Specific Targeted Research Projects (STReP)

SOCIOTAL

Creating a socially aware citizen-centric Internet of Things

FP7 Contract Number: 609112

WP2 – Decentralised governance and trust framework

Deliverable report

Contractual date of delivery: 30/11/14
Actual submission date:

Deliverable ID:	**D2.2**
Deliverable Title:	**Framework Specification for Privacy and Access Control**
Responsible beneficiary:	UMU, UNIS, UC, DNET
Contributing beneficiaries:	
Estimated Indicative Person Months:	

Start Date of the Project: 1 September 2013 Duration: 36 Months

Revision:
Dissemination Level: Public

Document Information

Document ID: D2.2
Version: 1.0
Version Date: 23 December 2014
Authors: José Luis Hernández Ramos, Jorge Bernal Bernabé, Antonio Skarmeta Gómez (UMU), Ignacio Elicegui Maestro (UC), Michele Nati (UNIS), Nenad Gligoric (DNET)
Security: **Confidential**

Approvals

	Name	Organization	Date	Visa
Project Management Team	Klaus Moessner	UNIS		

Document history

Revision	Date	Modification	Authors
0.1	26/09/2014	First ToC	UMU
0.2	05/11/2014	Added contribution to different sections	UMU
0.3	07/11/2014	Added contribution to different sections	UMU, DNET
0.4	14/11/2014	Added contribution and modifications to different sections	UMU, UC, UNIS
0.5	21/11/2014	General improvement in different sections. Editing work throughout the whole document. Conclusions and References.	UMU
0.6	28/11/2014	General improvement in different sections. Added contributions to section 3.1. Current version is ready for internal review	UMU, UC, UNIS
0.7	19/12/2014	Addressed comments from the internal review carried out by CRS4 and CEA	UMU
1.0	23/12/2014	Final version ready for submission	UMU

Content

Description of the deliverable content and purpose 4
Section 1 - State of the art 6
1.1 Access control and privacy-preserving technologies 6
1.2 IoT Access Control and Privacy related projects 10
1.3 IoT Access Control and Privacy Related Work 12
Section 2 - SocIoTal Access Control System for the IoT 14
2.1 Policy-based Access Control System 14
2.2 Capability-based Access Control System 16
2.3 Context driven Access Control System 20
2.4 Trust-based Access Control System 21
Section 3 - SocIoTal framework interactions for Privacy and Access Control 25
3.1 Context-aware security framework 26
3.2 Identity Management 36
3.3 Access Control 40
3.4 Secure Group Sharing 46
Section 4 - Managing Access Control and Privacy in SocIoTal communities and bubbles 51
4.1 Access Control in communities and bubbles 51
4.2 Secure Data Sharing in communities and bubbles 52
Section 5 - SocIoTal Security Framework in FI-WARE platform 57
5.1 Extending the Fi-ware Access Control with SocIoTal AC 57
5.2 Extending the Fi-ware Identity Management with SocIoTal IdM 59
Section 6 - Conclusion 63
Section 7 - References 64

Executive summary

Description of the deliverable content and purpose

SocIoTal aims to define a set of enablers in order to design a secure, trustworthy and privacy-preserving IoT environment. This ecosystem is intended to foster the creation of a socially aware citizen-centric IoT by encouraging people to contribute with their smart objects and information flows. One of the main goals of SocIoTal is the design of a framework enabling the application of proper security mechanisms to be deployed on IoT scenarios, in which IoT devices can be associated composing communities or bubbles. While Task 1.2 of SocIoTal is offering the design of an ARM-compliant architecture for the project, WP2 package provided an IoT security framework, which is in turn, based on the SocIoTal architecture, for establishing IoT identities and trust relationships between IoT devices. The framework allows sharing information within groups of entities (i.e. communities and bubbles) in a secure and privacy-preserving way, by giving users full control over their personal data disclosure.

Starting from the SocIoTal security framework which was presented in D2.1, this document provides the definition of the main interactions of such framework in order to show how access control and privacy are addressed by SocIoTal. Furthermore, it provides the design of specific access control and privacy-preserving mechanisms that are integrated in the proposed framework and applied for an effective management of IoT communities and bubbles. Thus, **Section 1** of this document provides a description of different access control and privacy technologies, as well as an analysis of related EU project addressing these issues. Then, the main foundations of the SocIoTal access control system are presented in **Section 2** of the document. It embraces the explanation of the four main aspects of the proposed system. Specifically, it is based on the use of access control policies (e.g. XACML), which are employed to generate capability tokens, specifying the granted privilege. These tokens are attached into requests to get access an IoT service. In addition, it considers trust and reputation values, as well as context information to drive access control decisions in an IoT environment.

While D2.1 provided a first description of the SocIoTal security framework, which complements the ARM architecture defined by IoT-A, **Section 3** shows the main interactions of the security functional components in order to realize the SocIoTal access control and privacy-preserving mechanisms. In particular, it proposes to use contextual information to drive security decisions for SocIoTal scenarios. The proposed Context Model is based on the specification of virtual entities as well as the OMA NGSI Context Management specification. This model is deployed by the Context Manager component of the SocIoTal framework and used by other security functional components. Thus, the main interactions for context-aware identity management, access control and group sharing mechanisms are provided.

Moreover, **Section 4** of the document provides how the proposed context-aware access control and privacy-preserving mechanisms are applied for a secure management and communications of communities and bubbles. It proposes appropriate procedures in order to deal with the concerns for secure interactions between trustworthy devices composing communities and bubbles. Specifically, the access control mechanism is based on the use of authorization credentials (i.e. capability tokens) which are obtained and used by entities in order to enable an end-to-end secure communication. Typically, these communities and bubbles will be managed by other entities responsible for generating and delivering proper credentials. Additionally, an overview of the group sharing mechanism based on the CP-ABE cryptographic scheme, and its application to the communities and bubbles scenarios is

provided, as part of the ongoing work related to Task 3.3, which will be described in detail at D3.3: “Secure Group Communication”.

Furthermore, **Section 5** gives a description about possible integrations of the SocIoTal security framework components into current IoT platforms, such as FI-WARE, which is derived by other European efforts. Indeed, it provides a first insight to enforce part of the SocIoTal security mechanisms over the existing, well-known and tested FI-WARE security platform. In particular, the integration of SocIoTal identity management and access control mechanisms into the Access Control Generic Enabler (GE) and the Identity Management GE, is provided.

