after ordering them.
  $\boldsymbol{B_i}$ corresponds to a permutation of the *ai*, in such a way that the ordering goes from the largest value to the lowest one.

The chosen operator should be used in the two following cases:

1. When computing/updating local reputation value about a single trustee. Each trustor would be in charge of this operation every time an interaction with a specific trustee occurs. The new rating will be properly aggregated with the older ones. However, as stated in (Gutowska et al, 2009), reputation lifetime should be considered, and this implies assigning a lower importance (weight) to older value (ratings), associated to older interactions.
2. When other agent's opinions about a specific trustee are considered: this factor is known as rater's credibility (Gutowska et al, 2009). Before initiating an interaction with an agent, the trustor should have the possibility to check other reputation values given by the other agents belonging to the marketplace to the chosen trustee. However, in order to avoid considering false, or misleading, reputation scores, all the values should be weighted accordingly to the reputation given by the trustor to the one who is providing the score, and finally aggregated. Then, the final result will be, in turn, aggregated with the actual reputation value computed by the trustor for the considered trustee. Finally, the result, a sort of global reputation value about the trustee, evaluated by the trustor, will be used for making the final trust-based decision about initiating or not the interaction with the counterpart.

It is easy to point out that both the weighted mean and the ordered weighted mean operators could be used as a starting point, considering that they allow weighting all the single values of the formula, for the computation of the final result. The weights will be computed accordingly to the peculiarity of each case, and the aggregator which fits better will be chosen (other aggregators which also allow weighting each value could be considered).

As explained in the previous section, reputation is just a means for computing trust. Other factors could be taken into account for making the final decision, which have a more subjective nature than reputation, as described in (Moyano et al, 2012) and (Sabater et al, 2001), such as psychological, sociological and ontological factors (i.e., competitors usually tend to avoid interactions among each other despite good

reputation values), as well as other contextual information which can be extracted from the message exchanged by the involved agents.

However, they should not be considered for the dynamic computation of the reputation. They could be used, together with the final reputation value, for making the trust-based decision, and aggregated using a specific trust metric (Moyano et al, 2012). However, the final reputation value alone could already enough as an indicator of trustworthiness, for the purpose of COMPOSITION, for each agent of the marketplace.

In Table 5 the requirements of the COMPOSITION Reputation Model are listed.

Table 5 - COMPOSITION Reputation Model Requirements

ID	REQUIREMENT
R1	Reputation and ratings should discriminate agents behaviour
R2	Incorrect reputation values should be detected (raters credibility), when used
R3	Local reputations should be available to all the agents belonging to the marketplace, if needed
R4	Reputation lifetime must be taken into account
R5	Agents should not be able to compute, or modify, their own reputation value
R6	Involved agents must use the same aggregator operator
R7	Reputation values must represent the evolution of the agent's behaviour
R8	New agents should not be penalized
R9	"Bad" agents should not be able to leave the marketplace, and re-join as different agents

Some of these requirements (R1, R5, R7 and R8) have been stated considering (Vavilis et al, 2014).

R1 will be fulfilled considering that each rating allows expressing a specific level of "satisfaction" for a particular interaction, while each local reputation score will be an indicator about the agent behaviour over time, from the trustor point of view. Regarding R2, it is necessary that each origin of reputation score can be identified (Vavilis et al, 2014), in order to check the reliability of their feedback, for including it or not in the final decision, and this will be more clear in section 5.1.2.

R4 has already been discussed previously, while for R6, the usage of the same aggregator operators by all the agents must be guaranteed, in order to provide coherency among different reputation values computed by different entities. When updating a certain local reputation value, the older ones are considered, in order to meet R7.

R8 is a very challenging requirement: a global default reputation value should be associated to a new user, allowing him to be trusted by other entities; however, this value should not be too high, because otherwise he could take advantage from this situation, and, obviously, "old" agents that built their reputation over time should not be penalized. R3, R5 and R9 will be discussed in section 5.1.2.

5.1.2 Blockchain, Trust and Reputation

COMPOSITION is relying on blockchain technologies as the central component of its log-oriented architecture. This technology will be used for implementing a secure, trusted and automated information exchange related to supply chain data. Considering the distributed nature of blockchain [Nak08] and, more in general, of the COMPOSITION infrastructure, it makes sense to rely on a distributed Reputation Model: each agent will compute his own reputation values, and will be in charge to provide these values to the other entities.

