Collaborative Network for Industry, Manufacturing, Business and Logistics in Europe

D6.2

Design and Implementation of Security and Privacy for Core Business Services

Project Acronym	NIMBLE	
Project Title	Collaboration Network for Industry, Manufacturing, Business and Logistics in Europe	
Project Number	723810	
Work Package	WP6	
Lead Beneficiary	SRFG	
Editor	Violeta Damjanovic-Behrendt	SRFG
Reviewers	B. Mandler	IBM
	W. Behrendt	SRFG
Contributors		
Dissemination Level	PU, ©NIMBLE consortium	
Contractual Delivery Date	30/11/2017	
Actual Delivery Date	22/12/2017	
Version	V1.0	

Abstract

The NIMBLE project performs research leading to the design and the development of a cloud and IoT-based multisided platform, targeting supply chain relationships and logistics in the EU. Core platform functionality will enable firms to register on the platform, publish machine-readable catalogues of their products and services, search for suitable supply chain partners, negotiate contracts and supply logistics, and develop private and secure information exchange channels between firms.

The aim of the project is to support a federation of NIMBLE platform instances providing a set of core services, each specifically being tailored to a different aspect (regional, sectorial, topical, etc.). The overall role of the NIMBLE multisided platform in digital automation is to increase speed to market, minimize costs, optimize manufacturing and logistic processes. Such goals open several side effects related to cybersecurity, which could cause serious harm to the participating companies, e.g. losing customers, facing a host of legal and financial penalties, putting businesses at risk. Hence, our focus in Work Package WP6 of the NIMBLE project is to ***(i) meet baseline security and privacy standards, (ii) enforce policies and procedures to prevent infiltration, (iii) provide means to detect inappropriate access to connected products, and (iv) minimize any potential damage caused by unauthorized access.***

After identification, specification, analysis and evaluation of the security and privacy requirements in task T6.1 (see D6.1 for details), our focus in task T6.2 is to design and integrate the most important set of security and privacy controls for the core services of the NIMBLE platform. At the time of writing this report (month M14 of the project), the NIMBLE project is in its close-to-release-of-v1.0 stage, i.e. the demo platform implements the core functionality including: registration, publishing product catalogue, searching through the product catalogue, and initiating simple business processes for buyers, sellers, and logistics. Hence, the security controls presented in this report emphasize the importance of mutual authentication (possibly, two factor authentication (password and PIN)) to secure the network against impersonating services and servers; encrypted communication to prevent illegitimate access to resources and provide data integration; implementation of access controls through access policies and identification mechanisms through tokens that store attributes; and design of security mechanisms that establish trust relationships. Finally, the privacy of the user's data must be preserved in compliance with the General Data Protection Regulation (GDPR), which will apply from 25 May 2018. The mapping between GDPR requirements and the platform-centric security and privacy requirements is given in D6.1, Appendix 1. The new GDPR is expected to have significant implications on businesses in the EU, and hence we design our privacy mechanisms to comply with it.

Document History

Version	Date	Comments
V0.1	30/05/2017	Initial version and assignments distribution
V0.2	31/08/2017	Section 2 finalized
V0.3	01/10/2017	Section 2 finalized
V0.4	30/10/2017	Basic security controls integrated in the demo
V0.5	30/11/2017	Future steps defined and document finalized
V0.6	05/12/2017	Version ready for internal review
V0.7	19/12/2017	Final QA comments
V0.8	21/12/2017	QA comments addressed
V1.0	22/12/2017	Consolidated final version (upload by coordinator on 02/01/2018)

Table of Contents

List of Figures

List of Tables

Glossary

GDPR (General Data Protection Regulation)	*The new regulation by which the European Parliament, the Council of the EU and the European Commission intend to unify data protection for all individuals within the EU. It will come into effect in May 2018.*
Confidentiality	*A set of rules that limits access or places restrictions on certain types of information.*
Data Integrity	*Security methods for assuring accuracy and consistency of data over its entire life-cycle. It is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data.*
Data Availability	*Security methods ensuring that services and data are functional and available when they are needed.*
Identification	*Assigning a responsible party (user) for taking an action.*
Authentication	*Comparing user identification means in order to prove his/her right to take on a role, or prove possession of attributes required for taking an action.*
Authorization	*Comparing user rights with the defined access policy.*
Policy enforcement	*Steps to enforce authorization decisions.*
STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege)	*STRIDE is an approach to threat modelling and requirement evaluation, developed by Microsoft.*

1 Introduction

This deliverable presents major security decisions, security design principles, implementation and integration aspects of *Information Security* in the NIMBLE project, specifically targeting core business services developed in the release V1.0 of the NIMBLE platform. Future platform releases require additional security controls, more security testing and monitoring (see Appendix 1, *NIMBLE Cybersecurity Plan,* for more details on future steps in WP6).

The major *Information Security* terms used in this document refer to the following:

- **Identification**: assigning a responsible party (user) for taking an action;
- **Authentication**: comparing user identification means in order to prove his/her right to take on a role, or prove possession of attributes required for taking an action;
- **Authorization**: comparing user rights with the defined access policy;
- **Access decision**: combining the above three methods in order to decide whether or not a user request should be honoured.

In deliverable D6.1 *"Security and Privacy Requirements"*, we identified federated identity management and access controls as one of the important security requirement in the project. The NIMBLE project has an up-front requirement to be designed in federated fashion, in order to enable collaboration with another 9 FoF (Factories of the Future) projects coordinated by the *ConnectedFactories* project (see: http://www.effra.eu/connectedfactories). Before designing federated identity management and access controls features for core NIMBLE services, we analysed several issues related to the traditional access control models that are summarized in [KAHD09]:

- **Achieving mutual agreements** on the meaning of user roles, credentials and attributes, which becomes difficult in complex, cross-domain applications;
- **Role explosion effect** when addressing differences in rights between different domains;
- **Excess authority** bringing the security risks of giving control of all the user's rights to the programmer and the data provider;
- **Delegation and revocation of subset of rights** (based on metadata), which leads to better security and auditability;
- **Avoiding transitive access problems**, by introducing the authorization and the concept of least privilege.

Based on the analysis of traditional access control models in Section 2.1, we suggest the **authoriZation Based Access Control (ZBAC) model** to be used as a reference access control model in NIMBLE, with the potential to reduce the number and scope of cross-domain agreements leading to management overhead, and reduce confused deputy[1] attacks through the enforcement of least privilege. For more details about ZBAC, please refer to section 3.3.

1.1 Document Organization

Section 2 gives an overview of the state-of-the-art methods and tools for implementing *Information Security* controls that have been previously identified in D6.1 to be of interest in NIMBLE, e.g. methods and tools for implementing access control management, identity management, cryptography and data integrity. Section 3 discusses authorization service life

[1] The **confused deputy** [HARD88] problem amounts to a Trojan horse attack. Both attacks relate to the

cycle in NIMBLE, which combines models such as Role Based Access Control (RBAC), Attribute Based Access Control (ABAC) and Authorization based Access Control (ZBAC). Section 4 presents the current state of security controls implementation in NIMBE. Section 5 describes future steps related to planned improvements of identity and access control management, future steps to support data integrity and data quality management (in tasks T6.4), and privacy mechanisms for the platform. Appendix 1 describes the *NIMBLE Cybersecurity Plan,* which shows the relation between WP6 tasks and activities, and the NIMBLE platform releases.

2 State-of-the-Art Methods and Tools for Implementing Information Security Controls

Access control models and techniques for controlling the use of resources and services in systems is fundamental to security. Apart from access control, designing and implementing secure systems includes data integrity, non-repudiation, privacy, etc. In NIMBLE, the entire work package WP6 is focused on designing, integrating and implementing security features of the platform. Task T6.2 specifically designs and integrates security features for the core security services, those that have been integrated and released with the very first version of the NIMBLE platform (release V1.0, month M15). Security features in V1.0 need to provide reasonable access controls for resources and services, which are bound to identity, authentication and authorization mechanisms.

In this section, we summarize state-of-the-art models and tools for identity and access control management, and make an observation on models and tools in cryptography and data integrity.

Please note that the state-of-the-art in privacy related methods and techniques is covered in D6.1, section 2.3 where we derived the *NIMBLE Privacy Framework* that also includes GDPR. However, these will only be implemented in later releases of NIMBLE and are still being designed (T6.3 and T6.4).

2.1 Access Control Models and Tools

2.1.1 Access Control Models

Access control models encompass traditional and some non- traditional (fine-grained) models:

- Traditional access control models are user-centric, e.g. Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role Based Access Control (RBAC) and Group Based Access Control (GBAC);
- Non-traditional access control models take into account some additional parameters, e.g. resource information, relationship between the user and the resource, and dynamic information, such as context: time, location, user address.

2.1.1.1 Traditional Access Control Models

Access control models vary from *Mandatory Access Control (MAC)* to more flexible, *Discretionary Access Control (DAC)*.

IDENTIFICATION BASED ACCESS CONTROL (IBAC). The IBAC model was one of the first access models that was based on an access control matrix for storing user permissions. IBAC did not include a specification of permissions for changing its entries, which was left to a trusted party, e.g. the system administrator. The administrator had rights to access all resources and services within the system, which with the rising number of users of distributed systems became untenable. At the same time, users could have many identities and often had to authenticate in different ways on different systems, which led to work to consolidate access control systems with Federated Identity Management (FIdM) and Single Sign-On/ Single Log-Out (SSO/ SLO).

MANDATORY ACCESS CONTROL (MAC). In the MAC model, which is a subcategory of IBAC, the system administrator gives permissions for subjects to access a specific object, by assigning security labels to both subject and object [SAND93]. The MAC model is restrictive in the sense that only the system administrator can modify security labels of objects. The MAC system needs to be isolated from the operating system, for maintaining the security policies and preventing unauthorized access, which makes these models difficult and expensive to implement and maintain [SOC14].

DISCRETIONARY ACCESS CONTROL (DAC). Unlike in MAC, where permissions are given through predefined policies by the administrator, in the DAC models, permissions are given by users which decide the access rights to the resources they own, by using the Access Control Lists (ACL) [MOST90]. Each entry in the ACL gives users (or group of users) permissions to access resources. DAC is broadly adopted by current operating systems based on UNIX, FreeBSD, and Windows [SOC14].

ROLE BASED ACCESS CONTROL (RBAC) - GROUP BASED ACCESS CONTROL (GBAC). Models such as Role Based Access Control (RBAC) and Group Based Access Control (GBAC) feature a centralized enforcement of a group and role management to ensure that security policies cannot be circumvented by adding entities to groups or changing roles. For example, in RBAC, users are assigned to roles, which are maintained in a centralized way, and the security policies grant rights to roles rather than to users [SCFY96]. Since the users are associated to the roles, a user can access certain resources and perform specific tasks.

The granting of rights and policy enforcement are carried out by the administrator and users cannot transfer permissions connected to their role to other users. Although RBAC is useful for handling system-wide policies in which certain groups or roles are allowed to perform certain operations, the model is not effective in use cases in which users allow other users to access their data without requiring a particular role to be defined by the system administrator. Access policies defined in terms of groups may be enough in cases where access policies are fairly simple, while the increase in the number of users and groups brings more complexity to the configuration of policies. Reaching agreement with all partners on what rights to associate with a role, as well as a mutual understanding of the meaning of defined roles, has proved to be difficult. Furthermore, RBAC had limited support for context, such as day vs. night, or war vs. peace, when such distinctions were important in the access decision [KAHD09].

2.1.1.2 Non-Traditional Access Control Models

ATTRIBUTE BASED ACCESS CONTROL (ABAC). To provide more fine-grained access mechanisms, the authors in [NIST-ABAC14] proposed the ABAC model. The authorization decisions in ABAC are based on attributes that the user has to prove (e.g., age, location, roles, etc.) as well as resources and environmental properties. The ABAC model allows specification of policies in terms of attributes that belong to a subject, an object, an action performed by the subject in an object, or the environment [HU13].

Every attribute defined in an ABAC model, consists of a key-value pair such as "*Role = Manager*". The privileges are granted to users through the usage of policies that combine attributes altogether. Everyone must agree on a set of attributes and their meaning when using ABAC, which is not easy to accomplish [SOC14].

The ABAC model is sometimes called - Policy Based Access Control (PBAC) [BLFI99] [PIKI06] or Claims Based Access Control (CBAC) [BROW07].

AUTHORIZATION BASED ACCESS CONTROL (ZBAC). Unlike ABAC and RBAC systems, in which the user submits an authentication along with the service request, the user in the ZBAC model submits authorization credentials along with a request for making an access control decision. With ZBAC, user's access to resources is based on authentication on the user's domain before the request is made. In other words, **ZBAC model requires that the user authenticate in order to know which authorizations to grant.** Such an approach requires agreements between the involved domains to trust each other. As a result, users obtain authorizations, which can be represented by cryptographically bound credentials or assertions. Then, the target service or its **Policy Decision Point (PDP)** verifies the validity of the authorization to make an access decision. This is a valuable feature for IoT scenarios in which constrained devices could interact with each other, since interactions with third parties are not required for each communication [SOC14].

ZBAC deals with authorization in distributed systems addressing issues like federated identity management, Single Sign-On (SSO), violations of least privilege, problems with updating rights, assigning responsibility, and coordinating with partners in information sharing. In NIMBLE, ZBAC enables federated identity management and prevents the role explosion and confused deputy effects. The authors in [KAHD09] summarize several additional arguments in favour of the ZBAC model:

- It simplifies systems and user management tasks,
- It is significantly more tolerant of intermittency,
- It supports background prepositioning of some security credentials and authorizations,
- It allows chaining of systems while enforcing least privilege, and
- It can accommodate low bandwidth constraints.

2.1.2 Standards, Languages and Tools for Access Management

Access control management tools include various technologies and tools for Centralized Authentication (CA), SSO, session management and authorization enforcement for applications in multiple use cases [KRSI17]. In addition to core functionality, which is about access features, these technologies and tools brings solutions to several security-related requirements of the NIMBLE platform, e.g. password reset, Enterprise Mobility Management (EMM), identity administration (user profile updates, self-service registration), identity synchronization to a limited set of target systems, etc.

In the following, we present the state-of-the-art technologies and tools that support federated access control management, such as OpenID and OAuth2 standards, languages like SAML and XACML, and tools like U-Prove and Keycloak. In NIMBLE, the design and implementation of federated identity and access control management is done in Keycloack, which supports OAuth2, XACML and SAML (for more information: https://github.com/keycloak/keycloak/tree/master/examples/saml).

2.1.2.1 OpenID Standard

OpenID[2] is an open standard and decentralized authentication protocol. It allows users to be authenticated using *Relying Parties* (RP) (cooperating websites) while avoiding a central authority for the user's authentication. OpenID also allows users to log into multiple websites using shared credentials (identity and password).