Section 1 - State of the art

The initial section of this document provides an overview of the main models and technologies for access control models, as well as other European efforts addressing these concerns. Additionally, an overview of some recent research works tackling the application of access control and privacy-preserving mechanisms to the IoT paradigm is given at the end of this section.

1.1 Access control and privacy-preserving technologies

1.1.1 Access Control Models

Nowadays, there is a plethora of access control models that are applied to different Internet scenarios in which security is required. Here, it is provided a brief description of the most popular models, which are commonly considered and deployed in such scenarios.

In the *Mandatory Access Control* (MAC) model [1] the administrator of the system can give permissions for subject to access object. The model assigns security labels to subjects and objects, and it is independent of the user operations, only the administrator can modify object security labels. The model puts restrictions on user actions that, while adhering to security policies, avoid dynamic alteration of the underlying policies, but it requires the isolation of the MAC system from the operating system in order to maintain the security policies and prevent unauthorized access. MAC models are not commonly used as access control system because they are difficult and expensive to implement and maintain. Its usage is usually limited to military applications.

Moreover, in the *Discretionary Access Control* (DAC) models [2], the access to resources is maintained by users, which can grant permissions to their resources by being included in Access Control Lists (ACL). Each entry in the access control list gives users (or group of subjects) permissions to access resources. The permissions are usually stored by objects to avoid having a unique and dense matrix which would imply a waste of memory and performance decline. Unlike in MAC, where permissions are given in predefined policies by the administrator, in DAC, permissions are given by users which decide the access rights to the resources they belong. DAC is broadly adopted by current operation systems based on UNIX, FreeBSD, and Windows.

The *Role-Based Access Control* (RBAC) model [3] encompasses features from MAC and DAC providing a more generalized framework that can be customized per application. 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. Since the users are associated to roles, the user can access certain resources and perform specific tasks. Right granting and policy enforcement are carried out by the administrator and users cannot transfer permissions to on their role, to other users. RBAC allows creating hierarchies of permissions and inheritance, wherein more restrictive permissions override more general permissions. Nonetheless, RBAC has some problems since the administrative issues of large systems where memberships, role inheritance, and the need for fine-grained customized privileges make administration potentially cumbersome.

Traditional access control models like MAC, DAC and RBAC are user-centric and do not take into account additional parameters such as resource information, relationship between the user (the requesting entity) and the resource, and dynamic information (e.g. contextual information, such as time, location, user address). In order to provide a more fine-grained access mechanism, the *Attribute-Based Access Control* (ABAC) model [4] was proposed, in

which authorization decisions are based on attributes that the user has to prove (e.g.: age, location, roles, etc.), as well as resources and environmental properties. Attributes labels can be used to describe the entities that must be considered for authorization purposes. Every attribute can consist of a key-value pair such as "Role=Manager". In ABAC, like RBAC, the privileges are usually granted to users through the usage of policies that combine attributes altogether. Thanks to the usage of ABAC, unlike in traditional RBAC models, the number of rules can be reduced, at the expense of more powerful (and complex) rules and more processing and data availability requirements. One of the main advantages of ABAC is requesters do not have to be known a priori by targets, providing a higher level of flexibility for open environments, compared to RBAC models. Nevertheless, in ABAC everyone must agree on a set of attributes and their meaning when using ABAC, which is not easy to accomplish.

The *AuthoriZation-Based Access Control* (ZBAC) [5] model uses authorization credentials, which are presented along a request to make an access control decision. Unlike ABAC and RBAC systems, in which the user submits an authentication along with the service request, in ZBAC systems, the user submits an authorization along with the request. ZBAC deals with authorization in distributed systems where problems like identity federated management, *Single Sign-On* (SSO) or last privilege appear. In traditional and single domain access control models, the authentication is done at request time in the system's domain; the access decision is made by using that authentication to determine the authorization. With ZBAC, users access to resources in a domain other than home user' domain, and it is based on authentication on the user's domain before the request is made. It should be noticed that this 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) only needs to verify the validity of the authorization to make an access decision. This is a valuable feature for IoT scenarios in which constrained devices could interact each other, since interactions with third parties are not required for each communication. Consequently, the SocIoTal access control system is based on the main foundations of this approach.

1.1.2 Technologies for Access Control and Privacy

1.1.2.1. OpenID

OpenID [6] is an open authentication standard so that users can be authenticated with a single identity by different Internet services, enabling a SSO mechanism. In this way, a user can prove his identity in different web sites without the need to hold several identifiers. Currently, OpenID is a widely deployed technology by large companies such as Google or Yahoo.

In a typical OpenID scenario, a user is enabled to create an account through a OpenID Provider (OP), which can be used to get access a web service acting as a Relying Party (RP). The communication between OP and RP is provided by OpenID. Additionally, it defines how user attributes can be transferred (e.g. name or age) from the OP to the RP depending on the access requirements from the latter. OpenID Providers are usually provided by major companies for their clients, which receive an OpenID identifier. This identifier takes the form of a unique Uniform Resource Identifier (URI) and is used by a user when he tries to get access a specific RP. After that, the RP visits this URI in order to reach the proper OP to authenticate the requesting user.

One of the main advantages of OpenID is that it does not force to use a specific authentication mechanism, allowing login/password or smart cards approaches, for instance.

However, OpenID does not provide proper procedures in order to preserve the privacy of users, which can be tracked because of the use of the same OpenID identifier in different web sites.

1.1.2.2. SAML

The *Security Assertion Markup Language* (SAML) [7] standard defines a framework based on XML for describing and exchanging security information between entities. Such security information is expressed using SAML statements regarding the identity, attributes and authorizations of a subject. SAML defines the syntax and a set of rules for the creation, communication and use of these sentences. Usually, SAML exchanges take place between an *asserting party* and a *relying party*. The former party is an entity that creates and communicates SAML statements or constructs, while the latter uses these sentences for a specific task according the content. Specifically, SAML defines three types of assertion: *Authentication Assertion* (which contains information about a user's authentication), *Attribute Assertion* (transferring a user's attributes), and *Authorization Decision Assertion* (including authorization decision statements (e.g., permit, deny). Depending on the type of protocols or statements which used for a particular service, SAML entities may adopt a different nomenclature. For example, 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 four main elements:

- *Assertions*: as already mentioned, they contain information about identity, authentication and authorization of a particular subject. The structure and content of these messages allows them to be used in different contexts.
- *Protocols*: they define the way in which assertions can be requested and answered.
- *Bindings*: they are used to define how protocols messages are transported by employing other lower layer protocols, such as HTTP.
- *Profiles*: protocols, bindings and SAML assertions are combined composing profiles. A profile can be considered as the set of elements and interactions between them, which are needed to realize a specific SAML use case.