Actually, in the literature, there are some academic papers related to the usage of blockchain in trust management and authentication (Alexopoulos et al, 2017), (Moinet et al, 2017). However, considering the dynamic and distributed nature of blockchain, some interesting scenarios could be explored.

In (Hoffman et al, 2009), the three fundamental dimension of a generic Reputation Model have been identified: formulation, calculation and dissemination. So far, the last dimension, which includes also how reputation values are stored, has not been taken into account.

Blockchain could be really helpful in this case: the idea is to exploit this technology for storing local reputation values (R2, R3): in this way each agent would be aware when a reputation value given by a specific agent A to another agent B has been updated (R5). Then, he can choose if consider or not this new value, basing on his local reputation value related to agent A, when he should interact with agent B. The usage of blockchain will also help recognizing possible cheating behaviours, for instance a "bad" agent who tries to submit misleading reputation values on behalf of other agents, or re-join the marketplace for resetting his low reputation (R9). With the adoption of the blockchain, all the agents will have a global view of every interaction related to each agent of the marketplace.

For concluding this section, the one presented in this deliverable was a first idea about how we are planning to design the COMPOSITION Reputation Model and why it will be helpful in the context of the project.

As future plans, the choice of the most suitable aggregator operator will be done. It will be adapted for meeting all the requirements, especially reputation lifetime and feedback credibility, stated in section 5.1.1 and the first tests will be performed.

5.2 Digital signature

One of the cornerstones to increase trust in the content of the messages flowing in COMPOSITION is the inclusion of the digital signature on all messages. For that reason messages will be digitally signed using JWS[27] (JSON Web Signature) which is an IETF[28] proposed standard for signing arbitrary data.

JSON Web Signature (JWS) represents content secured with digital signatures or Message Authentication Codes (MACs) using JavaScript Object Notation (JSON) based data structures and base64url encoding. [3]

Terminology: (RFC-7515, 2015)

- **JWS Header:** A JSON Text Object (or JSON Text Objects, when using the JWS JSON Serialization) that describes the digital signature or MAC operation applied to create the JWS Signature value. The members of the JWS Header object(s) are Header Parameters.
- **JWS Payload:** The sequence of octets to be secured – a.k.a., the message. The payload can contain an arbitrary sequence of octets.
- **JWS Signature:** A sequence of octets containing the cryptographic material that ensures the integrity of the JWS Protected Header and the JWS Payload. The JWS Signature value is a digital signature or MAC value calculated over the JWS Signing Input using the parameters specified in the JWS Header.
- **Base64url encoding:** Similar to base64 encoding except for the use of non-reserved URL characters (e.g. – is used instead of + and _ is used instead of /) and the omission of padding characters.
- **Encoded JWS Header:** Base64url encoding of the JWS Protected Header.
- **Encoded JWS Payload:** Base64url encoding of the JWS Payload.
- **Encoded JWS Signature:** Base64url encoding of the JWS Signature.
- **JWS Signing Input:** The concatenation of the Encoded JWS Header, a period ('.') character, and the Encoded JWS Payload.
- **JWS Compact Serialization:** A representation of the JWS as the concatenation of the Encoded JWS Header, the Encoded JWS Payload, and the Encoded JWS Signature in that order, with the three strings being separated by two period ('.') characters. This representation is compact and URL-safe.

The representation consists of three parts: the JWS Header, the JWS Payload, and the JWS Signature. In the Compact Serialization, the three parts are base64url-encoded for transmission, and represented as the

[27] https://tools.ietf.org/html/rfc7515
[28] https://www.ietf.org/

concatenation of the encoded strings in that order, with the three strings being separated by two period ('.') characters (see Figure 24) (RFC-7515, 2015)

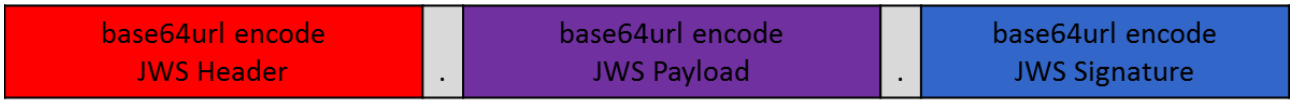

Figure 24 - JWS compact serialization

The JWS Header describes the signature or MAC method and parameters employed. The JWS Payload is the message content to be secured. The JWS Signature ensures the integrity of both the JWS Header and the JWS Payload. (RFC-7515, 2015)

Following an example of how to encode, decode and validate a JWS.