[2] http://openid.net/

OpenID provides a framework for the communication between the OpenID *Identity Provider* and the OpenID *Acceptor*, i.e. RP. The user receives an OpenID *identifier* via his OpenID *Identity Provider*. This identifier takes the form of a unique Uniform Resource Identifier (URI), when trying to get access to a specific RP. In the next step, the RP visits this URI in order to reach the proper OpenID *Identity Provider* and to authenticate the requesting user.

One of the main advantages of OpenID is that it does not force users to use a specific authentication mechanism, allowing for login/password or smart cards approaches to be applied. As a result, as of March 2016, there were over 1 billion OpenID accounts and a large number of organizations combining OpenID with their own authentication management. Today, OpenID is a widely deployed technology by companies such as Google, Yahoo, PayPal, IBM, BBC, and many more. The current version of OpenID is known as the OpenID Connect standard [SBJM11] (built upon OAuth2), which is controlled by the OpenID Foundation (see: http://openid.net/connect/).

2.1.2.2 OAuth2 Standard

OAuth2[3] is the industry standard for authorization, used for web and desktop applications, mobile phones and home automation and IoT devices. It is the evolution of the original OAuth protocol, created in 2006 for the purpose of enabling third party applications and clients to get access to resources and services, originally owned by the user, without a need to give user's credentials to these applications and services [HARD12]. Through the OAuth2 authorization framework, an application can obtain limited access to a resource, either by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing a third-party application to obtain access on its own behalf [SOC14].

The OAuth2 authorization approach has two main advantages. First, it solves the problem of trust between a user and third-party applications. In OAuth2, the user can allow applications to collect data and perform tasks on their behalf, without giving the authentication credentials to such applications. Second, OAuth2 allows a service provider to give third-party applications the possibility to expand their services with applications which enforce the secure use of data.

While OAuth2 is widely used by different services on the Web, it does not meet the requirements of IoT-based scenarios, where devices can interact with each other, avoiding the access to central authorization services for checking the access rights every time.

2.1.2.3 SAML

SAML[4] (Security Assertion Markup Language) is an XML-based framework for describing and exchanging security information between entities. Such security information is expressed using SAML statements for the identity, attributes and authorizations of a subject. SAML defines the syntax and a set of rules for the creation, communication and use of its statements. SAML information exchanges usually take place between an Asserting Party (AP) and a Relying Party (RP): An AP is an entity that creates and communicates SAML statements or constructs, while the RP uses these statements to perform a specific task [OASIS-SAML][SAML].

SAML defines three types of assertion:

- Authentication Assertion, that contains information about the user's authentication;
- Attribute Assertion, that transfers the user's attributes;

[3] https://oauth.net/2

[4] https://wiki.oasis-open.org/security/FrontPage

- Authorization Decision Assertion, that includes authorization decision statements (e.g., permit, deny).

In order to support a typical multi-domain SSO scenario, SAML defines the roles called Identity Provider (IdP) and Service Provider (SP). The SAML specification defines the structure and contents of the following four main elements:

- Assertions, with information about identity, authentication and authorization of a particular subject. Assertion can be used in different contexts.
- Protocols, defining the way in which assertions can be requested and answered.
- Bindings, used to define how protocol messages are transported by employing other lower layer protocols, such as HTTP.
- Profiles, which are composed of protocols, bindings and SAML assertions. A profile can be considered as a set of elements and interactions between them, which are needed to realize a specific SAML use case.

2.1.2.4 XACML

XACML (eXtensible Access Control Markup Language) is an OASIS standard defining an access control policy language and an access control decision request/response language (both XML-based) [OASIS-XACML] (for more, see http://docs.oasis-open.org/xacml/3.0). The policy language is used to describe general access control requirements, and has standard extension points for defining new functions, data types, etc., while the request/response language is used to create a query for checking if a given action should be performed or not. The response can have one of four values: Permit, Deny, Indeterminate (if an error occurred or some required value was missing, so a decision cannot be made) or Not Applicable (the request can't be answered by this service).

The XACML architecture consists of the following four elements (see Figure 1):

- **PEP** (Policy Enforcement Point): when the user wants to take some action on a resource, the PEP forms a request based on the user's attributes, the resource in question, the action, and other information pertaining to the request;
- **PDP** (Policy Decision Point): after creating a request, the PEP will send this request to a PDP, which evaluates the request and policies applying to this request, in order to make an authorization decision (if access should be granted or not). That answer will be returned to the PEP which can then allow or deny access to the requester;
- **PAP** (Policy Administration Point): it is used to create a policy or a set of policies;
- **PIP** (Policy Information Point): it acts as a source of attribute values.

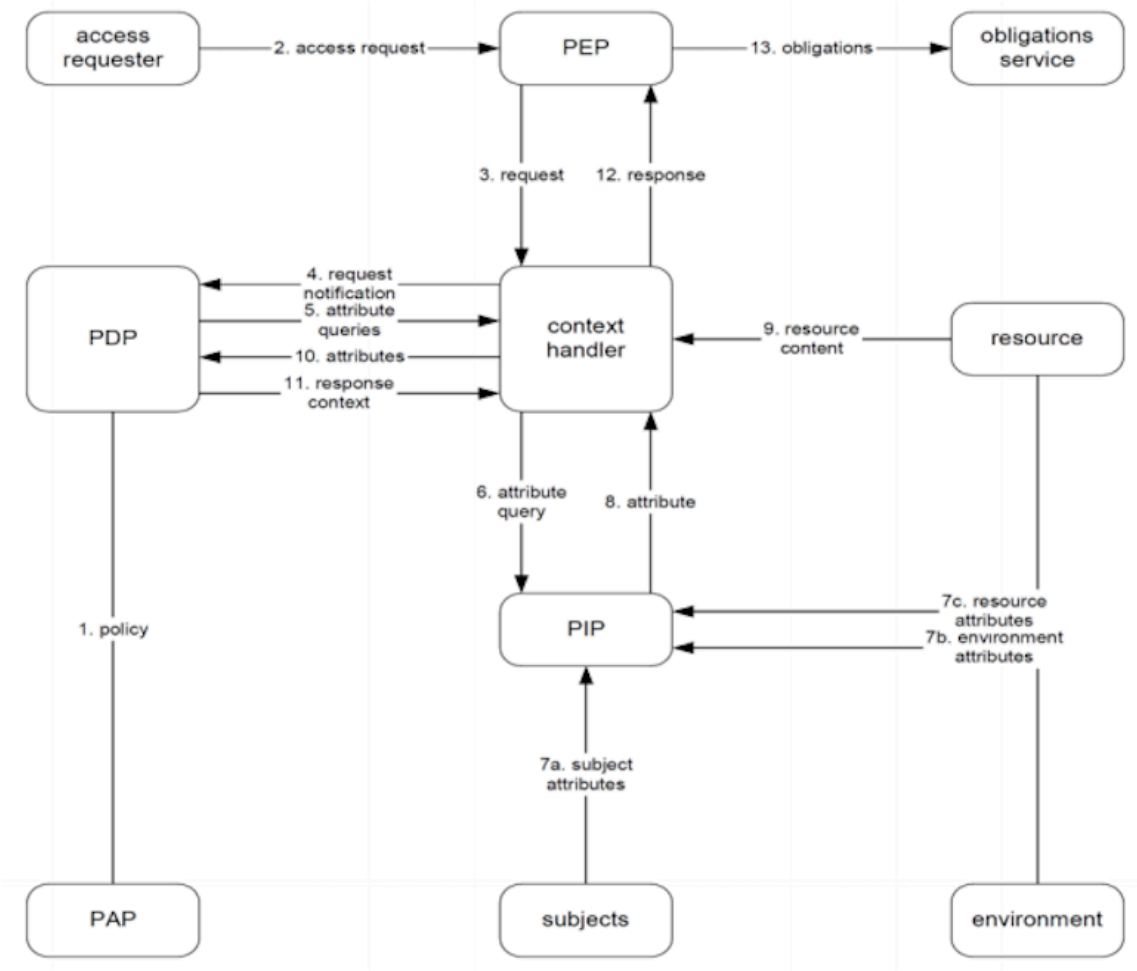

Figure 1: XACML Components: PDP, PAP, PIP, and PEP

Figure 2 shows the XACML Policy Language Model. It includes the following elements:

- **A PolicySet.** At the root of all XACML policies is a Policy or a PolicySet. A PolicySet is a container that can hold other Policies or PolicySets, as well as references to policies found in remote locations.
- **A Policy.** It represents a single access control policy, expressed through a set of Rules. Each XACML policy document contains exactly one Policy or PolicySet root XML tag.
- **A Rule.** Because a Policy or PolicySet may contain multiple policies or Rules, each of which may evaluate to different access control decisions, XACML needs a way of reconciling the decisions each makes. This is done through a collection of Combining Algorithms, i.e. algorithms combining multiple decisions into a single decision. Combining Algorithms are used to build up complex policies.
    - There are Policy Combining Algorithms (used by PolicySet) and Rule Combining Algorithms (used by Policy). An example of these is the Deny Overrides Algorithm, which says that no matter what, if any evaluation returns Deny, or no evaluation permits, then the final result is also Deny.

In addition to the above three elements of the XACML data model, there is also a feature called **a Target**, which is a set of simplified conditions for the Subject, Resource and Action that must be met for a PolicySet, Policy or Rule to apply to a given request. Rules have **a Condition**, which is a boolean function of either Permit or Deny logical values. If the Condition evaluates

to true, then the Rule's Effect (a value of Permit or Deny) is returned. Evaluation of a Condition can also result in an error (Indeterminate) or discovery that the Condition doesn't apply to the request (NotApplicable).

Attributes. Attributes are named values of known types that may include an issuer identifier or an issue date and time. A user's name, their security clearance, the file they want to access, and the time of day are all attribute values. When a request is sent from a PEP to a PDP, that request is formed almost exclusively of attributes, and they will be compared to attribute values in a policy to make the access decisions. **Attribute Values** can be resolved by a Policy through two mechanisms:

- **the AttributeDesignator** lets the policy specify an attribute with a given name and type, and then the PDP will look for that value in the request, or elsewhere if no matching values can be found in the request.
- **the AttributeSelector** allows a policy to look for attribute values through an XPath query. A data type and an XPath expression are provided, and these can be used to resolve some set of values either in the request document or elsewhere.

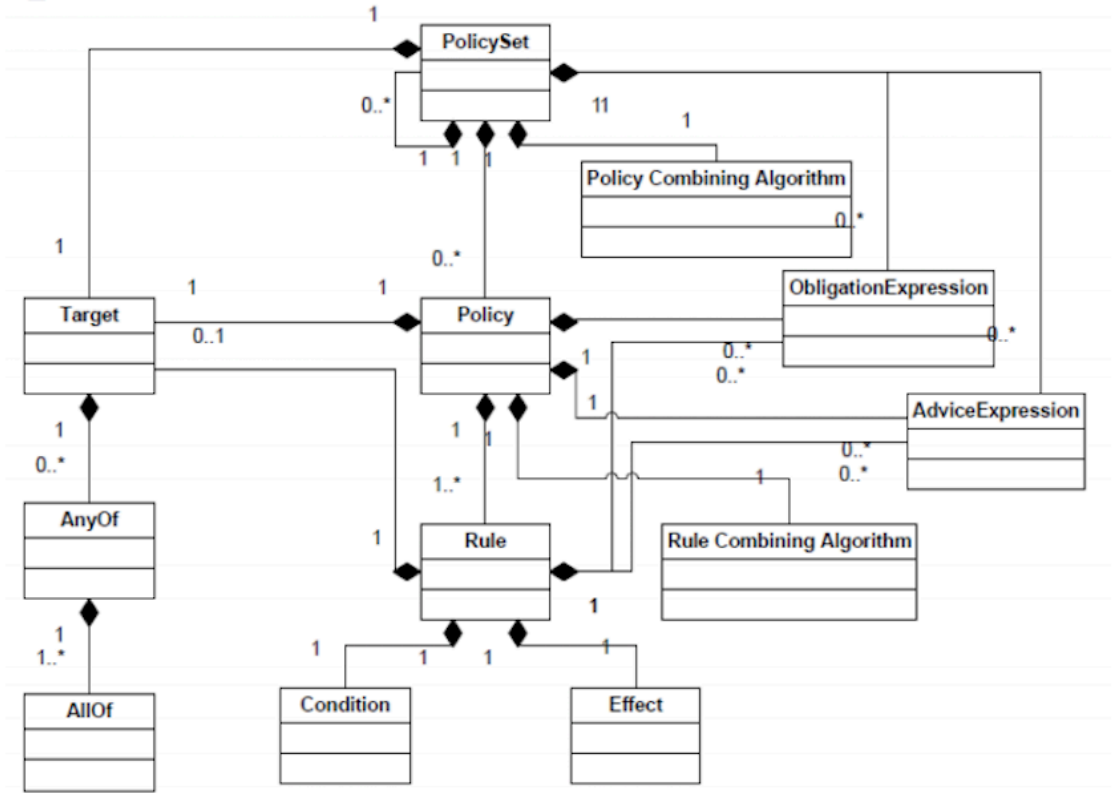

Figure 2: XACML Policy Language Model

2.1.2.5 U-Prove Tool

Unlike traditional identity management technologies, anonymous credential systems such as U-Prove, allow for selective disclosure of personal identity information in order to protect user's privacy. They allow for the use of cryptographic proofs, which can be derived from identity credentials (claims-based identity) without the need to set the whole credential protocol, as required for certificates-based solutions. U-Prove is based on the use of U-Prove tokens as attribute information containers, which are generated and delivered by an *Issuer* to a *Prover* [PAZA13].

The communication between an *Issuer* and a *Prover* is carried out through an issuance protocol, while a presentation protocol is used to enable the communication between a *Prover* and a *Verifier*. Because of the use of the public key and the signature encoded in the token, the issuance and presentation of a token is unlinkable. This prevents unwanted tracking of users when they use their U-Prove tokens. For presenting the token to a *Verifier*, a specifically created message can be used to prevent replay attacks. The U-Prove token, the presentation proof, and the message can be kept in an audit log for later verification.

The use of a U-Prove token does not reveal its private key, which ensures that the token cannot be stolen through eavesdropping or phishing, and prevents unauthorized replay by introducing legitimate *Verifiers*. Finally, many presentation proofs or signatures may be created with the same U-Prove token.

2.1.2.6 Keycloak Tool

Keycloak[5] is an open source identity and access control management tool that offers some modern features such as User Federation, Identity Brokering and Social Login that is designed to simplify logins for end users and enable signing into a third party website using existing information from a social networking services (Facebook, Twitter, Google+, WordPress, etc.).