1.1.2.3. OAuth2

OAuth 2.0 [8] is the evolution of the OAuth 1.0 protocol as a protocol enabling third parties applications or clients to get access resources or services, which are owned by a user. These resources are commonly hosted in trusted applications. The main motivation behind the use of OAuth2 is allowing the access to resources without the need to give user's credentials to a third-party application. Consequently, they can access resources owned by the user, but these applications do not know the authentication credentials. In this way, through the OAuth 2.0 authorization framework, an application can obtain limited access to a resource, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.

OAuth 2.0 introduces an authorization layer and separates the role of the client from that of the resource owner. In this way, the client requests access to resources which are managed by the resource owner and hosted by the resource server, and is issued a different set of credentials than those of the resource owner. This approach has two main advantages. On the one hand, it solves the problem of trust between a user and third-party applications. In Oauth 2.0, a user can allow applications to collect data and perform tasks on their behalf but without giving the authentication credentials to such applications. On the other hand, it allows a service provider to give third-party applications the possibility to expand their services with applications that use data users in a secure way. Users can decide when and to who revoke

or provide access to their data, creating an application ecosystem around the service provider. While Oauth 2.0 is widely used by different services into the Internet nowadays, it does not meet the requirementes of certain IoT scenarios where devices can interact each other, avoiding the access to central authorization services to check access rights every time.

1.1.2.4. U-Prove

Most of previous technologies are not able to address properly privacy issues that are required at IoT scenarios. In this sense, stakeholders using IoT services should be able to control how and to whom their private information is disclosed. Unlike traditional identity management technologies, anonymous credential systems (e.g. U-Prove and Idemix) enable selective disclosure of personal identity information in order to enhance users' privacy. They allow the use of cryptographic proofs, which can derived from identity credentials in order to prove a specific set of identity attributes, without the need to set the whole credential, as in the case of certificates-based solutions.

U-Prove [9] is an anonymous credential system intended to manage *claims-based identity* [10]. It is based on the use of U-Prove tokens as attribute information containers, which are generated and delivered by an *Issuer* to a *Prover*. Then, a Prover can use these credentials to prove an identity to a service acting as a *Verifier*. The communication between an Issuer and a Prover is carried out through an *issuance* protocol, while a *presentation* protocol is employed for the communication between a Prover and a Verifier. Because of the use of the public key type 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, even by colluding insiders.

In a typical scenario, each U-Prove token corresponds to a unique private key, which is generated by a Prover during an issuance protocol. Then, when such token is used by a Prover, it applies the token's private key to a message in order to generate a presentation proof. This proof is actually a proof-of-possession of the private key as well as a digital signature of the Prover on the message. When presenting the token to a Verifier, the message can be used as a presentation challenge 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; this ensures that the token cannot be stolen through eavesdropping or phishing and prevents unauthorized replay by legitimate Verifiers. Arbitrarily many presentation proofs or signatures may be created with the same U-Prove token.

1.1.2.5. Idemix

Identity Mixer (Idemix) [11] is an alternative anonymous credential system, which provides selective disclosure of attributes, enabling users to minimize the personal data they have to disclose in electronic transactions. With a similar nomenclature, Idemix defines the roles of *issuers, recipients, provers and verifiers*. During a *credential issuance* stage between an *issuer* and a *recipient*, the latter gets an Idemix credential. This credential consists of a set of attribute values as well as cryptographic information that allows the owner of the credential to create a proof of possession. That is, when creating a proof the credential owner (i.e. the user) acts in the role of the *prover* communicating with a *verifier* (i.e. Service provider). Several zero-knowledge proofs are performed to convince the Service provider the possession of a signed statement about and attribute.

Two kinds of protocols are required for a normal operation of Idemix. On the one hand, an *issuance protocol* is used by a recipient to obtain a credential against an issuer. The recipient computes a cryptographic message with the set of attributes to be included in the credential, which is signed by the issuer. Furthermore, the recipient and the issuer share the same XML-based credential structure definition. On the other hand, a *prove protocol* is initiated by the prover (e.g. a user) in order to prove the possession of a certain credential to the verifier (e.g. a service provider), without revealing the credential itself. For this purpose, the prover firstly creates a cryptographic proof defining a proof which is sent to the verifier. Such proof actually contains a proof of possession of a CL-signature [12], which ensures that the prover possess a credential signed by the issuer. Then the verifier can verify the proof and send the response to the user.

1.1.2.6. XACML

RBAC and ABAC models are usually specified in a policy-oriented way by using the *eXtensible Access Control Markup Language* (XACML) [13]. XACML is a language intended to express access control policies in a standard way, specifying the set of subjects who can perform certain actions on a specific set of resources, based on information (attributes) of them. In general, an individual can request an action to be executed on a resource, and XACML policies are then evaluated in order to deny or allow the requested operation. Furthermore, XACML is also a representation format in order to encoding access control requests and responses which are generated according to the standard specification. XACML is usually deployed and combined with previous technologies (e.g. SAML) in order to enable a more fine-grained access control mechanism. Since XACML is part of the SocIoTal access control system, section 2.1 provides a more detailed description of this technology.

1.2 IoT Access Control and Privacy related projects

The application of access control and privacy-preserving mechanisms on IoT environments is attracting the interest of the European community, in order to achieve proper security mechanisms meeting the needs of the different users involved in the resulting ecosystem. Although not focused on security issues, this section outlines some recent European initiatives that provide some foundations and proposals, which are intended to serve as a basis for the development of suitable IoT access control and privacy-preserving solutions.

1.2.1 FI-WARE

FI-WARE [14] is the cornerstone of the Future Internet PPP (FI-PPP) program, a joint action by European Industry and the European Commission. This initiative aimed to provide an open architecture and a set of specifications in order to allow developers, service providers, enterprises and other organizations to develop products that satisfy their needs while still being open and innovative. The main result of FI-WARE is an open platform which is based on the use of *Generic Enablers* (GE). These components provide reusable and commonly shared functions making it easier to develop Future Internet Applications in multiple sectors. The resulting infrastructure is intended to bring significant and quantifiable improvements in the performance, reliability and production costs linked to Internet Applications.