Encoding (RFC-7515, 2015)

The following example JWS Header declares that the encoded object is a JSON Web Token[29] (JWT) and the JWS Header and the JWS Payload are secured using the HMAC SHA-256 algorithm:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Base64url encoding the bytes of the UTF-8 representation of the JWS Header yields this Encoded JWS Header value:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
```

The following is an example of a JSON object that can be used as a JWS Payload. (Note that the payload can be any content, and need not be a representation of a JSON object.)

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

Base64url encoding the bytes of the UTF-8 representation of the JSON object yields the following Encoded JWS Payload:

```
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9
```

Computing the HMAC of the bytes of the ASCII representation of the JWS Secured Input (the concatenation of the Encoded JWS Header, a period ('.') character and the Encoded JWS Payload) with the HMAC SHA-256 algorithm using a key and base64url encoding the result yields this Encoded JWS Signature value:

```
TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

Concatenating these parts in the order Header.Payload.Signature with period ('.') characters between the parts yields this complete JWS representation:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.
TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

The JWS example explained before can be summarized in table below (Table 6)

Table 6 - JWS example

JWS Header	{
	"alg": "HS256",
	"typ": "JWT"
	}
		eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

[29] https://tools.ietf.org/html/rfc7519

JWS Payload	{
	"sub": "1234567890",
	"name": "John Doe",
	"admin": true
	}
		eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9

JWS Signature		TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.
TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

Decoding (RFC-7515, 2015)

Based on the previous example (Table 6) to decode JWS we need to follow the following steps:

1. Remove the base64url encoding from the Encoded JWS Header, the Encoded JWS Payload, and the Encoded JWS Signature.
2. Base64url decode the inputs and turn them into the corresponding byte arrays.
3. Translate the header input byte array containing UTF-8 encoded characters into the JWS Header string

Validation (RFC-7515, 2015)

After decoding JWS, the next logical step is the validation of the decoded JWS.

Since the alg parameter in the header is “HS256”, we validate the HMAC SHA-256 signature contained in the JWS Signature. If any of the validation steps fail, the JWS must be rejected.

1. Validate that the JWS Header string is legal JSON.
2. To validate the HMAC value, we repeat the previous process of using the correct key and the UTF-8 representation of the JWS Secured Input (which is the same as the ASCII representation) as input to the HMAC SHA-256 function and then taking the output and determining if it matches the JWS Signature.
3. If it matches exactly, the HMAC has been validated.

To perform all three processes described above: encoding, decoding and validation, libraries for token signing and verification can be found in: http://jwt.io/

5.3 Cryptographic Hash

Another level of trust will be added to the COMPOSITION platform with the calculation of the cryptographic hash of the messages flowing in the platform. A cryptographic hash is like the fingerprint of the data being hashed. A hash is a fixed length (length may vary depending on the hashing function used) string of characters that uniquely identifies the data being hashed, and has the peculiarity that the same hash value is obtained every time a hash is calculated over the same data and using the same cryptographic hash function.

It has to be clear that hashing is not encrypting; encryption is a two way function where data is encrypted with the purpose in mind of being decrypted in the future. Hashing, however, is not meant to be reversed. It is not a way to store data secured, but a way to easily compare two pieces of data, as a hash uniquely identifies pieces of data.

Hashing has an inherent problem, and that is collisions. A collision is when two different pieces of data produce exactly the same hash result. To prevent (or at least minimise) the collision problem, hash functions resulting on greater length hash values should be used. Some older hashing functions like MD5, which produce 128-bit hash values, should be avoided; it’s preferred the use of 256-bit or greater hashing functions.

One of the popular hashing functions nowadays is SHA which stands for Secure Hashing Algorithm. There are different types of SHA, being SHA-256 one of the most often used for common purposes today.

SHA are a family of cryptographic functions designed to keep data secured. It works by transforming the data using a hash function: an algorithm that consists of bitwise operations, modular additions, and compression functions. The hash function then produces a fixed size string that looks nothing like the original. These algorithms are designed to be one-way functions, meaning that once they're transformed into their respective hash values, it's virtually impossible to transform them back into the original data. A few algorithms of interest are SHA-1, SHA-2, and SHA-5, each of which was successively designed with increasingly stronger encryption in response to hacker attacks. SHA-0, for instance, is now obsolete due to the widely exposed vulnerabilities. (SHA, 2017)

6 Transport security

Internal communication between COMPOSITION components as well as external communication with other systems and/or users connecting to COMPOSITION platform shall be encrypted by using Transport Layer Security[30] (TLS). TLS is a cryptographic protocol that allows and guarantees the privacy and data integrity in the exchange of data between two communicating applications. (RCF-5246, 2008)

TLS is composed of two layers:

TLS Record Protocol (RCF-5246, 2008)

The TLS Record Protocol provides connection security that has two basic properties:

- The connection is private. Symmetric cryptography is used for data encryption.
- The connection is reliable.