Keycloak is based on OAuth2 and supports fine-grained authorization policies and combines a variety of access control mechanisms, e.g. ABAC, RBAC, User-Based Access Control (UBAC), Context-Based Access Control (CBAC), Rule-Based Access Control based on JavaScript and Jboss Drools, Time-Based Access Control and access control mechanisms through a Policy Provider Service Provider Interface (SPI) [Keycloak]. It enables the creation of permissions for the use of resources, associates these permissions with authorization policies, and enforces authorization decisions in applications and services. The most basic authorization decisions in Keycloak follow the RBAC model, which is also associated with a few limitations [Keycloak]:

- Changes to roles can impact multiple resource (roles and resources are tightly coupled);
- Changes to security requirements can imply changes to application code and roles to reflect these changes;
- Role management might become difficult and error-prone in complex systems;
- Roles lack contextual information which is a must in distributed and heterogeneous environments (e.g. users are distributed across different regions, with different local policies, using different devices).

Keycloak Authorization Services improve the authorization capabilities of applications and services by providing:

- Resource protection using fine-grained authorization policies and different access control mechanisms;
- Centralized Resource, Permission, and Policy Management
- Centralized Policy Decision Point (PDP)
- REST-based authorization services
- Authorization workflows and User-Managed Access
- The infrastructure to help avoid code replication across projects (and redeploys) and quickly adapt to changes in security requirements.

Below are some of the most important concepts used by the Keycloak Authorization Services (see Figure 3):

- **Resource Server.** It is the server hosting the protected resources that is also capable of accepting and responding to protected resource requests. It relies on a decision if access

[5] http://www.keycloak.org/index.html

should be granted to a protected resource, which is usually obtained from **a security token** that is sent on every request to the RESTful-based resource server. For web applications, that decision is stored in a user's session and retrieved from there for each request. In Keycloak, any **confidential** client application can act as a resource server. This client's resources and their respective scopes are protected and governed by a set of authorization policies.

- **Resource.** It can be a set of one or more endpoints, a classic web resource such as an HTML page, etc. In authorization policy terminology, a resource is the *object* being protected. Every resource has a unique identifier that can represent a single resource or a set of resources.
- **Scope.** It is a bounded extent of access that is possible on a resource. It indicates what can be done with a given resource. Examples of scopes are view, edit, delete, etc. However, a scope can also be related to specific information provided by a resource; for example, a cost scope, which is used to define specific policies and permissions for users to access a project's cost.
- **Permission.** It associates the object (resource) being protected with the policies that must be evaluated to determine whether access is granted. For example,

  ```
  X [user, roles, groups] CAN DO Y [an action to be performed] ON RESOURCE Z [a protected resource]
  ```

- **Policy.** A policy defines the conditions that must be followed to grant access to a resource. Policies relate to the different access control mechanisms that can be used to protect resources. Keycloak enables so called aggregated policies, and creating individual policies, that can be reused with different permissions and build complex policies by combining individual policies.
- **Policy Provider.** It is an implementation of specific policy types. Keycloak provides a SPI (Service Provider Interface) that is used to plug in a specific policy provider implementation.
- **Permission Ticket.** A permission ticket represents resources and / or scopes being requested by a client as well as the policies that must be applied to a request for authorization data (Requesting Party Token (RPT)). It is a type of token defined by OAuth2's User Managed Access (UMA). In UMA, permission tickets are crucial to support person-to-person sharing and also person-to-organization sharing. Using permission tickets for authorization workflows enables a range of scenarios from simple to complex, where resource owners and resource servers have complete control over their resources based on fine-grained policies that govern the access to these resources.

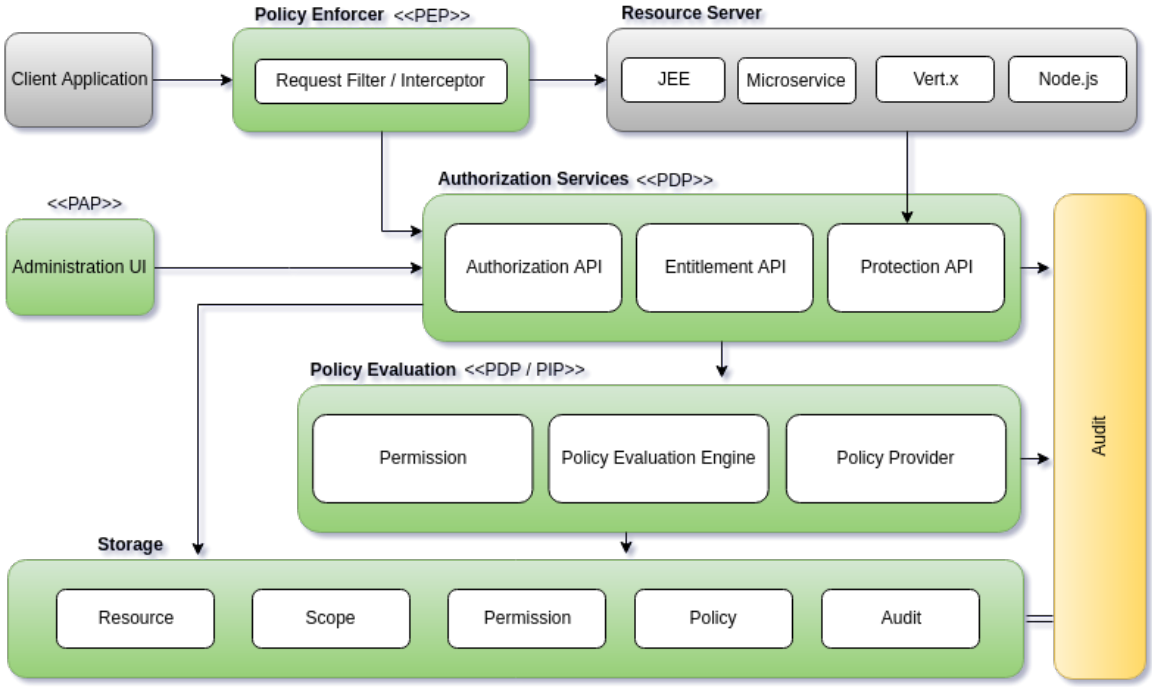

Figure 3: Keycloak architecture

2.2 Identity Management Methods and Tools

The core process of gaining access to various resources and services is based on the Identification - Authentication – Authorization life cycle. The concept of Identity Management in the IoT extends from the users and services to IoT devices and things, in which the authentication proof for devices can be obtained from ownership or identity relationships of devices with an owner, administrator, user or even group of stakeholders. Identity management models differ between: core and temporary identities, identity of a group of objects, identity based on owner's identity or on specific features (e.g. quantity, ingredients). Another classification of identity models refers to user-centric, device-centric and hybrid identity management.

2.2.1 Authentication Methods and Tools

Authentication is an identity agreement between communicating parties. It can be performed using various authentication methods, e.g. passwords, two-factor authentication (password plus one-time unique code), biometric authentication (face recognition, voice recognition, fingerprint, etc.), gesture based authentication (keypad gestures, free form gestures, etc.).

The main characteristics of several authentication approaches are summarized below [DECI13]:

- **Service authentication based on combination of username and password**. This is the most commonly used authentication method in web services and mobile applications. It is simple to implement, but has several weaknesses. For example, simple passwords can be broken by brute-force methods, while complex passwords require a password management system.
- **Federated web authentication based on a combination of username and password.** Federated identity refers to the linking of the user's electronic identity across different services. It enables the user to be authenticated by one service and access resources at another service (SSO). This approach has become widely used in the context of social media services (Facebook, Google, Microsoft), which use username/ password authentication. It is also called Social Login. It is easy to implement and is user-friendly (same password can be used to access many services). However, this is not considered to be a strong authentication method.
- **Challenge question/response method.** This method requires the user to answer a question, which the user has previously defined. The response is typically related to the user's personal life and is therefore easy to memorize for the user. The method provides additional security to other methods, but is not secure enough as the sole method of authentication.
- **One-time passwords method.** It is widely used in online services with high security demand e.g. in online banking. In this method, the password is continuously changed and can be provided to the user through multiple channels, e.g. Short Message Service (SMS), or by sending a list of passwords to the user by ordinary mail.
- **Smartcard Public Key Infrastructure (PKI) authentication.** The PKI is an arrangement for binding the public key of a user with the user's identity by means of a certificate authority. Typically, the user's certificate and private key are installed on a smart card.
- **Mobile (SIM-card) PKI authentication.** This method stores the private key and certificate in a Subscriber Identity Module (SIM) card, which also provides a secure execution and storage environment. The SIM-card resides in a mobile phone which provides the required "smart card reader functionality". Some mobile phones include so called NearField Communications (NFC) functionality, which enables mobile phones to connect with specific NFC enabled smart cards. The method is attractive for mobile apps with high security demands.
- **Biometric authentication.** It uses a scan of a fingerprint, face, iris, voice or other characteristic of the user for identification. These methods are user-friendly, but considered as not being secure enough as a sole method - an additional pin-code or password is needed. Biometric identifiers are not anonymous and they cannot be revoked.
- **Device authentication.** It provides an additional security dimension to authentication using some specific solutions included in the device hardware, e.g. the Intel Identity Protection Technology, or SecurityKey. This method can be used to make basic person-level authentication more reliable. The main disadvantage is that hardware based solutions are not available on all platforms.

There is also the **two-factor authentication** that requires a usual static password, which counts as the first factor, while the second factor could be either a one-time password PIN-code, or biometric factors (voice, retina, fingerprint), smart cards, certificates with a digital signature or authentication via text messages. In the IoT, the classic authentication mechanisms with passwords are confirmed to be ineffective, e.g. biometric factors are not reliable when used remotely. Some recent work in this area recommends security tokens to be complemented with a stronger authentication that combines multiple factors, e.g. the context and the environment of

the authentication process, use case specific factors, internal machine IDs, etc. (known as **context-based authentication**) [FRIE15].

2.2.2 Authorization Techniques, Methods and Tools

Authorization of users, groups of users or user's devices to get access to specific resources is the final response of the system, allowing users/ groups/ devices to perform certain actions. Authorization services are usually based on a well-defined set of authorization patterns, which include the following elements:

- **Policy Administration Point (PAP).** It provides a set of UIs for managing resource servers, resources, scopes, permissions, and policies.
- **Policy Decision Point (PDP).** It provides a point to where authorization requests are sent and policies are evaluated accordingly with the permissions being requested.
- **Policy Enforcement Point (PEP).** It provides implementations for different environments to enforce authorization decisions at the resource server side. For example, Keycloak provides some built-in Policy Enforcers.
- **Policy Information Point (PIP).** It is based on Keycloak Authentication Server, and can be used to obtain attributes from identities and runtime environment during the evaluation of authorization policies.

2.2.2.1 Authorization Patterns in Keycloak

In this section, we describe two authorization patterns in Keycloak: Authorization Process and the Authorization Services pattern. The Keycloak identity and access control management tool is used in NIMBLE to define basic security controls for the core business services.

AUTHORIZATION PROCESS. To enable fine-grained authorization of applications, Keycloak requires the following three main processes to be performed [Keycloak]:

- Resource Management (see Figure 4);
- Permission and Policy Management (see Figure 5);
- Policy Enforcement (see Figure 6).

Resource Management starts with the specification of the Resource Server (a web application to be protected, or a set of services) (see Figure 4). The Resource Server is managed using the Keycloak Administration Console. A Resource to be protected can be a web page, a RESTFul resource, a file in a file system, etc. It can be a single and specific resource, or a group of resources. As with the Resource Server, Resources can be also managed using the Keycloak Administration Console or the Protection API (enables remote management of resources). Finally, a Scope needs to be created. Scopes represent the actions that can be performed on a resource, or can be used to represent one or more attributes within a resource.

Figure 4: Resource Management in Keycloak

Permission and Policy Management. Next step after creating the Resource Server and defining Resources and Scopes, is to set up Permissions and Policies (see Figure 5), which define the security and access requirements that govern resources.

Figure 5: Permission and Policy Management in Keycloak

Policy Enforcement. It involves steps to enforce authorization decisions to a Resource Server (see Figure 6). In Keycloak, this is achieved by enabling a PEP at the Resource Server to communicate with the Keycloak Authorization Server, ask for authorization data and control access to protected resources based on the decisions and permissions returned by the server. Keycloak provides some built-in Policy Enforcement implementations that can be used to protect applications depending on the platform they are running on.

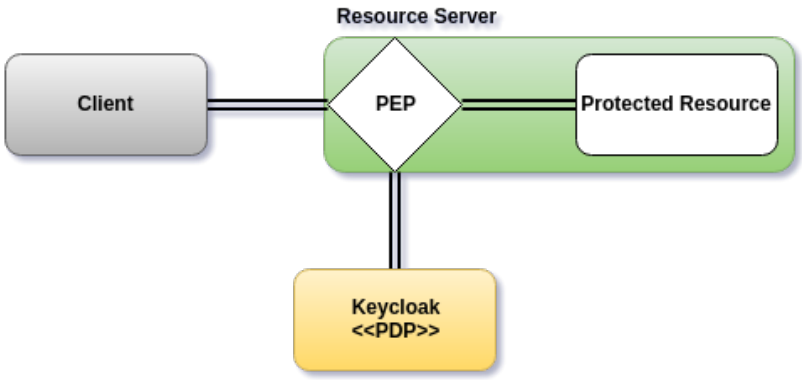

Figure 6: Policy Enforcement in Keycloak

AUTHORIZATION SERVICES. They consist of the following RESTFul APIs covering specific steps in the authorization process:

- Protection API;
- Authorization API;
- Entitlement API.

The Protection API. It is an UMA-compliant endpoint providing a small set of operations for resource servers to help with the management of resources and scopes. The operations provided by the Protection API can be organized in two main groups:

- Resource Management: Create Resource; Delete Resource; Find by ID; Find all; Find with filters (e.g., search by name, type, or URI);
- Permission Management: Issue Permission Tickets, which confirm that the permissions have been requested by the client and sent to the server to obtain a final token. The final token contains all permissions granted during the evaluation of the permissions and policies that are associated with the resources and requested scopes.

The Authorization API. It is also a UMA-compliant endpoint providing a single operation that exchanges an Access Token and Permission Ticket with a Requesting Party Token (RPT). The RPT contains all permissions granted to a client and can be used to call a

resource server to get access to its protected resources. Requesting an RPT can also provide a previously issued RPT. In this case, the resulting RPT will consist of the union of the permissions from the previous RPT and the new ones within a permission ticket (Figure 7).