Within the set of GEs provided by FI-WARE, the platform supplies an Access Control Generic Enabler, which offers an API to manage authorization policies, and based on those policies, authorization decisions for access requests to REST APIs of other GEs or services. The API of the Access Control GE itself follows a REST architecture style, and uses XACML

for the specification of access control policies and their evaluation. Specifically, under the XACML standard foundations, the Access Control GE can play the role of *Policy Administration Point* (PAP) and *Policy Decision Point* (PDP). Additionally, in order to fulfill the XACML architecture, FI-WARE defines a Security Proxy GE which can optionally act as a *Policy Enforcement Point* (PEP). Furthermore, an Identity Management GE based on the use of OAuth2 tokens, which can be combined with the Access Control GE in order to provide a more fine-grained access control mechanism to services. The SocIoTal access control proposal is indented to provide a comprehensive solution for IoT scenarios, in which smart objects can be grouped composing trust bubbles or communities. Furthermore, these mechanisms could be integrated on the FI-WARE platform in order to provide an open solution to be exploited by IoT community. Indeed, section 5 of this document provides some notions about this possible integration.

1.2.2 BUTLER

The FP7 BUTLER project [15] was an initiative focused on the IoT with the main goal to enable the development of secure and smart life assistant applications, by using context and location awareness, as well as pervasive information systems. In particular, it aimed to integrate current and develop new technologies to form a “bundle” of applications, platform features and services in order to realize IoT scenarios (home, office, transportation, health, etc.) For this purpose three main challenges were addressed: how to handle personalized and dynamic demands; transparency, privacy and security in heterogeneous systems; and collective behavior modeling to maximize efficiency in order to optimize the impact that ICT systems have in influencing human behavior.

As a significant result, a platform was developed as a set of enablers and services that provide means for building context-aware applications on top of smart connected objects. It provides not only generic APIs to access resources provided by IoT devices, but also additional services such as security service, localization services, behaviour prediction services and context management services that applications can reuse to enhance the user experience and security. The BUTLER Platform is specifically conceived for IoT devices and applications. It integrates different IoT devices and communication technologies in order to provide a homogeneous access to the underlying heterogeneous networks. Specifically, access control concerns are addressed according to the usage of an Authorization Server, whose main task is to authorize a consumer to access a resource. For this purpose, the Authorization Server registers resources and delivers access tokens and security material to secure the exchange of messages by using OAuth 2.0 with specific formats and procedures for the access tokens and the derivation of security material. Furthermore, the Authorization Server implements a basic authentication protocol based on login/password paradigm, but it also enables the use of SAML v2.0.

1.2.3 IoT@Work

The main high level objective of the IoT@Work project [16] was to reduce operational costs in configuring, commissioning, and maintaining manufacturing solutions mainly by targeting disruption times due to reconfigurations or all types of changes to the system. While it was built on the results of recent research projects, IoT@Work focused on enhancing the communication and middleware infrastructure, so as to construct new self-managing and resilient networks and middleware and service oriented application architectures that are well adapted for use in a factory environment. One of the arguments for an Internet of Things (IoT) is allowing devices, machines, and objects to interact with each other without relying on

human intervention to set-up and commission the embedded intelligence. The project focuses on harnessing IoT technologies in industrial and automation environments.

The main technical objectives of the project were centred around the following goals: decoupling automation application/controller programming from network configuration and operation, integration of more self-management in a plug and work network, as well as ensuring resilience and security in running automation systems. This goal was intended to support adaptive and agile manufacturing scenarios while securing and protecting the reliability and resilience of running systems, and the integration of strong security mechanisms at the architectural level in order to avoid unauthorized access. Indeed, as a result of this, IoT@Work proposed a capability-based access control mechanism [17], through the use of unforgeable tokens of authority. A capability token refers to a value that uniquely references an object along with an associated set of access rights. This mechanism is used as a basis to build the proposed SocIoTal access control system which is detailed in section 2 of this document.

1.3 IoT Access Control and Privacy Related Work

The realization of IoT scenarios imposes significant restrictions on privacy and access control, since everyday physical objects are being seamlessly integrated into the Internet infrastructure. Specifically, IoT environments are expected to be composed by heterogeneous devices, interacting with any other entity on the Internet through different underlying network technologies. Thus, interoperability is a key factor when solutions are designed for these scenarios. Furthermore, due to the development of IPv6 technologies [18], the potential number of devices on the Internet creates the need to develop scalable mechanisms. These requirements make the application of existing security and access control solutions to these emerging ecosystems is a challenging task. In particular, current access control mechanisms need to consider efficient and proper identity management schemes to be able to cope with scenarios with billions of objects while end-to-end security is preserved. Additionally, IoT scenarios are intended to manage particularly sensitive data and any information leakage could seriously damage the user privacy. This problem is exacerbated in IoT, since any entity connected to the Internet will be able to create new information and communicate it to any other entity.

Tradition access control approaches solutions were not designed with these aspects in mind and, in the most of cases, they are not able to meet the needs of these incipient ecosystems regarding scalability, interoperability and flexibility. These challenges have attracted more and more attention from research community and recently several efforts are starting to emerge in this direction. The authors in [19] present an abstraction of the Usage Control (UCON) model [20] in IoT. The proposal is based on a trust management centre, which is responsible for updating trust values of devices and services in each usage request. The application of a fuzzy trust-based access control (FTBAC) model is proposed in [21]. This work presents a trust-based access control solution by using the linguistic values of experience, knowledge and recommendation as inputs. Then, these fuzzy trust values are mapped to access permissions to achieve access control in IoT. Moreover, the work presented in [22] provides an approach based on *Elliptic Curve Cryptography* (ECC) for key establishment and RBAC model for the definition of access control policies. They consider an inter-domain scenario in which different registration authorities are responsible for the authentication process.