TLS Handshake Protocol (RCF-5246, 2008)

TLS Handshake Protocol allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. The TLS Handshake Protocol provides connection security that has three basic properties:

- The peer's identity can be authenticated using asymmetric, or public key, cryptography.
- The negotiation of a shared secret is secure.
- The negotiation is reliable.

Current version of TLS protocol is 1.2, although the TLS 1.3 is in the works. It has not been finalized yet and is still a draft.

7 Next Steps

The next step will be to deploy the first prototype of the COMPOSITION Security Framework, which will implement the architecture proposed, as well as make use of components and technologies reported on this deliverable. The results of the Security Framework prototype will be reported in D4.4 Prototype of the Security Framework I due on M20.

The Reputation Model proposed in this deliverable will be refined and the results will be reported in the aforementioned deliverable D4.4.

8 Summary

This deliverable updates and complements what was reported back in M12 in D4.1 Design of Security Framework I. It offers a general view of the architecture of the COMPOSITION Security Framework as well as a description of the components and technologies that are part it.

It provides information on different alternatives to the architecture for some components of the Security Framework and reports on the developments that are actually active to provide access to the different

[30] https://www.ietf.org/rfc/rfc5246.txt

Security Framework services to other COMPOSITION components; or to customize already available services as is the case of the Authentication Service (Keycloak). It also reports on the different integration and deployment tasks that have taken place in this time.

The deliverable also provides detailed information on the proposed technologies to bring Integrity of data and Trust on data to COMPOSITION platform, as well as the proposed technology to secure communication among the different COMPOSITION components and with the outside world. It also contains the proposal for a Reputation Model to be implemented in the COMPOSITION platform. This Reputation Model will be refined in the upcoming deliverable in D4.4 Prototype of the Security Framework I due on M20.

9 List of Figures and Tables

9.1 Figures

9.2 Tables

10 References

(Netflow, 2007) Netflow Export Datagram Format. https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html

(COMPOSITION D4.1, 2017) D4.1 Design of Security Framework I.

(RMQ-Auth-Http, 2017) HTTP-based authorisation and authentication for RabbitMQ. https://github.com/rabbitmq/rabbitmq-auth-backend-http

(K-Mappers, 2018) Keycloak Protocol Mappers. http://www.keycloak.org/docs/latest/server_admin/index.html#_protocol-mappers

(K-SPI, 2018) Keycloak Service Provider Interfaces (SPI). http://www.keycloak.org/docs/latest/server_development/index.html#_providers

(OIDC, 2014) OpenID Connect specification. http://openid.net/specs/openid-connect-core-1_0-final.html

(RFC-1321, 1992) MD5 Message-Digest Algorithm. https://www.ietf.org/rfc/rfc1321.txt

(RFC-4634, 2006) US Secure Hash Algorithms (SHA and HMAC-SHA). https://tools.ietf.org/html/rfc4634

(RCF-5246, 2008) RCF-5246 The Transport Layer Security (TLS) Protocol Version 1.2. https://tools.ietf.org/html/rfc5246

(RFC-6151, 2011) Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms. https://tools.ietf.org/html/rfc6151

(RFC-6234, 2011) US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). https://tools.ietf.org/html/rfc6234

(RFC-6749, 2012) RFC-6749 The OAuth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749

(RFC-7515, 2015) RFC-7515 JSON Web Signature (JWS). https://tools.ietf.org/html/rfc7515

(RFC-7519, 2015) RFC-7519 JSON Web Token (JWT). https://tools.ietf.org/html/rfc7519

(SAML 2.0, 2013) SAML V2.0 specification. http://saml.xml.org/saml-specifications

(SHA, 2017) Secure Hashing Algorithms. https://brilliant.org/wiki/secure-hashing-algorithms/

(TLS 1.3, 2018) The Transport Layer Security (TLS) Protocol Version 1.3. https://tools.ietf.org/html/draft-ietf-tls-tls13-23

(XACML 3.0, 2013) eXtensible Access Control Markup Language (XACML) Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf

(Moyano et al, 2012) F. Moyano, C. Fernandez Gago and J. Lopez, “A Conceptual Framework for Trust Models,” in Trust, Privacy and Security in Digital Business, 2012.