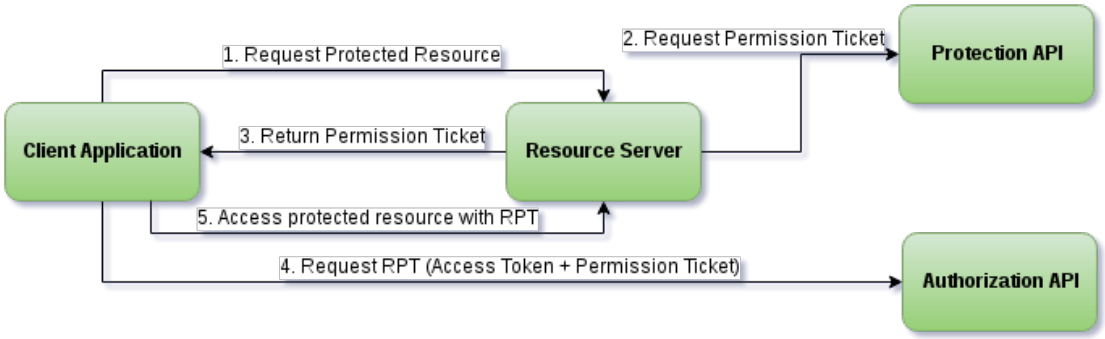

Figure 7: Authorization Services in Keycloak

The Entitlement API. It provides a protocol to issue RPTs. Unlike the Authorization API, the Entitlement API expects only an access token. From this API, all the entitlements or permissions for a user (based on the resources managed by a given resource server) or just the entitlements for a set of one or more resources can be obtained (see Figure 8).

Figure 8: Entailment Service in Keycloak

2.2.2.2 Authorization Protocols in OAuth2

OAuth2 is a standard for authorisation based on HTTP [RFC6750]. It solves issues that arise when users want to grant specific permissions to an application, that are either limited in time or require limited functionality. To solve issues related to the user distributed password management (e.g. changing distributed credentials, revoking permissions, etc.), OAuth2 defines several roles [RFC6750]:

- **Resource Owner.** An entity capable of granting access to a protected resource. When the resource owner is a person, he/she is referred to as an end-user.
- **Client.** The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.
- **Resource Server.** An application making protected resource requests on behalf of the resource owner and with its authorisation.
- **Authorisation Server.** The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorisation.

TYPES OF AUTHORIZATION GRANTING PROCESSES. OAuth2 differentiates between the following four types of authorization grants: Authorization Code, Implicit Authorization,

Authorization based on Resource Owner and Password Credentials, and Authorization based on Client Credentials. Each of these methods is described below which illustrates the communication between a Resource Owner – a Client – and a Resource Server. A Resource Owner sends an authorization grant to a specific service (Client) to perform some action that includes resources which are hosted by the Resource Server.

Figure 9: OAuth2 communication between Resource Server - Client - Resource Owner

The entities required to implement the OAuth2 authorization process are as follows [RFC6749]:

- **Authorisation Endpoint.** It is used by the Client to obtain authorization from the Resource Owner via a user-agent redirection.
- **Token Endpoint.** It is used by the Client to exchange an authorisation grant for an access token, typically with Client authentication.
- **Client Endpoint.** It is used by the Authorisation Server to return responses containing authorisation credentials to the Client via the Resource Owner user-agent.

The authorization grant process involves different endpoints and offers different kinds of security protection.

Authorisation Code. In this method, the Resource Owner and the Client are authenticated before the Client receives the token. This approach enables confidential communication between the entities and is considered to be the most resilient approach to various types of attacks. It requires a confidential channel in order to prevent eavesdroppers from using tokens or authorization codes maliciously. It requires the Authorization Endpoint (for user authentication and authorization) and the Token Endpoint (with an access token that allows for the action to be performed) (see Figure 10). After obtaining the Access Token (and a Refresh Token), the Client sends a request to the Resource Server to check the scope of the token and verify permissions granted by the user (as shown in Figure 9).

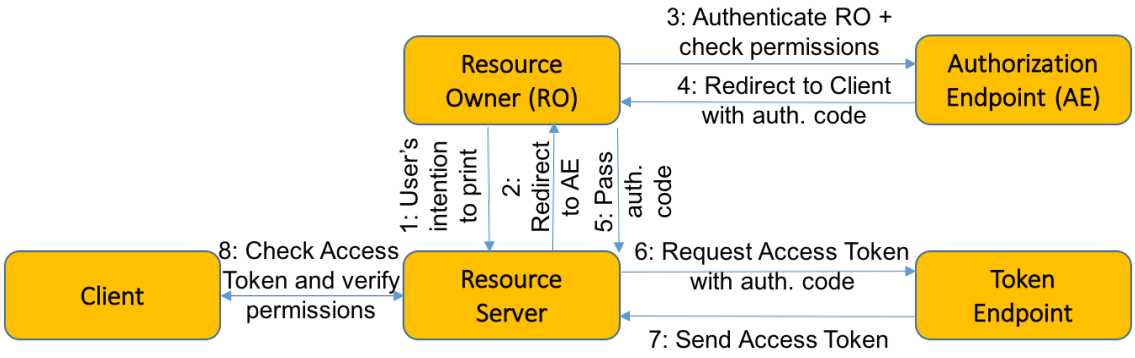

Figure 10: Authorization Code method in OAuth2

Implicit Authorisation. This method only authenticates the Resource Owner, but not the Client. It requires fewer steps than the Authorization Code method: the Token Endpoint has been excluded here, which introduces some risks as the Resource Owner can impersonate the Client, or the HTTP referral can be exposed to third parties. In its

final step, the Client sends a request to the Resource Server to check the scope of the token and verify permissions granted by the user.

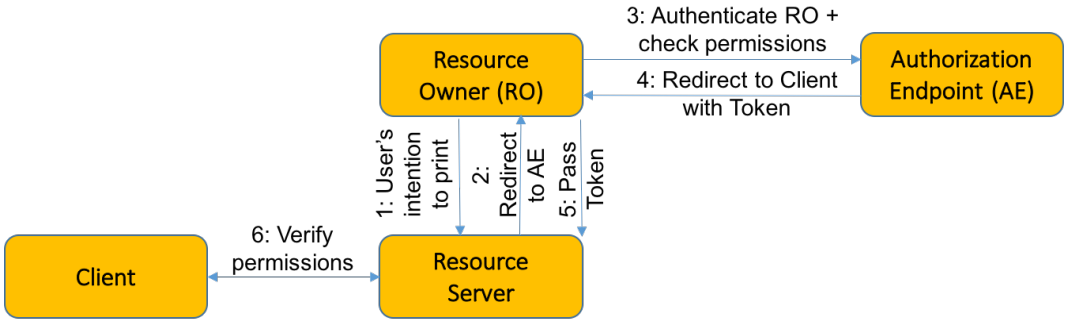

Figure 11: Implicit Authorization method in OAuth2

Authorisation based on Resource Owner and password credentials. This method requires the Client to get the access credentials from the Resource Owner, and then the Client exchanges these credentials for a token (from the Authorization Endpoint). Practically, this method includes even fewer steps than the above two methods, as the Resource Owner is not directly involved in the authorization granting procedure. In its final step, the Client sends a request to the Resource Server to check the scope of the token and verify permissions granted by the user.

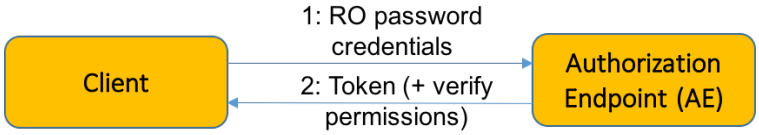

Figure 12: Authorization based on Resource Owner and Password Credentials in OAuth2

Authorisation based on Client credentials. This method assumes that there is a previous negotiation phase of credentials between the Client and the entity granting tokens. In its final step, the Client sends a request to the Resource Server to check the scope of the token and verify permissions granted by the user.

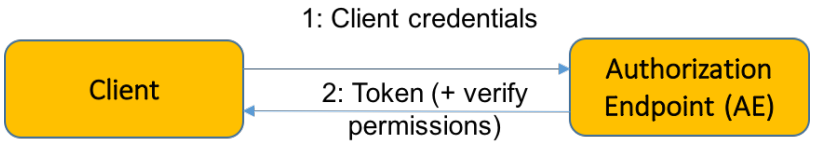

Figure 13: Authorization based on Client credentials in OAuth2

2.3 Cryptography and Data Integrity Techniques and Tools

2.3.1 Cryptography Techniques and Tools

Cryptography covers various aspects of Information Security, such as data confidentiality, data integrity, data provenance, and authentication. Applications of cryptography are many, including ATM cards, computer passwords, electronic commerce, etc. Some well-known approaches used to enable cryptographic techniques are: Public Key Cryptography, Secure and Fast Encryption Runtime (SAFER), Symmetric Key Cryptography, Hash functions and Key Exchange Algorithms [AHSM12] [OWASP-CRYPTO].

Public Key Cryptography. This technique requires two separate keys: the first is used to lock (encrypt) the plaintext, and the second one to unlock (decrypt) the cyphertext. One of these keys is always public, while the other one is kept private. This approach uses asymmetric key algorithms; hence it is often referred to as - **asymmetric key cryptography**. By publishing the public key, the key producer empowers anyone with a copy of the public key to produce messages that s/he can read. The key producer has a copy of the private key, which is required for decryption. When someone wants to send a secure message to the creator of those keys, the sender encrypts it using the recipient public key. To decrypt the message, the recipient uses his private key. The asymmetric key algorithms are based on mathematical relationships that have no efficient solution (e.g. the integer factorization and discrete logarithm problems).

Secure and Fast Encryption Runtime (SAFER). SAFER is the name of a family of block ciphers (algorithms) operating on fixed-length groups of bits and using a symmetric key. SAFER also refers to a family of block ciphers designed by James Massey, which are available for unrestricted use [MAKK00]; for example, SAFER K-64 (designed in 1993), SAFER K-128, SAFER SK-40, SAFER+ (designed in 1998), SAFER++ (designed in 2000).

Symmetric Key Cryptography. The modern symmetric-key ciphers relate to the study of block ciphers, stream ciphers and their applications [AHSM12].

- A blockciphers take as input a block of plaintext and a key, while the output is a block of cipher text of the same size. For example, the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are block cipher designs, which have been designated as cryptography standards by the US government [FIPS]. DES is used across a wide range of applications, e.g. ATM encryption, e-mail privacy, and secure remote access.
- Stream ciphers create an arbitrarily long stream of key material in a stream cipher, the output stream is created based on a hidden internal state which changes as the cipher operates. That internal state is initially set up using the secret key material.

Hashes. Hash functions take data of an arbitrary length and generate a fixed-length hash. Some common hashing algorithms are MD5 and SHA-1, both considered weak in terms of their security properties. New applications are encouraged to migrate to SHA-256, which implements a larger key size.

Key Exchange Algorithms. Lastly, key exchange algorithms, such as Diffie-Hellman for SSL. allow for safe exchange of encryption keys with an unknown party.

2.3.2 Data Integrity and Data Availability

Both data integrity and data availability are becoming major concerns for cloud data owners.

DATA INTEGRITY. Data integrity is about protecting data from unauthorised modifications and use by third parties. There exist some standards and workflows to achieve accuracy, consistency and trustworthiness in the entire data lifecycle. For example, the **Transport Layer Security (TLS)** is used to ensure data integrity on a communication level. **Asymmetric cryptography** is applied to encrypt messages that need to be exchanged, while integrity is checked using a **Hash Message Authentication Code (HMAC)**. **HTTPS** is built on top of TLS and is the underlying technology for RESTful communication between services. Still, integrity on a data store level is a major challenge, due to highly diversified technologies used in system architectures. In order to keep track of changes in separate storage systems in a microservices architecture (such as the NIMBLE architecture), there are tools that can perform Master Data Management (MDM). MDM tools examine every database created under a specific reference, e.g. IDs of the user or device creating an entry to a data store. MDM operates in the background, searching for solutions to fix inconsistencies.

Approaches for Data Integrity Verification. In cloud systems, the data owner needs to support one of the following approaches for data integrity verification [JEKU16]:

- **Reed–Solomon code.** Widely used in large cloud storage systems to correct burst errors that are associated with media defects. These codes are an important group of error-correcting codes.
- **Checksums.** A count of the number of bits in a cloud data outsourcing process unit that can be controlled by the clients. If the total bit count matches, then it is assumed that the process is not affected by any integrity problem.
- **Message Authentication Code (MAC).** In cloud security, MAC is a small section of information used to authenticate outsourced data. It also verifies sender authenticity, which achieves data integrity.
- **Digital Signatures.** A popular approach used to authenticate and verify the client for outsourced digital data.

Furthermore, methods used to provide data integrity verification in the cloud may be based either on deterministic or probabilistic protocols [JEKU16]. In **deterministic data integrity verification protocols**, the examiner checks all data blocks of the outsourced file at each server. This protocol is not suitable for big data environments. In **probabilistic data integrity verification protocols,** the verifier checks the integrity of random subsets of the data chunks. This is widely used in cases where the clients have large and dynamic datasets.

DATA AVAILABILITY. The system must be protected against downtimes that could be caused by malicious attackers. Therefore, threat intelligence and measures against Denial-Of-Service (DoS) attacks and network intrusion should be regularly practiced. For example, the system should be up and running even if a single microservices becomes unreachable, and fall-back mechanisms must be activated. **Netflix Hystrix** (https://github.com/Netflix/Hystrix) provides fault tolerant mechanisms and avoids failure propagation in the system. Netflix Hystrix is part of the Spring Cloud technology stack and can therefore be easily integrated into any existing technology stack. Intrusion detection systems mainly perform log analysis and policy monitoring in order to detect unauthorised intruders in the system.

3 Authorization Service Life Cycle in NIMBLE

Security controls are of the utmost importance in NIMBLE and they range from identification and authentication services, usage controls and access policies, to trusted and secure information sharing, data integrity and data quality management. Identity and access control management in NIMBLE are implemented using the Keycloak tool, in a way that combines role-based, attribute-based and authorization-based access controls. The Keycloak tool is explained in Section 2.1.6.2 and our major decision to use this tool relates to its support of not only role-based access control features, but also attributes and authorization through tokens.

The first version of the authorization service life cycle in NIMBLE has been designed during the evaluation phase of security requirements in task T6.1 (see D6.1, Section 6, for more details). The evaluation of the security requirements in T6.1 was based on STRIDE threats and vulnerability analysis [SHOS14], and the creation of DFDs (Data Flow Diagrams) of core business services covered the following platform functionalities: user registration, user login, search for products, publishing new products/catalogues and negotiating some product features. In T6.2, we use DFDs to extract roles, attributes and authorization features and policies, which are necessary to implement the identity and access control management in NIMBLE. In the following, we summarize the roles, attributes and authorization features and policies in NIMBLE.

3.1 Role Based Access Controls (RBAC) in NIMBLE

The initial set of roles identified in NIMBLE is presented in Figure 14. Here we differentiate between three scopes of user roles, namely platform, person and company. User roles relate to their specific scope; they are authorized by another role in the platform and define specific permissions. Table 1 describes user roles, their scopes, authorization features and permissions in more details.