Under the main foundations of ZBAC and SPKI Certificate Theory [23], the application of *Capability-Based Access Control* (CapBAC) [24] on IoT scenarios is considered in [17],

which is based on the work carried out in the EU FP7 IoT@Work project. The proposed approach is based on *Policy Decision Points* (PDPs) which are queried by services to get authorization decisions. Therefore, when the subject tries to access data of a particular resource, such user attaches the capability token to the access request. Then, the PDP is responsible for deciding whether the entity is authorized or not. The decision is based on the received capability and the internal rules defined for such resource. CapBAC is also considered by [25] for secure access to services. Once the capability is verified by the service, a protected session is established for subsequent communications. In addition, it is employed by [26], in which capabilities are exchanged in conjunction with a SHA-1 message digest, which is used to check the tampering and forgery of the capabilities. Based on the main foundations of these works, Distributed Capability-Based Access Control (DCapBAC) [27] has been recently introduced as a feasible access control approach to be deployed on IoT, even when constrained devices are used. DCapBAC allows a distributed approach in which constrained devices are enabled with authorization logic by adapting the communication technologies and data-interchange format. Specifically, it is based on *JavaScript Object Notation* (JSON) [28] as representation format for the token, the use of emerging communication protocols such as the *Constrained Application Protocol* (CoAP) [29] and 6LoWPAN, as well as a set of cryptographic optimizations for ECC.

The proposed SocIoTal access control system is based on the used of DCapBAC, along additional access control and security features in order to provide a holistic IoT access control system. The next section provides a more detailed description of it.

Section 2 - SocIoTal Access Control System for the IoT

The inherent requirements and constraints of IoT environments, as well as the nature of the potential applications of these scenarios, have brought about a greater consensus among academia and industry to consider access control as one of the key aspects to be addressed for a full acceptance of all IoT stakeholders. The proposed SocIoTal access control system is based on a combination of different access control models and authorization techniques, in order to provide a comprehensive solution for the set of considered scenarios. Specifically, while Section 2.1 describes the use of access control policies to make authorization decisions, Section 2.2 proposes to employ authorization tokens as access control mechanism to be used by IoT devices. Furthermore, Sections 2.3 and 2.4 specify the use of contextual information and trust values to be taken into account by the SocIoTal access control system.

2.1 Policy-based Access Control System

Over recent decades, a plethora of access control models have been proposed to be used on different Internet scenarios. However, RBAC and ABAC are probably the most consolidated and widely used access control models, and they are commonly deployed in a huge range of scenarios. In the case of RBAC, users are associated to roles which, in turn, are bound to specific set of privileges. RBAC also allows users to be members of multiple roles and consequently, it provides a mapping between users and roles to specify which users are allowed to play which role. Additionally, the flexibility of RBAC allows creating hierarchies of permissions and inheritance, wherein more restrictive permissions override more general permissions. However, several well-known drawbacks have been identified for RBAC model. On the one hand, it requires a mutual understanding of the meaning of roles, which could be challenging in inter-domain scenarios or unrealistic for situations in which domain boundaries are not clearly defined. On the other hand, mismatches of the set of privileges associated with a role lead to try specifying more granular roles, which is known as the *role explosion* problem. In order to provide a more fine-grained access mechanism, the ABAC model was proposed in which authorization decisions are based on attributes that the user has to prove (e.g.: age, location, roles, etc.), as well as resources and environmental properties. Thanks to the usage of ABAC, unlike in traditional RBAC models, the number of rules can be reduced, at the expense of more powerful (and complex) rules and more processing and data availability requirements. One of the main advantages of ABAC is requesters do not have to be known a priori by targets, providing a higher level of flexibility for open environments, compared to RBAC model.

For RBAC and ABAC models, the *Policy-Based Access Control model* (PBAC), enables the definition of access control rules in a policy-oriented way. PBAC, and consequently ABAC and RBAC, are usually deployed by using the *eXtensible Access Control Markup Language* (XACML) [13]. XACML is a standard, declarative and XML-based language to express access control policies, which allows specifying the set of subjects which can perform certain actions on a specific set of resources, based on attributes of them. Under the XACML data model, the definition of access control policies is mainly based on three elements: *PolicySet*, *Policy* and *Rule*. A *PolicySet* may contain other *PolicySets* and *Policies*, whereas a *Policy* includes a set of *Rules*, specifying an *Effect* (*Permit* or *Deny*), as a result of applying that *Rule* for a particular request. The *Target* sections of these elements define the set of attributes from *resources*, *subjects*, *actions* and *environment* to which the *PolicySet*, *Policy* or *Rule* are applicable. Moreover, since different *Rules* might be applicable under a specific request, XACML defines *Combining Algorithms* in order to reconcile multiple decisions. In addition, a set of obligations (*Obligations* class) can be used to notify a set of actions to be performed related to an authorization decision. Figure 1 shows the XACML Policy Language Model.

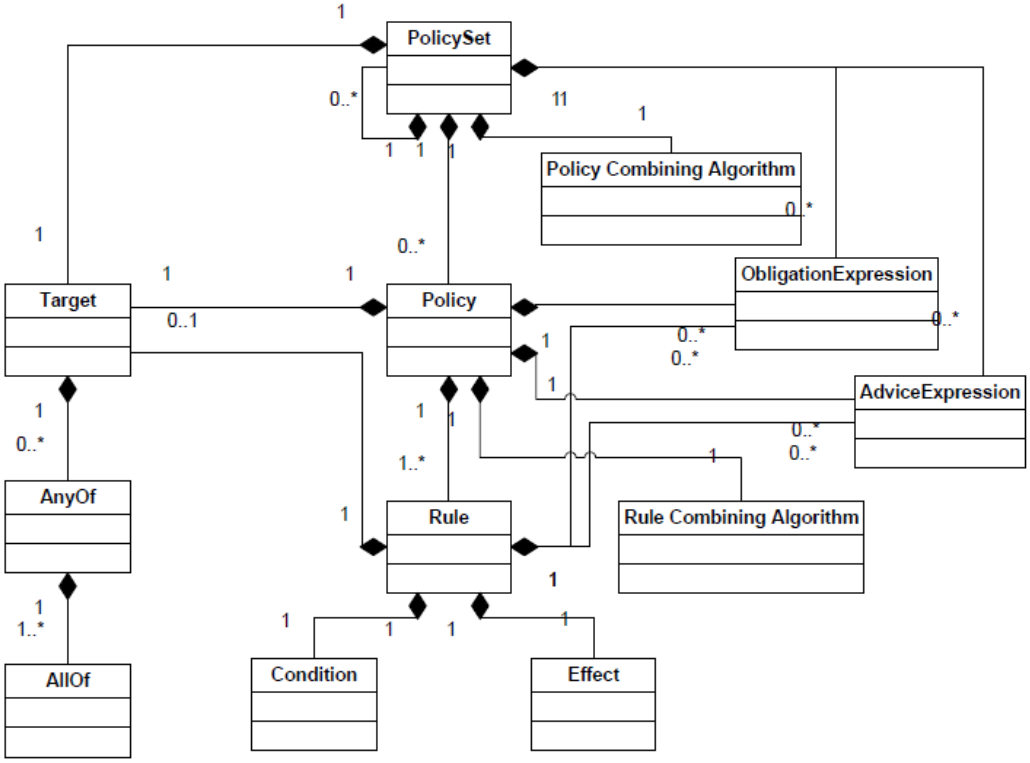

Figure 1. XACML Policy Language Model

XACML architecture consists mainly of four elements:

- *PEP* (Policy Enforcement Point): it is responsible for performing access control, by making decision requests and enforcing authorization decisions.
- *PDP* (Policy Decision Point): it evaluates applicable policies and makes authorization decisions
- *PAP* (Policy Administration Point): it is used to create a policy or set of policies.
- *PIP* (Policy Information Point): it acts as a source of attribute values

Finally, the main interactions between these components under the XACML standard are shown in Figure 2.

As already mentioned XACML supports RBAC and ABAC. The SocIoTal access control system is based on the use of XACML for making authorization decisions, which are embedded into authorization tokens to be presented to devices providing IoT service. Therefore, according to the model of XACML, IoT devices act actually as PEPs. While XACML requires moderate computation capabilities to run the engine and process the policies, it is not actually a problem because the PDP is not intended to be deployed on constrained devices, which only require enough computation resources to validate the generated tokens. According to the XACML standard, it also allows describing obligations in the policies, which enables the PDP to define directives that must be carried out by the PEP before an access is granted. This feature is important in the SocIoTal access control system since the PDP can specify a set of contextual conditions or trust values to be locally verified by the end IoT device

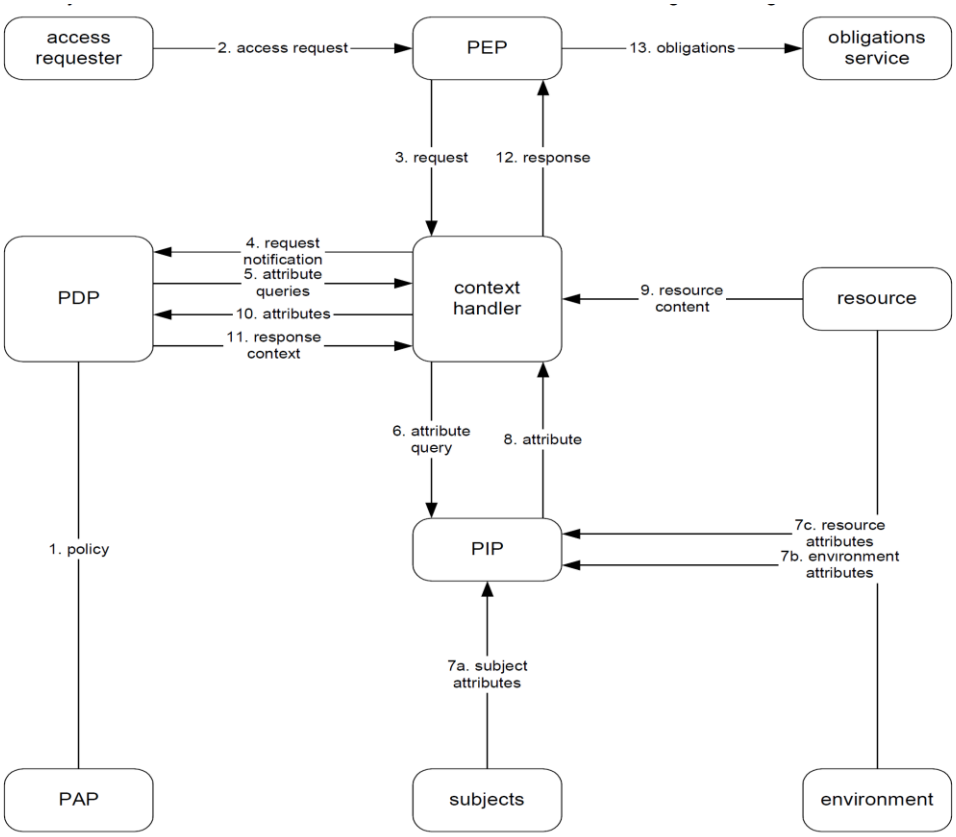

Figure 2. XACML Data-flow diagram

Furthermore, the use of JSON to represent XACML policies, requests and responses, is already proposed by OASIS [30] as a way to improve the performance of the evaluation process and to reduce the amount of information being exchanged. This feature could be exploited by the SocIoTal access control system in order to provide a more lightweight and efficient solution.

2.2 Capability-based Access Control System

Due to heterogeneous nature of IoT devices and networks, most of recent access control proposals have been designed through centralized approaches in which a central entity or gateway is responsible for managing the corresponding authorization mechanisms, allowing or denying requests from external entities. Since this component is usually instantiated by unconstrained entities or back-end servers, standard access control technologies are directly applied. However, significant drawbacks arise when centralized approaches are considered on a real IoT deployment. On the one hand, the inclusion of a central entity for each access request clearly compromises end-to-end security properties, which are considered as an essential requirement [31][32][33] on IoT, due to the sensitivity level of potential applications. On the other hand, the dynamic nature of IoT scenarios with a potential huge amount of devices complicates the trust management with the central entity, affecting scalability. Moreover, access control decisions do not consider contextual conditions which are locally sensed by end devices.

These issues could be addressed by a decentralized approach, in which IoT devices (e.g. smartphones, sensors, actuators, etc.) are enabled with authorization logic without the need to delegate this task to a different entity when receiving an access request. In this case, end

devices are enabled with the ability to obtain, process and transmit information to other entities in a protected way. However, in a fully distributed approach, the feasibility of the application of traditional access control models, such as RBAC or ABAC, has not been demonstrated so far. Indeed, as previously mentioned, such models require a mutual understanding of the meaning of roles and attributes, as well as complex access control policies, which makes challenging the application of them on IoT devices. Moreover, the impact of the potential applications of IoT in all aspects of our lives is shifting security aspects from an enterprise-centric vision to a more user-centric one. Therefore, usability is a key factor to be considered, since untrained users should be able to control how their devices and data are shared with other users and services.