(Jaydip Sen, 2010) J. Sen, “A Survey on Reputation and Trust-Based Systems for Wireless Communication Networks,” CoRR, vol. abs/1012.2529, 2010.

(Sabater et al, 2001) J. Sabater and C. Sierra, “Regret: A reputation model for gregarious societies,” Fourth workshop on deception fraud and trust in agent societies, vol. 70, pp. 61-69, 2001.

(Vavilis et al, 2014) S. Vavilis, M. Petković and N. Zannone, “A reference model for reputation systems,” Decision Support Systems, vol. 61, pp. 147-154, 2014.

(Ruohomaa et al, 2007) S. Ruohomaa, L. Kutvonen and E. Koutrouli, “Reputation management survey,” in Availability, Reliability and Security (ARES), 2007.

(Sabater et al, 2005) J. Sabater and C. Sierra, “Review on computational trust and reputation models,” Artificial intelligence review, vol. 24, no. 1, pp. 33-60, 2005.

(Gomez et al, 2011) F. Gomez Mármol and G. M. Pérez, “Trust and reputation models comparison,” Internet research, vol. 21, no. 2, pp. 138-153, 2011.

(Gutowska et al, 2009) A. Gutowska and A. Sloane, “Modelling the B2C Marketplace: Evaluation of a Reputation Metric for e-commerce,” in International Conference on Web Information Systems and Technologies, 2009.

(Nakamoto, 2008) S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

(Alexopoulos et al, 2017) N. Alexopoulos, J. Daubert, M. Mühlhäuser and S. M. Habib, “Beyond the Hype: On Using Blockchains in Trust Management for Authentication,” in Trustcom/BigDataSE/ICESS, 2017.

(Moinet et al, 2017) A. Moinet, B. Darties and J.-L. Baril, “Moinet, Axel, Benoît Darties, and Jean-Luc Baril,” in arXiv preprint arXiv:1706.01730, 2017.

(Hoffman et al, 2009) K. Hoffman, D. Zage and C. Nita-Rotaru, “A survey of attack and defense techniques for reputation systems,” ACM Computing Surveys (CSUR), vol. 42, no. 1, 2009.

(Jøsang et al, 2007) A. Jøsang, R. Ismail and C. Boyd, “A survey of trust and reputation systems for online service provision,” Decision Support Systems, vol. 43, no. 2, pp. 618-644, 2007.

(Artz et al, 2007) D. Artz and Y. Gil, “A survey of trust in computer science and the Semantic Web,” Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, no. 2, pp. 58-71, 2007.

(Sloman et al, 2000) T. Sloman and M. Grandison, “A survey of trust in internet applications,” IEEE Communications Surveys & Tutorials, vol. 3, no. 4, pp. 2-16, 2000.

(Ramana et al, 2010) K. Ramana, A. Chari and N.Kasiviswanth, “A Survey on Trust Management for Mobile ad Hoc Network,” International Journal of Network Security & Its Applications (IJNSA), vol. 2, no. 2, 2010.

(Torra, 2017) V. Torra, “Aggregation functions and information fusion. Modeling decisions,” 2017. http://www.mdai.cat/ifao/slides/transparencies.SFLA.2017.pdf

(Derakhshandeh et al, 2011) S. Derakhshandeh and N. Mikaeilvand, “Fuzzy Method for Identification of Aggregate Weights in Ordered Weighted Averaging Operators,” Middle-East Journal of Scientific Research, vol. 7(3), pp. 293-295, 2011.

(Torra et al, 2007) V. Torra and Y. Narukawa, Modeling Decisions: Information Fusion and Aggregation Operators, Springer-Verlag Berlin Heidelberg, 2007.

(Cornelis et al, 2010) C. Cornelis, P. Victor and E. Herrera-Viedma, “Ordered Weighted Averaging Approaches for Aggregating Gradual Trust and Distrust,” in XV Spanish Congress on Technologys and Fuzzy Logic ESTYLF, 2010.

(Ravana et al, 2009) S. D. Ravana and A. Moffat, “Score Aggregation Techniques in Retrieval Experimentation,” in Twentieth Australasian Database Conference, 2009.

(Resnick et al, 2002) P. Resnick and R. Zeckhouser, “Trust among strangers in internet transactions: Empirical analysis of eBay' s reputation system,” in The Economics of the Internet and E-commerce, Emerald Group Publishing Limited, 2002, pp. 127-157.

(Yan et al, 2008) Z. Yan and S. Holtmanns, “Trust Modeling and Management: from Social Trust to Digital Trust,” in Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions, 2008.