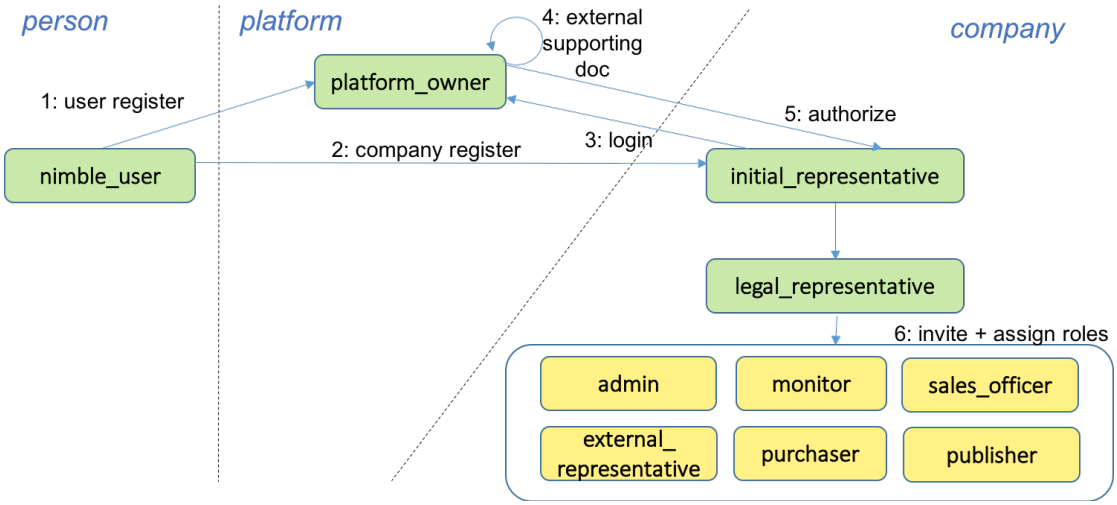

Figure 14: User Roles in NIMBLE

Table 1: User Roles in NIMBLE

Scope	User Role	User Role is Authorized By ...	User Role is Allowed to Do ...
platform	*platform_owner*	keycloak / NIMBLE admin at start-up	platform access control management; review trust and risk levels of companies; disable companies involved in suspicious activities; review transaction statistics; change the governance rules of the platform
person	*nimble_user*	platform manager	limited search of the product catalogue (cannot publish products nor perform any business negotiation); can register a company
company	*initial_representative*	platform manager	temporarily set up a company on the platform
company	*legal_representative*	Initial representative + external supporting documentation	change master data of the company; invite other company members to join the platform; assign further user roles
company	*external_representative*	legal representative + external supporting documentation	change master data of the company; assign further user roles; act as (purchaser or sales officer or publisher) for company
company	*admin*	legal representative	assign further user roles as specified by legal representative; (purchaser | sales officer | publisher); set monitoring permissions for data to be exchanged with third parties
company	*purchaser*	legal representative | admin	initiate contractually valid purchases

company	*sales officer*	legal representative | admin	initiate contractually valid sales
company	*publisher*	legal representative | admin	change the presentation of the company on the platform
company	*monitor*	legal representative | admin	monitor certain processes that are visible on the platform

A person can register on the NIMBLE platform by providing minimal personal information (first name, last name, email address). After a successful registration, this person is given the role ***nimble_user***, which enables limited search functionality, while publishing and negotiating features are not available. This person can also register a company, which is the first step towards accessing NIMBLE's B2B platform functionality.

The initial presence of a company on the platform is started by the registered user who initiates the company registration process. This person supplies company details and a telephone number where messages (e.g. TANs) can be received. This person is given the role ***initial_representative*** for a specified time, during which the ***platform_owner*** checks the validity of the provided company details. The suggested time frame is 4 weeks. If no supporting evidence is received within that time period, the company is marked as *"initial presence period has expired"* and runs the risk of being removed from the platform. If supporting evidence is received in time, the *platform_owner* is obliged to verify the documentation, possibly ask for further evidence, and - if all is well - grants the company the status "NIMBLE-approved", with trust level "unknown" and risk level "unknown" which is the starting position of any new company on the platform.

The full presence of a company on the NIMBLE platform must be started by appointing a legal representative for that firm. That person has to first register with NIMBLE and then supply physical written evidence that they are who they claim to be and that they have the right to be the legal representative for the company. (This is similar to the LEAR role in the EU participant portal). That person is assigned the role ***legal_representative***.

Once the company is approved, the legal representative can replace the preliminary information about the company with full information about products, services, terms and conditions. The legal representative can appoint further people with specific roles as described above (Figure 14, Table 1).

3.2 Attribute Based Access Controls (ABAC) in NIMBLE

One special case that we foresee for SMEs in NIMBLE is for an external representative to manage the presence of one or more companies on the platform, acting on behalf of these companies. Such a person is effectively the legal representative for these firms on the NIMBLE platform, and can be associated to different countries and their country specific laws and

policies. Therefore, the access control model in NIMBLE extends from RBAC and leverages the ABAC model and token-based authorization techniques.

In ABAC, an attribute may refer either to a user or to a resource or to the environment. An attribute is a particular property, role or permission associated with a component in the system [SCIA17][HUFE14]. It is assigned after an authentication procedure by the system administrator (*platform_manager* in NIMBLE). The Keycloak tool which is used to implement identity and access control management methods in NIMBLE, enables storing arbitrary user attributes (Attributes tab in Keycloak enables entering in the attribute name and its value).

The mapping between attributes (e.g. in different platforms, different countries) is to be defined as part of the NIMBLE policies, which will be designed as an **attribute mapping function**. For example, the login functionality of the NIMBLE platform presented in D6.1 (see Figure 15), requires from the user to enter the login parameters in the login form. Login procedure starts with the authentication step, in which the user's account parameters are first checked in the *Accounts DB* (checking if the user is registered on the platform), and second, the user identity is checked in the *Log record DB*. If the user account is found in the *Accounts DB* and the user identity exists in the *Log record DB*, the authorization step is successfully finished. The final identity request is sent to the *Login Processing service*, which performs the login of the user.

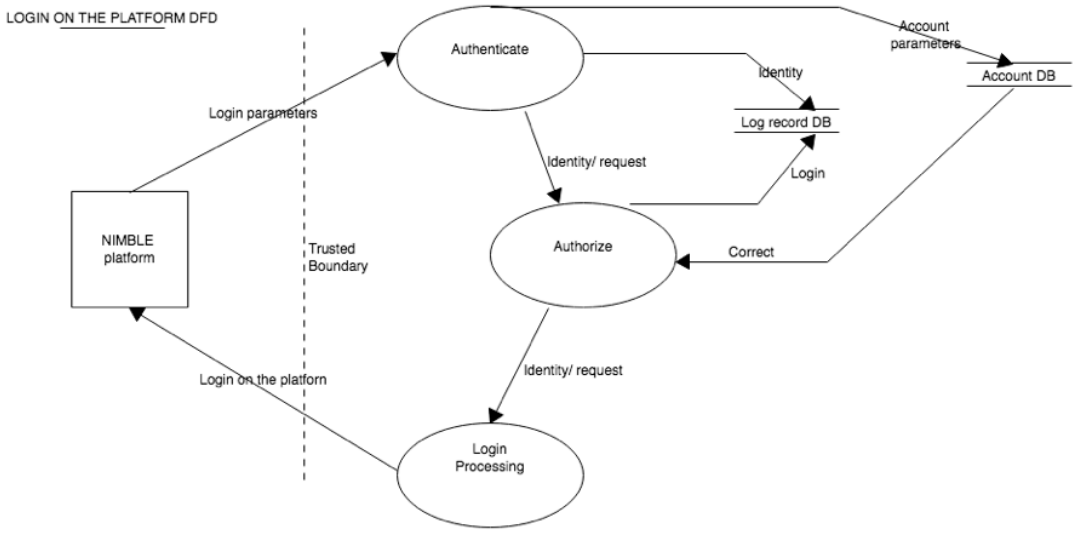

Figure 15: Login DFD

Adding the ABAC model to the above login procedure introduces an addition step between of core authentication and resource access authorization, as shown in the sequence diagram presented in Figure 16.

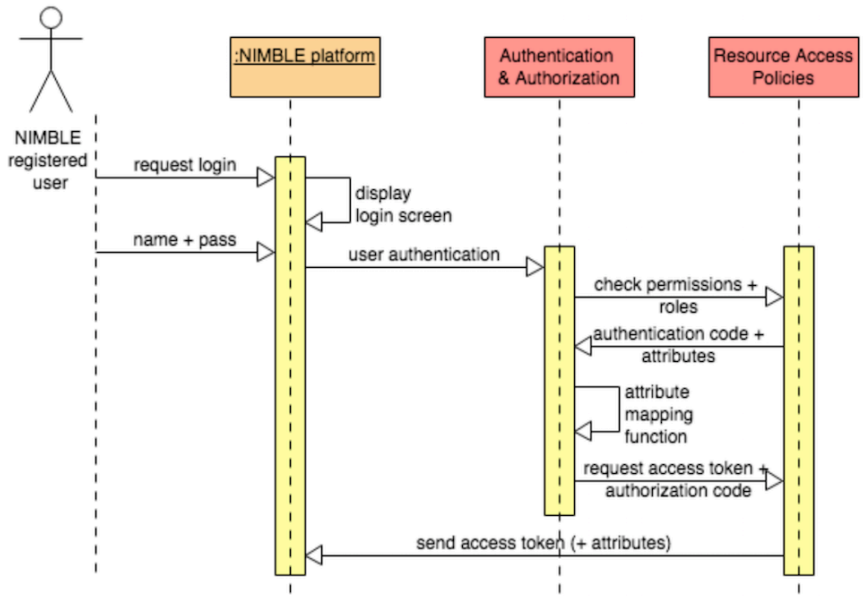

Figure 16: Login procedure with ABAC

After the core user authentication (e.g. authentication procedure checking user permissions and roles), the Resource Access Policies Endpoint sends back the authentication code with attributes to the Authentication and Authorization Endpoint. An **attribute mapping function** checks for the validity of interlinks and attribute based policy definitions, before a request for authorization code and access token is issued. Finally, an access token (with attributes) is sent to the NIMBLE platform.

The access token in NIMBLE is created using the JSON (JavaScript Object Notation) Web Token (JWT). JWT is an open industry standard for the implementation of authentication and authorization mechanisms on the Web. It contains a set of certified statements (claims) with the following structure:

- a header - contains information about the type of the token and the crypto algorithm used to build the sign of the token),
- a body - contains a set of claims for a specific token, and
- a sign - contains cryptographic validation of the token.

An example of JWT token in NIMBLE is shown in Figure 23.

3.3 Authorization Based Access Controls (ZBAC) in NIMBLE

Unlike the ABAC and RBAC systems, users in the ZBAC model access resources based on their authentication in the user's domain, before the access request is made [KADH09]. Such an approach requires agreements between the involved domains to trust each other. As a result, users obtain authorizations, while the target service (or its PDP) verifies the validity of the authorization to make an access decision.

ZBAC reduces the number and scope of cross-domain agreements. By combining designation with authorization, it eliminates misunderstanding that leads to confused deputy attacks, and eliminates the need to manage users from other domains while simplifying the enforcement of

least privilege. Implementing ZBAC require some changes to the underlying system. Figure 17 illustrates which parts of the service life cycle are handled differently in systems adopting the ZBAC models. The ZBAC implementation is analysed based on service life cycle example with ZBAC, discussed in [KADH09].

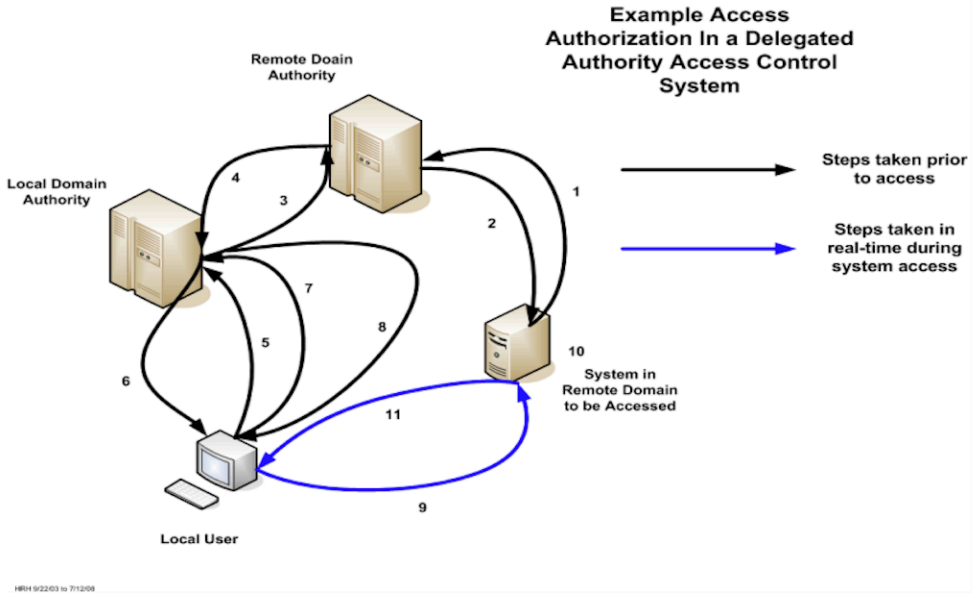

Figure 17: Federated Identity Management and Service Lifecycle with ZBAC

Figure 17 illustrates the steps to be taken prior to the user accessing the system, plus steps taken in real-time, during the access to the system (steps 9 and 11). The system presented in Figure 17 supports federated identity management and those situations in which the users are not known to the service domain. Dealing with this requires managing multiple users' IDs and multiple authentication mechanisms. In the ZBAC model, users are managed only in their own domains. In addition, it requires a **mutual understanding of the meaning of roles**. A role in one domain can be associated with nearly the same rights as the corresponding role in another domain, but not exactly the same. Introducing new roles to cover the discrepancies inevitably leads towards the problem known as role explosion. With ZBAC, the set of authorizations being applied to a specific task is an intersection of authorization and policy vectors. Attributes, such as country or citizenship can be also included in ZABC authorization mechanisms.

The steps required to adopt the ZBAC model, shown in Figure 17, are as follows:

1. The Remote Domain System accesses the Remote Domain Authority in order **to register itself** (by providing identity information and attributes).
2. The Remote Domain Authority issues approval and credentials in the form of **a certificate signed by the Remote Domain Authority's private key**. Such a certificate contains identity information of the Remote Domain System.
3. The Local Domain Authority registers with the Remote Domain Authority and **agrees to specific governance rules for the Remote Domain System**. Here the Remote Domain Authority can be seen as the root of trust for all services under its control.
4. The Remote Domain Authority issues approval and credentials in the form of **an authorization assertion/credential** permitting the Local Domain Authority to issue rights delegated to it for accessing the Remote Domain System cryptographically bound to the Remote Domain Authority's private key, and the Local Domain Authority's public key. This authorization credential may also include terms of use, expiration rules, and other governance restrictions.