As already mentioned, DCapBAC has been postulated as a feasible approach to be deployed on IoT scenarios [16] even in the presence on devices with tight resource constraints. Inspired by SPKI Certificate Theory and ZBAC foundations, it is based on a lightweight and flexible design and that allows authorization functionality is embedded on IoT devices, providing the advantages of a distributed security approach for IoT in terms of scalability, interoperability and end-to-end security. The key element of this approach is the concept of capability, which was originally introduced by [24] as "token, ticket, or key that gives the possessor permission to access an entity or object in a computer system". This token is usually composed by a set of privileges which are granted to the entity holding the token. Additionally, the token must be tamper-proof and unequivocally identified in order to be considered in a real environment. Therefore, it is necessary to consider suitable cryptographic mechanisms to be used even on resource-constrained devices which enable an end-to-end secure access control mechanism. This concept is applied to IoT environments and extended by defining conditions which are locally verified on the constrained device. This feature enhances the flexibility of DCapBAC since any parameter which is read by the smart object could be used in the authorization process. DCapBAC will be part of the SocIoTal access control system and extended with an XACML in order to infer the access control privileges to be embedded into the capability token.

2.2.1 Capability Token

The format of the capability token is based on JSON. Compared to more traditional formats such as XML, JSON is getting more attention from academia and industry in IoT scenarios, since it is able to provide a simple, lightweight, efficient, and expressive data representation, which is suitable to be used on constrained networks and devices.

Figure 3 shows a capability token example. Below, a brief description of each field is provided.

- Identifier (ID). This field is used to unequivocally identify a capability token. A random or pseudorandom technique will be employed by the issuer to ensure this identifier is unique.
- Issued-time (II). Following the notation of [34], it identifies the time at which the token was issued as the number of seconds from 1970-01-01T0:0:0Z.
- Issuer (IS). The entity that issued the token and, therefore, the signer of it.
- Subject (SU). It makes reference to the subject to which the rights from the token are granted. A public key has been used to validate the legitimacy of the subject. Specifically, it is based on ECC, therefore, each half of the field represents a public key coordinate of the subject using *Base64*.
- Device (DE). It is a URI used to unequivocally identify the device to which the token applies.

- Signature (SI). It carries the digital signature of the token. As a signature in ECDSA is represented by two values, each half of the field represents one of these values using *Base64*.
- Access Rights (AR). This field represents the set of rights that the issuer has granted to the subject.
  - Action (AC). Its purpose is to identify a specific granted action. Its value could be any CoAP method (GET, POST, PUT, DELETE), although other actions could be also considered.
  - Resource (RE). It represents the resource in the device for which the action is granted.
  - Condition flag (F). Following the notation in [35], it states how the set of conditions in the next field should be combined. A value of 0 means AND, and a value of 1 means OR.
  - Conditions (CO). Set of conditions which have to be fulfilled locally on the device to grant the corresponding action.
    - Condition Type (T). The type of condition to be verified as stated by [35].
    - Condition value (V). It represents the value of the condition.
    - Condition Unit (U). It indicates the unit of measure that the value represents. Its value could be any of those stated by [36].
  - Not Before (NB). The time before which the token must not be accepted. Its value cannot be earlier than the II field and it implies the current time must be after or equal than NB.
  - Not After (NA). It represents the time after which the token must not be accepted.


```
{
  "id": "7g3vfT_q9vTL2aQ4",
  "ii": 1415174237,
  "is": "issuer@um.es",
  "su": "zNwS5FetB4rwzSKsWwSBAxm5wDa=JgLjHU8zSnmeSFQgSG9HhdsJrE8=",
  "de": "coap://sensortemp.floor1.computersciencefaculty.um.es",
  "si": "SbUudG4zuXswFBxDeHB87N6t9hR=PBQqCN3gpu7nSkuPzDk7kaR3dq1=",
  "ar": [
     {
        "ac": "GET",
        "re": "temperature",
        "f": 1,
        "co":  [
           {
             "t": 5,
             "v": 25,
             "u": "Cel",
           },
           {
             "t": 6,
             "v": 20,
             "u": "Cel",
           }
        ]
     }
  ],
  "nb": 1415174237,
  "na": 1415175381
}
```


Figure 3. Capability Token Example

2.2.2 DCapBAC scenario

In a typical DCapBAC scenario, an entity (*subject*) tries to access a resource of another entity (*target*). Usually, a third party (*issuer*) generates a token for the subject specifying which privileges it has. Thus, when the subject attempts to access a resource hosted in the target, it attaches the token which was generated by the issuer. Then, the target evaluates the token granting or denying access to the resource. Therefore, a subject which wishes to get access certain information from a target, requires sending the token together the request. Thus, the target device that receives such token can know the privileges (contained in the token) that the subject has and it can act as a Policy Enforcement Point (PEP). This simplifies the access control mechanism, and it is a relevant feature on IoT scenarios since complex access control policies are not required to be deployed on end devices.