5. The Local User **registers** user attributes with the Local Domain Authority.
6. The Local Domain Authority issues an **identity credential** that is cryptographically bound to the Local Domain Authority's private key and the Local User's public key.
7. The Local User **requests access** to the Remote Domain System from the Local Domain Authority.
8. By following relevant governance guidance, the Local Domain Authority issues a **delegation credential** to the Local User encoding the user's rights, signed with its private key and containing the public key of the Local User. The authorization credential issued by the Remote Domain Authority to the Local Domain Authority as proof of the right to delegate is also sent in systems that have not pre-cached it.
9. The Local User **accesses** the Remote Domain System with a transaction signed by the Local User's private key, which may contain the delegation credential issued by the Local Domain Authority as signed by the Local Domain Authority's private key and the authorization credential which was issued to the Local Domain Authority and signed by the Remote Domain Authority.
10. The Remote Domain System performs several validation checks in order to verify the **authorization credential** issued by the Remote Domain Authority, the **delegation credential** issued by the Local Domain Authority, the **signature on the whole transaction** using the public key contained in the delegation credential, and it also validates the **assertion of rights** by the Local User against the policy vector encoded in the delegation credential, and **format and content of the transaction against local policy.**
11. **Transaction is executed**. The Remote Domain System signs it with its private key. The **transaction can be verified** using the Remote Domain System's public key that is contained in the authorization credential.

4 Current State of Security Controls Integration and Implementation in NIMBLE

At the time of writing this report (month M14 of the project duration), the first demo of the NIMBLE platform was released featuring some core platform functionalities, e.g. user registration, user login, company registration, searching product catalogues, publishing product catalogues, sending invitation to other company members to join the platform, and performing some basic business processes such as ordering some goods and invoking a logistics service between a supplier and a manufacturer.

The implementation of the Information Security controls in NIMBLE, such as user identification, authentication, authorization and access controls, is managed by Keycloak. The basic security controls *in* NIMBLE are described below.

USER REGISTRATION. A new user must register on the platform by providing minimum information, e.g. email address, password, first and last name (see Figure 18). The platform demo executes some basic form and field controls, e.g. validity of email format. More validity controls will be added with the following platform releases. After providing basic user details and clicking the Submit button, the user is registered. Note the message at the bottom of the registration form (Figure 18): "*Registration successful*".

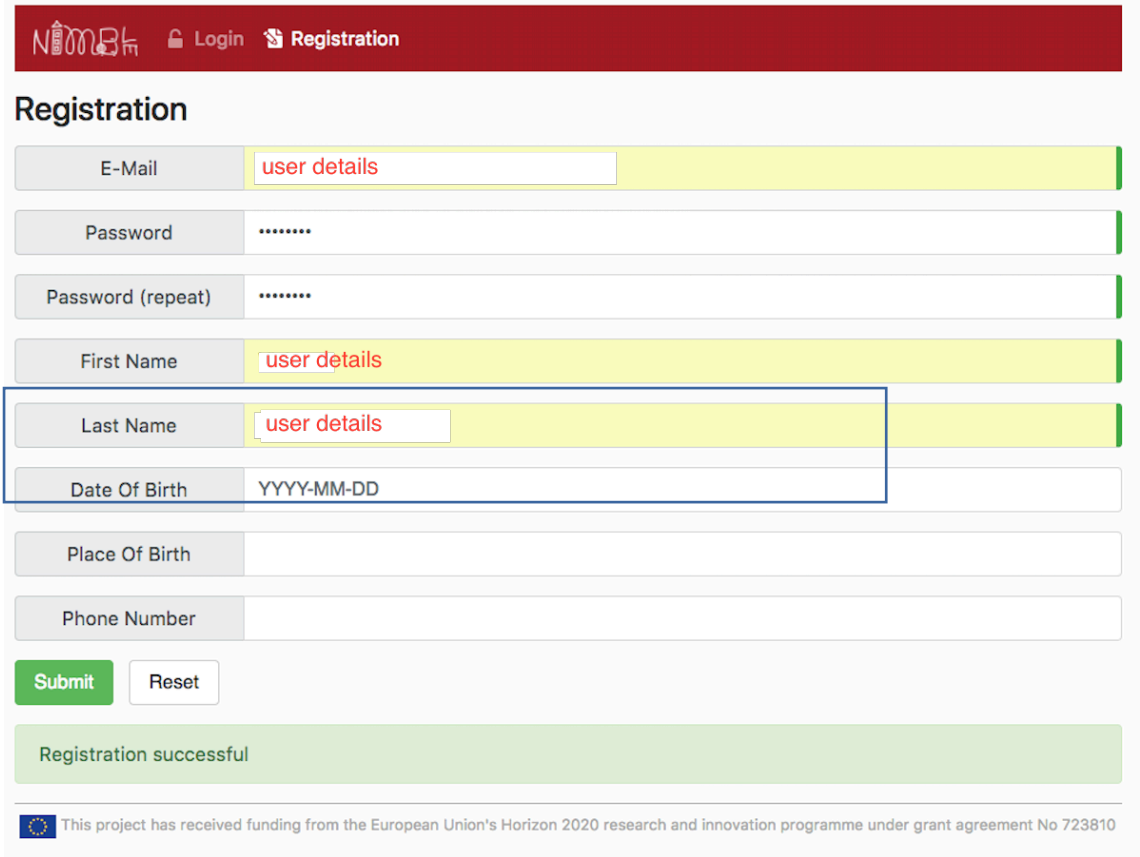

Figure 18: User registration process successfully finished

The Keycloak tool now contains some information about the registered user, e.g. by default, each new user has a generic assigned role: *nimble_user* (see Figure 19). Assigned roles *offline_access* and *uma_authorization* are created by the Keycloack tool as default values.

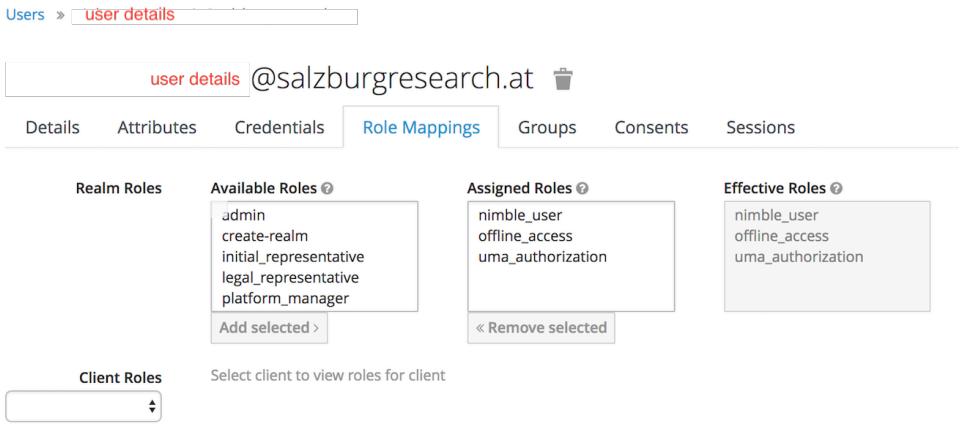

Figure 19: User details in Keycloak (after an initial user registration)

USER LOGIN. After the initial registration, the user must login to the NIMBLE platform (see Figure 20). The user is not linked yet to any company and has permissions to perform only search of available product catalogues (see the message shown in Figure 21).

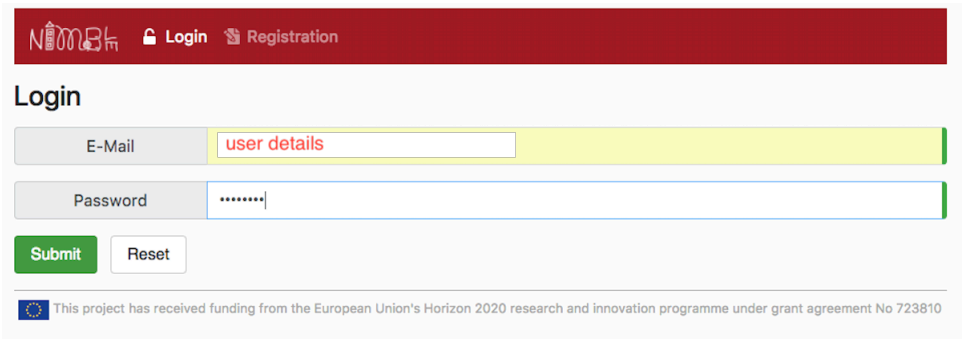

Figure 20: User login on the NIMBLE platform

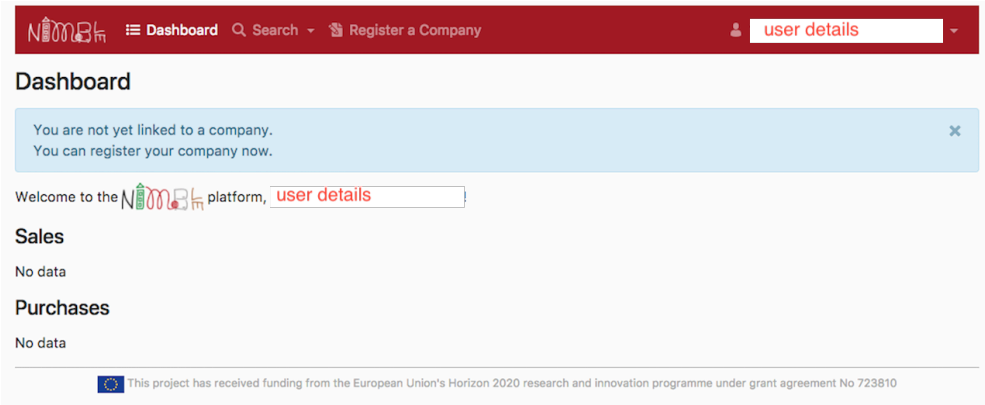

Figure 21: Login procedure finished

We switched the debugger on to show some of the technical details, e.g. Figure 22 shows a bearer token that is used for the user authorization during the login process. The token can be inspected in JSON Web Token (https://jwt.io/) which is an open, industry standard RFC 7519 method for representing claims between two parties. By adding bearer code generated after the user login, into a JWT debugger, we can inspect decoded token values, e.g. algorithm and token type, public key and certificate and some user data (user name, email, and role).

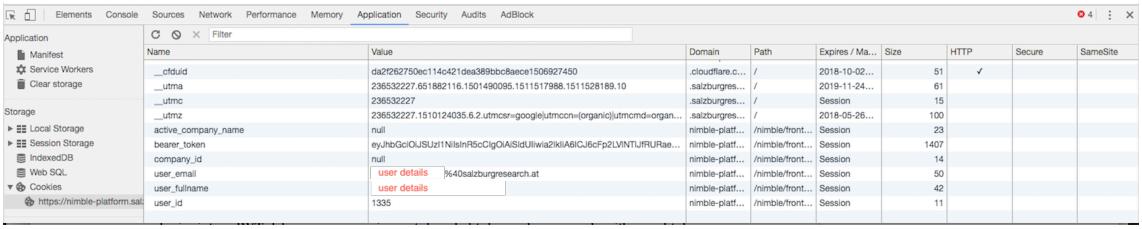

Figure 22: A bearer token after user's login

Figure 23 shows the JWT debugger with specific details of the generated bearer token. For example, the JWT includes the following elements:

- *Iss* - uniquely identifies the entity that issued the token,
- *Sub* - uniquely identifies the entity for which this token has been released (it is a key field when a token needs to be used also for authentication purposes),
- *Exp* - indicates the expiration time, after which this token should not be used and processed by any entity in the system,
- *Nbf* - identifies the time in which this token becomes effectively valid and can be processed by any entity in the system,
- *Iat* - identifies uniquely the time in which this token has been created,
- *Jti* - uniquely identifies the token.

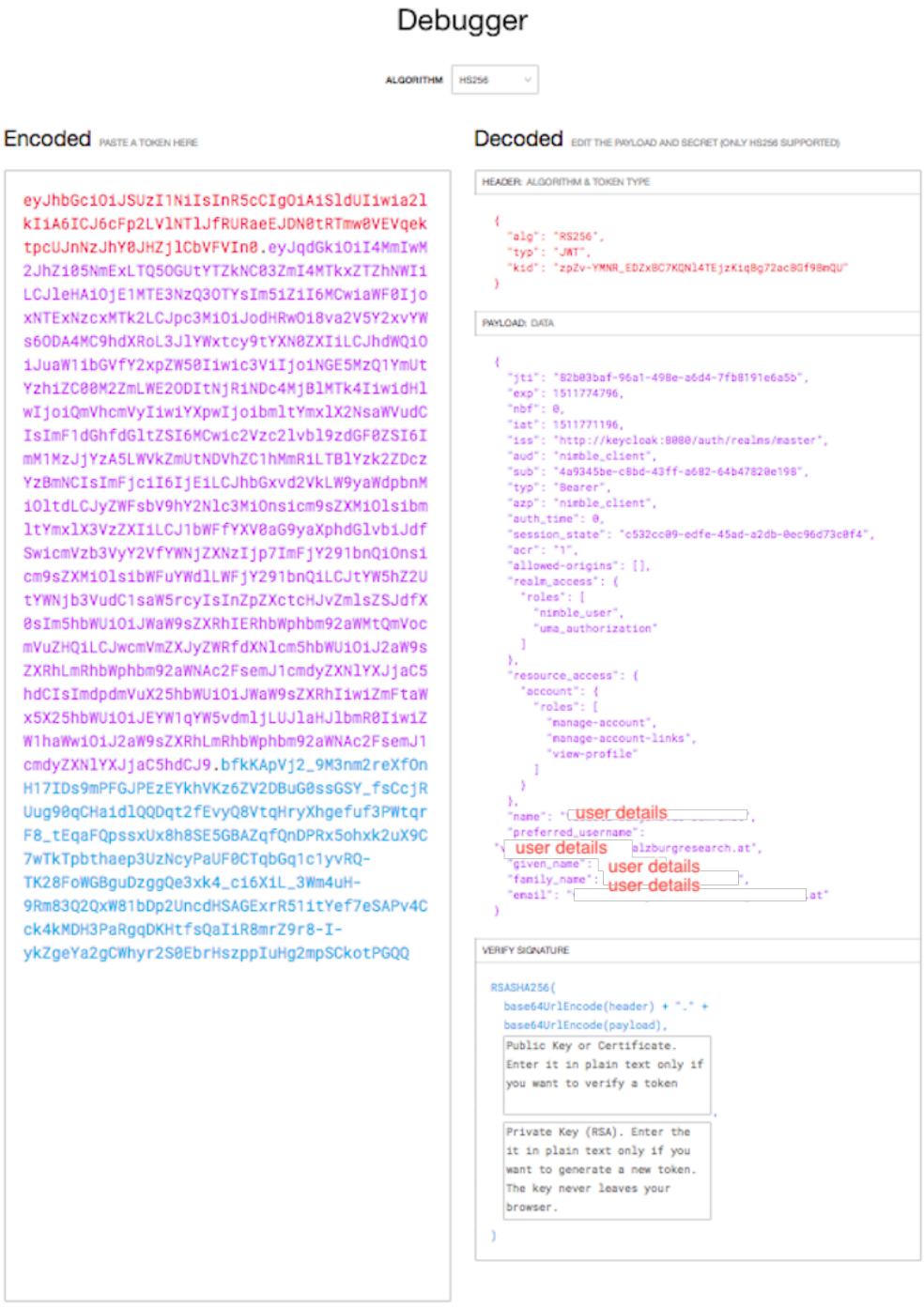

Figure 23: JWT debugger

COMPANY REGISTRATION. In order to perform B2B collaboration via the NIMBLE platform, the registered and logged in user needs to register a company. Figure 21 shows the user's dashboard displaying the message: "*You are not linked to a company. You can register your company now.*" Figure 21 also shows that at this stage, the user can either perform search of available product catalogues or register a company. By inspecting Figure 22, we can see that the *active_company_name* equals null (before the company registration).

In order to register a company, the user needs to enter several details about the company, e.g. company legal name and company address. The validity check of company information will be done manually for the release v1 of the platform, while future platform releases will integrate services for company validation. Figure 24 shows the form for the company registration (e.g. the user registers the company called MIMICUNA which is located in Austria).

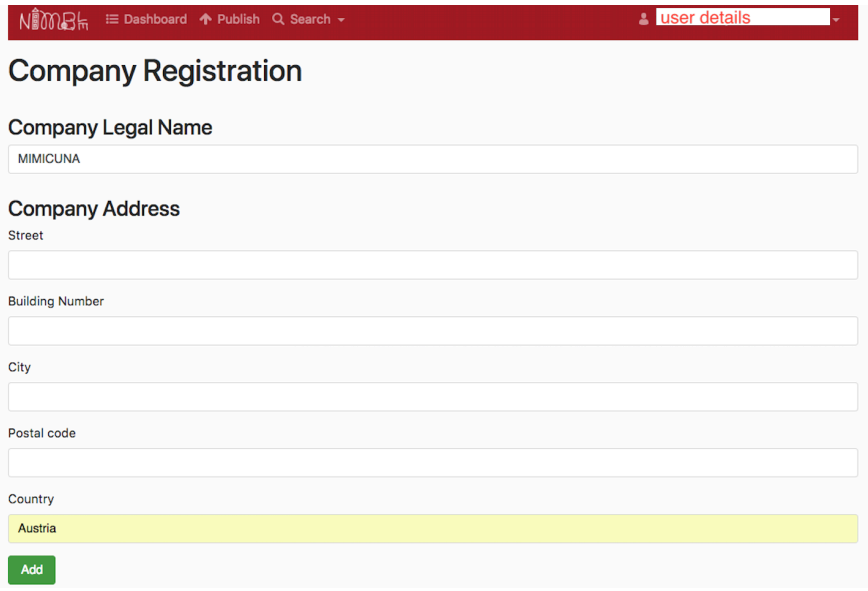

Figure 24: Company registration form

After clicking on the Add button, the company is registered and the user is bound to it (Figure 25). By inspecting Figure 26 we can see that the new *active_company_name* is not anymore null. It now has a specific *value* (e.g. MIMICUNA) and a *company_id* (e.g. 1372).

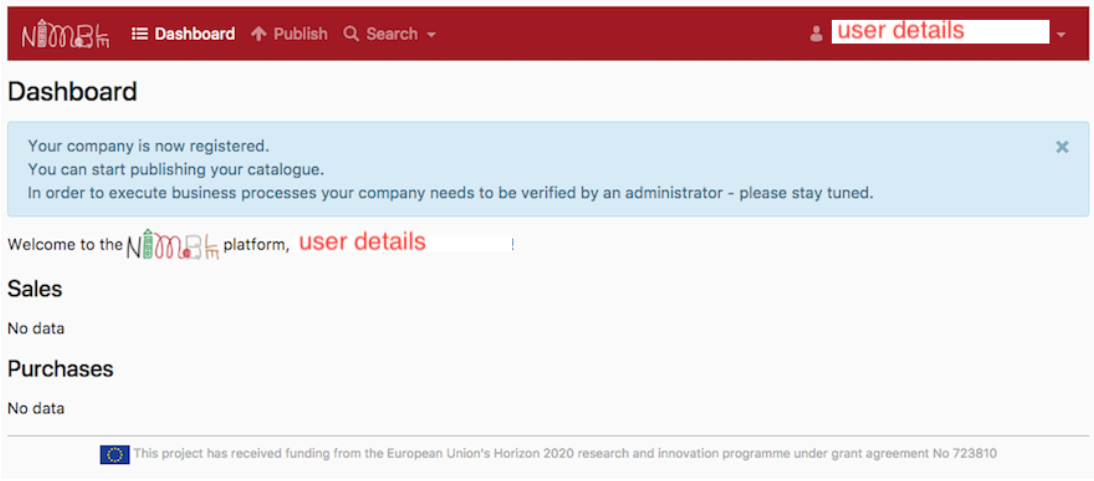

Figure 25: Company registration confirmation

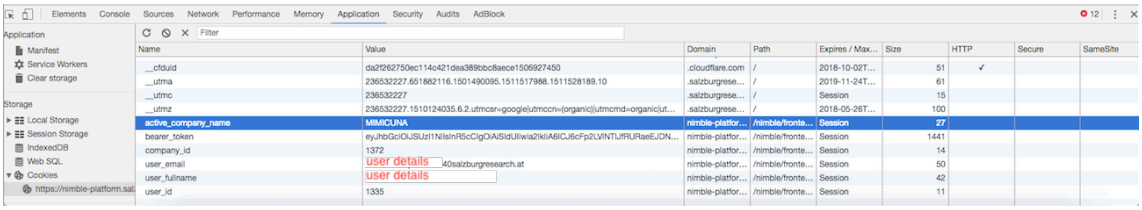

Figure 26: A bearer token after company registration confirmation

After the company registration, the role of the user is automatically changed from a generic *nimble_user* into *initial_user*, which can be seen in the Keycloak management tool. In addition, the user with the role *platform_owner* receives an automatically generated message about the newly registered company (see Figure 26).

Dear platform manager,

a new company registered on NIMBLE: MIMICUNA

user details is the current initial representative.

Sincerely,
your platform

Figure 27: Message to the NIMBLE platform manager (owner) confirming the presence of a newly registered company on the platform

The user can now invite other users to register to the NIMBLE platform and perform various B2B collaborative activities via the platform (Figure 28).

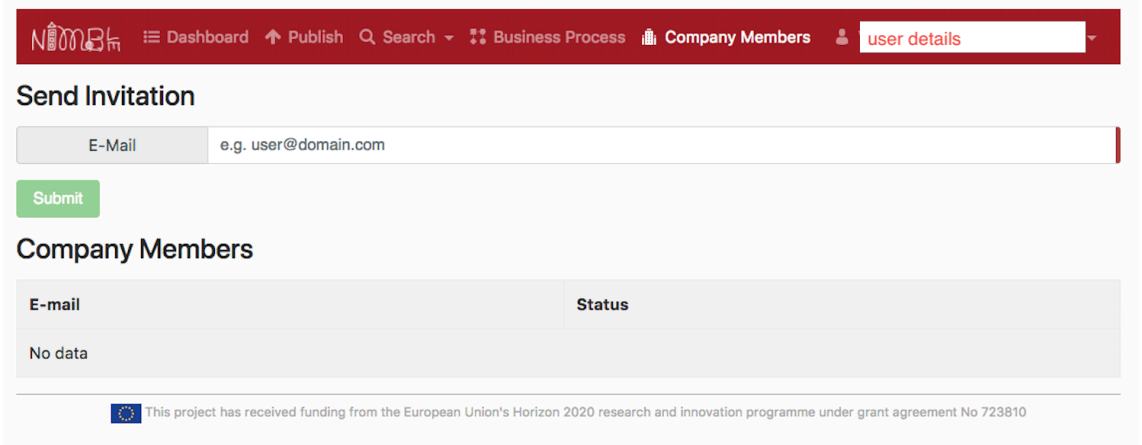

Figure 28: Sending invitations to other selected company members to join the platform

Figure 28 shows the user with maximum permissions designed in the demo platform: visiting the user dashboard, publishing the product catalogue, searching through the product catalogue, performing some business processes, and inviting new company members to join the platform.

5 Future Steps in Security Controls Implementation in NIMBLE

Future progress of the NIMBLE project development is expected to add more services to the platform's business functionality, improve consistency and trustworthiness in the entire data lifecycle, and advance the platform's security features. Hence, the security related work will be continued through the remaining planned activities in tasks T6.3 and T6.4. The most important progression of the platform's security features is expected in the areas of identity and access control management, data integrity, data quality and privacy and GDPR-related mechanisms.

5.1 Future Work on Identity and Access Control Management

Identity and access control management enables the creation, management and use of a digital identity related to the management of platform services, administration, the distribution of company information via the platform, etc. The complexity of identity and access management should neither be traded for poor security nor for poor user experience. For example, access to the resources and services involving multiple user accounts and passwords results in user frustration and confusion. Too simplistic access controls result in security vulnerabilities, while adding access control complexity will limit information sharing across the platform and will add to the administrative overhead. The associated identity and access management limitations lead to further security risks and reputation issues, which can have negative impact on the platform.

In NIMBLE, the identity and access management features are designed using the Keycloak tool, which facilitates the management of access controls and eliminates the need to duplicate identification and access control relevant information. Keycloak combines the RBAC and ABAC approaches for defining identity and access management in a way that is intuitive to an end-user and effective for both the platform system and the platform manager. It ensures that only registered and verified companies and their authorized users can access online resources and services, share information and perform transactions via managed permissions. It should also allow users to select the credentials for access needs (e.g. to assign specific roles to specific users), quick provisioning of services and online resources, and their protection through identifying policy violations and through removing inappropriate access privileges. For example, for most activities in NIMBLE, the scope of the data to be managed is within a single firm. In the case of production monitoring, a third party may be given rights to monitor certain data. This is yet to be designed in detail (see Appendix 1: *NIMBLE Cybersecurity Plan*), and there needs to be a policy specification language that allows company representatives to specify what can be monitored by who and when. Production monitoring is a very wide term, and the policy specification language needs to enable precisely defined monitoring scopes, for example:

```
from date (StartDate) to date (EndDate) user(fred01) can_monitor
(sensor(sense777) on machine(m222) from company (firm333) when
machining_part(?MP_X) of order(order999)) is being produced.
```

The Policy Enforcement Point (PEP) in Keycloak is a design pattern and can be implemented in different ways. For example, we can use XACML policies for making authorization decisions (see Figure 28 that shows an example of XACML based policy definition). The definition of the authorization policies will be designed in close relationship with the context (country, scope of business, location, etc.) and business functionality of services in NIMBLE. We will also extend current RBAC and ABAC access control models towards the ZBAC model, which requires that

the user authenticates in order to know which authorizations to grant. Such an approach is explained in detail in Section 3.3.

```
<Policy NIMBLE_access>
      <Target>
            <Resource>
                  <AttributeValue …> … </AttributeValue …>
                  <ResourceAttributeDesignator …> …
            </ResourceAttributeDesignator…>
            </Resource>
            <Action>
                  <AttributeValue …> … </AttributeValue …>
                  <ActionAttributeDesignator …> …
            </ActionAttributeDesignator…>
            </Action>
      </Target>
      <Rule x>
            <Target>
                  <Subject>
                        <AttributeValue …> … </AttributeValue …>
                        <SubjectAttributeDesignator …> …
                        </SubjectAttributeDesignator…>
                  </Subject>
            </Target>
      </Rule>
      <Obligation>
            <Apply …> …
            </Apply>
      </Obligation>
</Policy>
```

Figure 29: XACML policy definition in NIMBLE

5.2 Future Work on Data Integrity and Data Quality Management

Failure to control the distribution of data, data integrity and data quality often leads to data breaches, loss of sensitive information and data manipulation. Adequate security controls for identity, access controls, user authentication and authorization provide certain guarantees for data integrity and quality. Data manipulations related to cloud services, or based on comparison of products and suppliers, or filtering and ordering information to gain an unfair advantage in trading thus leading to monopolies require additional approaches to be taken. For example, cloud-based platforms put user's data into remote storages such that users lose control over their data. Therefore, proof of data integrity that guarantees the correctness of user's data in cloud storages needs to be provided. Otherwise, company data may become corrupted, without users being aware of it or being able to check for it. For this purpose, some general methods are proposed like mirroring, check-summing and using third party auditors amongst others [IMRA17]. These methods use extra storage space by maintaining multiple copies of data or the presence of a third-party verifier is required. The authors in [IMRA17] propose a data provenance based scheme through which users are able to check the integrity of their data stored in clouds, and track the violation of data integrity if occurred.

Recently, a new approach to managing identities, based on the distributed trust models and **blockchain technology** for controlling user's identities occurred. The two fundamental principles of such trusted identity management approach include (i) **the self-sovereign identity (user-centric identity) principle** empowers users to take full ownership and control of their identity information [IBM17], and (ii) **distributed trust model,** in which all parties can use an agreed-upon set of identity attributes to authenticate, verify and authorize individuals in order to perform trusted business or social transactions. In NIMBLE (tasks T6.3 and T6.4), we will explore the use of blockchain technology for digital identity.

The secure exchange of business information through file sharing, email and messaging system for negotiation, is another big concern for platform participants interacting over the NIMBLE platform. Our approach in NIMBLE is to use *provenance information* about access to the system and various actions performed via the platform, which need to be kept in audit logs. Provenance information matters in cybersecurity as a measure for preventing data manipulation that can cause harmful changes of product specifications (e.g. data sabotage, etc.). In addition, security controls for *anomaly detection* need to be regularly performed to capture unusual behaviour via the platform. For that purpose, in T6.4 we will use tools around the threat intelligence project MISP, which is open source Cybersecurity Threat Intelligence (CTI) platform and open standard for threat information sharing (for more details: http://www.misp-project.org/).

5.3 Future Work on Implementing Privacy Mechanisms in NIMBLE

In the literature, there exist various privacy mechanisms for IoT and the Web of Things [RONL11]:

- *privacy by design* [RUBI12] [GÜTD11] [KUFK11], *data transparency mechanisms*, *data management mechanisms* (policy-enforcement mechanisms, trust and governance);
- *obscurity by design* [STHA12|;
- *identity management mechanisms: digital shadowing* [SAGI09], *revocable access delegation* [RERG10], *privacy coach* [BROE10]), *data obfuscation* [SHOK15], etc.

To protect devices, privacy mechanisms need to preserve location of devices, personal information stored on devices, resilience to side channel attacks (by adding randomness or noise, having synchronous CPUs, etc.). Privacy models also vary from the *k-anonymity model* [SWEE02], the *data transformation/ randomization model*, the *cryptography and cryptographic techniques* such as *Secure Multiparty Computation (SMC)*. The SMC models [SEN11] could be based on *the secure sum protocol* [CKVL02], *the homomorphic encryption* [PAIL99], *the location- and identity-based encryption* [KOOL05], *privacy preserving attribute-based access control system* [FRAL06], *private collaborative forecasting and benchmarking* [ABLF04].

In NIMBLE, **data privacy will be based on trust**, that can be grounded on previous interactions and peer recommendations (to read more about a similar approach, see [GHSZ07] [SOMA08] [RFSM10]). Privacy based on trust is the most common situation in which trust evaluation process of the user's system is based either *on regulations/ legislations* that compel companies and agencies to guarantee the privacy of their users (e.g., [EUR-Lex][EC2012]) *or privacy policies or Statistical Disclosure Control (SDC)* that aims to protect the privacy of individuals while allowing the release of their data for secondary use [EKSS12][HUND10]. In May 2018, the privacy of the user's data must be preserved and **in compliance with the GDPR**,

which will have significant implications on online businesses in the EU. Hence, privacy mechanisms in NIMBLE need to be designed in accordance with GDPR.

In addition, in T6.4 we will further explore the design of optimal security mechanisms against adaptive inference, which is formulated as a *Stackelberg security game* [SHOK15]. Our approach to it had been presented in the NIMBLE paper "*Stackelberg Security Game for Optimizing Security of Federated IoT Platform Instances*" [DAMJ17].

- With respect to NIMBLE policy and governance models, they will be defined in the *Plan for NIMBLE Platform Governance* document, which will also cover the Data Integrity and Data Quality Policy (see D6.1, Table 28. The NIMBLE policy complies to the GDPR requirements. The *Plan for NIMBLE Platform Governance* will be a part of D8.8 *NIMBLE Platform SEED Programme: Manual and Materials Package*, and will be available in M17. Overall, the NIMBLE policy complies to the GDPR requirements (https://gdpr-info.eu/, see Deliverable D6.1).

REFERENCES

[ABLF04] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara, 2004. Private collaborative forecasting and benchmarking, in ACM workshop on Privacy in the electronic society, Washington DC, USA. Last accessed: November 2017.

[AHSM12] M. Ahmed, T.M.S. Sazzad, M.E. Mollah (2012). Cryptography and State-of-the-Art Techniques. In International Journal of Computer Science Issues (IJCSI), Vol. 9, Issue 2, No. 3, ISSN (Online): 1694-0814. Online available from: https://www.ijcsi.org/papers/IJCSI-9-2-3-583-586.pdf. Last accessed: November 2017.

[ANGG15] Anggorojati, B. (2015). Access Control in IoT/M2M - Cloud Platform. Department of Electronic Systems, Aalborg University. PhD Thesis. Online available from: http://vbn.aau.dk/files/208812056/Thesis_Bayu_Anggorojati.pdf

[BLFI99] M. Blaze, J. Feigenbaum, J. Ioannidis, (1999).The KeyNote Trust-Management System Version 2, IETF RFC 2704, Online available: http://www1.cs.columbia.edu/~angelos/Papers/rfc2704.txt Last accessed: Nov. 2017.

[BROE10] G. Broenink et al., 2010. The Privacy Coach: Supporting Customer Privacy in the Internet of Things, Proc. Workshop on What Can the Internet of Things Do for the Citizen?. Online: http://dare.ubn.ru.nl/bitstream/2066/83839/1/83839.pdf

[BROW07] K. Brown,(2007). Exploring Claims-Based Identity, Online available from: https://blogs.msdn.microsoft.com/msdnmagazine/2007/08/20/security-briefs-exploring-claims-based-identity/

[CKVL02] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, 2002. Tools for privacy preserving distributed data mining, SIGKDD Explor. Newsletter. Vol. 4, pp. 28-34.

[COMP-D5.1.1] COMPOSE project (2013). D5.1.1 Security requirements and architecture for COMPOSE. Online available from: http://bit.ly/2Bkb4zS (Last accessed: Nov. 2017)

[DAMJ17] V. Damjanovic-Behrendt, 2017. Stackelberg Security Game for Optimizing Security of Federated IoT Platform Instances. In Proc. of the International Conference on Decision and Game Theory for Security (ICDGTS 17), Barcelona, Spain. Online available from: http://bit.ly/2relT3s (Last Accessed: December 2017)

[DECI13] DECIPHERpcp project, 2012. D5.1 EHR-PHR interfaces and PHR platforms. State-of-the-art report. Available online from: http://www.decipherpcp.eu/sites/default/files/attachments/decipher_d5_1_phr_platforms_state-of-the-art_v1_0wb_r.pdf Last accessed: Nov. 2017.

[EC2012] European Commission. Justice. Commission proposes a comprehensive reform of the data protection rules. January 2012. Online: http://ec.europa.eu/justice/newsroom/data-protection/news/120125_en.htm

[EKSS12] M.J. Elliot, D. Karla, P. Singleton, D. Smith, 2012. Statistical Disclosure Control and Protecting Privacy within a Clinical Data Warehouse. Manchester. Online: http://www.ccsr.ac.uk/documents/Statistical_Disclosure_Control.pdf

[EUR-Lex] EURATEX Annual Report 2014. The European Apparel and Textile Confederation. Online: bit.ly/1OI38L8.

[FIPS] FIPS PUB 197: The official Advanced Encryption Standard.

[FRAL06] K. Frikken, M. Atallah, and J. Li, 2006. Attribute-Based Access Control with Hidden Policies and Hidden Credentials, IEEE Trans. Comput., vol. 55, pp. 1259-1270.

[GHSZ07] P.D. Giang, L.X. Hung, R.A. Shaikh, Y. Zhung, S. Lee, J.K. Lee, H. Lee, 2007. A Trust-Based Approach to Control Privacy Exposure in Ubiquitous Computing Environments, Pervasive Services. Online available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4283905&isnumber=4283875 Last accessed: Nov. 2017.

[GUTD11] Gurses, S., Troncoso, C., and Diaz, C., 2011. Engineering privacy by design. Paper presented at CPDP 2011, Belgium.

[HARD12] Hardt, D.: The OAuth 2.0 Authorization Framework. IETF RFC 6749, 2012.

[HARD88] N. Hardy, 1988. The Confused Deputy (or why capabilities might have been invented). ACM SIGOPS Operating Systems Review, Volume 22, Issue 4, pages 36-38.

[HUFE14] V. Hu, et al. (2014). Guide to Attribute Based Access Control (ABAC) definition and considerations. NIST special publication 800-162. NIST, January 2014.

[HUND10] A. Hundepool et al., 2010. Handbook on Statistical Disclosure Control, A Network of Excellence in the European Statistical Systems in the field of Statistical Disclosure Control. January 2012. Online: http://neon.vb.cbs.nl/casc/.%5Csdc_handbook.pdf

[IBM17] Trust Me: Digital Identity on Blockchain. ExpertInsights@IBV. 2017. Online available: https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=GBE03823USEN& Last access: September 2017

[IMRA17] Imran M, Hlavacs H, Haq IU, Jan B, Khan FA, Ahmad A. Provenance based data integrity checking and verification in cloud environments. Shi Y, ed. PLoS ONE. 2017;12(5):e0177576. doi:10.1371/journal.pone.0177576.

[JEKU16] A.P. Jesu, S.R. Kumar, 2016. A Survey of Techniques and Tools Used for Cloud Data Integrity Verification. In Int. Journal of Innovative Research in Computer and Communication Engineering. Vol. 4m Issues 2. Online available: https://www.ijircce.com/upload/2016/february/208_A%20Survey.pdf

[KAHD09] Karp, A.H., Haury, H., Davis, M.H. (2009). From ABAC to ZBAC: The Evolution of Access Control Models. HPL Labs. Online available from: http://www.hpl.hp.com/techreports/2009/HPL-2009-30.pdf

[Keycloak] Authorization Services Guide. Online available from: http://www.keycloak.org/docs/latest/authorization_services/index.html

[KOOL05] G. M. Koien and V. A. Oleshchuk, 2005. Location Privacy for Cellular Systems; Analysis and Solution, in 5th Workshop on Privacy Enhancing Technologies (PET 05), 2005, pp. 40-58.

[KUFK11] Kung, A., Freytag, J., and Kargl, F., 2011. Privacy-by-design in ITS applications. Proc. IEEE WoWMoM.

[MAKK00] J. Massey, G. Khachatrian, M. Kuregian, 2000. "Nomination of SAFER++ as Candidate Algorithm for the New European Schemes for Signatures, Integrity, and Encryption (NESSIE)" Presented in First Open NESSIE Workshop, November 2000.

[MOST90] Moffett, J., Sloman, M., & Twidle, K. (1990). Specifying discretionary access control policy for distributed systems. Computer Communications, 13(9), 571-580.

[NIST-ABAC14] Vincent C. et a. NIST Special Publication 800-162. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. Online available from: http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf (Last accessed: November 2017).

[OASIS-SAML] Organization for the Advancement of Structured Information Standards (OASIS): Official Wiki of the OASIS security services (SAML) technical committee. https://wiki.oasis-open.org/security/FrontPage

[OASIS-XACML] OASIS Standard. eXtensible Access Control Markup Language (XACML) Version 3.0. January 2013: http://docs.oasis-open.org/xacml/3.0

[OpenID] OpenID Authentication 2.0. Finalized OpenID Specification (Dec. 2007)

[OWASP-CRYPTO] OWASP. Guide to Cryptography. Online available from: https://www.owasp.org/index.php/Guide_to_Cryptography

[PAIL99] P. Paillier, 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, in EUROCRYPT 99, 1999, pp. 223-238.

[PAZA13] Paquin, C. and Zaverucha, G. (2013). U-Prove Cryptographic Specification V1.1. Revision 3. Available at http://research.microsoft.com/en-us/projects/u-prove/

[PIKI06] A. Pimlott and O. Kiselyov, (2006). Soutei, a Logic-Based Trust-Management System, FLOPS 2006, 8th International Symposium on Functional and Logic Programming. Fuji-Susono, Japan. Also in Springer's Lecture Notes in Computer Science 3945/2006, pp. 130-14.

[RERG10] E. Rekleitis, P. Rizomiliotis, & S. Gritzalis, 2010. A Holistic Approach to RFID Security and Privacy, Proc. 1st International Workshop Security of the Internet of Things (SecIoT 10), Network Information and Computer Security Laboratory. Online: www.nics.uma.es/seciot10/files/pdf/rekleitis_seciot10_paper.pdf

[RFC6749] D. Hardt (Ed.), (2012). RFC6749: The OAuth 2.0 Authorization Framework, IETF. Online available from : http://tools.ietf.org/html/rfc6749

[RFC6750] M. Jones & D. Hardt, (2012). RFC6750: The OAuth 2.0 Authorization Framework: Bearer Token Usage, IETF. Online available: http://tools.ietf.org/html/rfc6750

[RFSM10] D. Rebollo-Monedero, J. Forné, A. Solana, A. Martinez-Balleste, 2010. Private location-based information retrieval through user collaboration. Computer Communications, Vol.33, No.6, pp. 762-774.

[RONL11] R. Roman, P. Najera, J. Lopez, 2011. Securing the Internet of Things. In IEEE Computer, Vol. 44, No. 9, pp. 51-58.

[RUBI12] Rubinstein, I. 2012. Regulating Privacy by Design. Berkeley Technology Law Journal.

[SAGI09] A. Sarma & J. Girao, 2009. Identities in the Future Internet of Things, Wireless Personal Comm., pp. 353-363

[SAML] SAML. Online available: http://docs.oasisopen.org/security/saml/v2.0

[SAND93] Sandhu, R. S. (1993). Lattice-based access control models. Computer, 26(11), 9-19.

[SBJM11] Sakimura, D. N., Bradley, J., Jones, M., de Medeiros, B., & Jay, E. (2011). OpenID Connect Standard 1.0-draft 20.

[SCFY96] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. Computer, 29(2), 38-47.

[SCIA17] S. Sciancalepore, et ll. (2017). Attribute-Based Access Control Scheme in Federated IoT Platforms. In Podnar Zarko at al. (Eds.): InterOSS-IoT 2016, LNCS 10218, pp. 123-138, 2017.

[SEN11] J. Sen, 2011. Privacy Preservation Technologies in Internet of Things, Proc. International Conf. Emerging Trends in Mathematics, Technology, and Management, 2011; http://arxiv.org/ftp/arxiv/papers/1012/1012.2177.pdf

[SHOK15] R. Shokri, 2015. Privacy Games: Optimal User-Centric Data Obfuscation. In Proceedings on Privacy Enhancing Technologies 2015 (2), 1-17.

[SHOS14] A. Shostack (2014). Threat Modeling. Designing for Security. John Wiley & Sons, Inc.

[SOC14] SOCIoTAL (2014). D2.2 Framework Specification for Privacy and Access Control. Online: http://cordis.europa.eu/docs/projects/cnect/2/609112/080/deliverables/001-609112SOCIOTALD22renditionDownload.pdf (Last accessed: November 2017).

[SOMA08] A. Solanas and A. Martinez-Balleste, 2008. A TTP-free protocol for location privacy in location-based services. Computer Communications 31(6): 1181-1191.

[STHA12| F. Stutzman, W. Hartzog, 2012. Obscurity by Design: An Approach to Building Privacy into Social Media. In Proceedings of the CSCW 12, Seattle, USA.

[SWEE02] Latanya Sweeney, 2002. K-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems, 10(5):557-570.

Appendix 1: NIMBLE Cybersecurity Plan

Project month & platform release	Project related task/ WP	Description
M13	T6.1: Security and Privacy Requirements	Identification and classification of security and privacy requirements in NIMBLE
M14 = V1.0	T6.2: Design and Implementation of Security and Privacy for Core Business Services	Identity and access control management; User authentication
M17	T8.8: NIMBLE Platform SEED Programme: Manual and Materials Package	*Plan for NIMBLE Platform Governance* (document covering the data integrity and data quality policies)
M18 = V2.0	WP3 tasks; T8.8: NIMBLE Platform SEED Programme: Manual and Materials Package	User authorization; Privacy mechanisms based on the NIMBLE Privacy Framework; GDPR privacy compliance; Data integrity;
M19	T6.3: Trust and Reputation Management	Trust models; Reputation mechanisms; Privacy based on trust;
M21 = V3.0	T6.4: Advanced Trust, Security, Privacy and Reputation Management Tools	Designing game theory advanced security optimization models
M24 = V4.0	T6.4	Designing data quality mechanisms
M27 = V5.0	T6.4	Security analysis of provenance information
M30 = V6.0	T6.4	Data integration and anomaly detection (MISP threat intelligence)
M32	T6.4	Implementing game theory based security optimization models
M33 = V7.0	T6.4	MISP threat intelligence
M36 = V8.0	T6.4	MISP threat intelligence