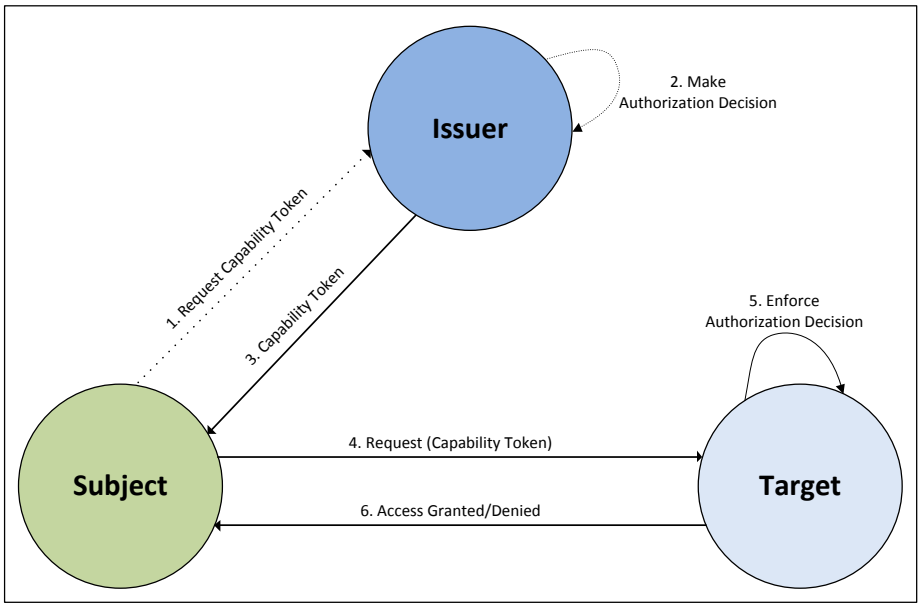

Figure 4. DCapBAC scenario overview

The basic operation of DCapBAC is shown in Figure 4. As initial step, the *Issuer* entity, which could be instantiated by the device owner or another entity in charge of the smart object, issues a capability token to the *Subject* to be able to access such device. Additionally, in order to avoid security breaches, such token is signed by the *Issuer*. In the SocIoTal access control system, this process is based on the use of XACML policies. Therefore, in the case of a “Permit” decision, a capability token is generated with that specific privilege. In addition, XACML Obligations can be used in order to embed contextual conditions to be locally verified by the target device. Once the *Subject* has received the capability token, it attempts to access the device data. For this purpose, a request is generated (e.g. by using CoAP), in which the token is attached. According to Figure 4, this request does not have to be read by any intermediate entity. When the *Target* receives the access request, the authorization process is carried out. First, the application checks the validity of the token (i.e. if it has expired) as well as the rights and conditions to be verified. Then, the *Issuer* signature is verified with the corresponding public key. Depending on the specific scenario, this key can be delivered to smart objects during the commissioning or manufacturing process, or it can be recovered from a predefined location. Finally, once the authorization process has been completed, the *Target* generates a response based on the authorization decision.

Additionally, this approach provides support for advance features, such as *access delegation*. In this case, a subject S (acting as a *delegator)* with a capability token CT can

generate another token CT' for S' (acting as a *delegated*), in which a subset of the privileges of CT are embedded. Consequently, CT' can be used by S' to get access to a resource in a target smart object. Furthermore, S can grant the right to S' for additional delegations. This feature is valuable in order to address the dynamic and pervasive nature of IoT scenarios and everyday life. For example, elderly people can provide temporary privileges or delegation of them to home help personnel to get access to their homes in case of an emergency situation. In the case of delegation, it is necessary to sign each new capability token with the corresponding subset of privileges, in order to allow a full auditability of access and avoid security breaches.

2.3 Context driven Access Control System

The high level of pervasiveness of IoT scenarios needs be addressed by a suitable access control mechanism which captures contextual data and acts accordingly. While several context-aware access control models have already been proposed, context awareness is becoming particularly important as we move towards the era of pervasive computing. On the one hand, in this new digital scene, resources are remotely accessed by users via their mobile devices anywhere, anytime. On the other hand, the environment in which the transaction is performed can be dynamically changing, and consequently, contextual information can enable more effective access control mechanism if such data is used as an input for the authorization decision.

In the literature, different definitions of context have been proposed so far. Nevertheless, most of them are not able to capture the inherent nature and requirements of emerging IoT scenarios. One of the most commonly accepted definitions is found in [37], which consider context as *"any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"*. However, this definition needs to be adapted and extended in the scope of a suitable access control mechanism for IoT, in which at least three levels of context must be considered related to the requester and target entities, as well as the context in which the transaction is being performed. In this way, the access control system should consider security-relevant contextual information, such as time, location, or environmental data from all involved entities, and use it to make access control decisions accordingly.

These challenges are exacerbated when the application of context-aware access control mechanisms is considered on ad hoc and opportunistic communities. As previously described in Section 2.2, these coalitions are spontaneously set up allowing temporary sharing transactions without the need of additional infrastructure. In addition, smart objects composing such dynamic communities, could share information with entities whose identity is not trustworthy (or unknown) in highly dynamic environments in which contextual conditions frequently change. Therefore, unlike more traditional identity-centric access control solutions, this scenarios need to consider context as a first-class element to specify access control policies. Consequently, a suitable model of context is paramount to be considered even over constrained devices. Such context model could be fed by enablers, which are described in D3.1.1 [38] and used as an input for the corresponding access control mechanism.

The SocIoTal Access Control System is able to obtain the required context by means of the Context manager component of the Security Framework. The context handling and processing is part of the work being carried in WP3. Nonetheless, the general procedure to access the context is described in Section 3.1.

2.4.1 Trust quantification

The problem of Trust and Reputation (T&R) is concerned by a context and application in which data is collected. The final reputation mechanism should utilize model that best fits to a particular scenario. The aim of SocIoTal project is to offer a platform that will provide, among other functionalities an out of the box mechanism that will base on user inputs be capable to give some outputs, i.e. the reputation. Reputation is based on observations of historic data of providers as well as correlation with geospatially close providers, taking into considerations knowledge of deployment and meta-information of IoT providers. Reputation information is used along with other information to quantify trustworthiness and establish trusted relationships among entities.

In addition, different application has different requirements: in some use cases user can manually send entities; and in other, such as in crowd-sensing applications, specific issues emerges; e.g. higher noise level is detected closer to a sound source; device is placed in wrong position while sensing. Accordingly, the T&R mechanism should be application agnostic in order to provide reputation computation for contextually different entities. These requirements can be satisfied only if we enable user to define own set of rules. In SocIoTal deliverable D1.3.1 [39], an initial set of API is defined for managing trust and reputation rules enabling CRUD (Create, Read, Update and Delete) operations on the algorithm rule table.

The envisioned Trust Manager of the security framework consists of two parts:

- Crowdsourcing component
- Rules management component

Crowdsourcing component is an algorithm component for reputation computation for IoT devices such as sensors (e.g. temperature, humidity, accelerometer, etc), mobile phones; based on statistical analysis of data coming from different sources of the same context.

Rules management component is SW component based on a state of the art algorithm [44] Fuzzy logic used to map input metric to a final reputation score. This component enables manual management of the rule table of the fuzzy algorithm.

In preliminary research, we have investigated deployment of reputation mechanism in report based Android application mojNS [40] that enables citizens to send reports directly to communal department of the City of Novi Sad municipality. Each report can later be voted from other users allowing them to express their opinion about a certain problem. Following SOTA metric for assessing rating through voting commonly used in online auctions and e-commerce websites [41][42][43], we have derived metric for computing reputation scores for reported events as well as reputation score for the users.

2.4.1.1 User behavior as measure of trust and reputation (UNIS)

Reputation scores are an important aspect of devices (e.g., smartphones) that perform the action of providing data. If a device is being used by a genuine user or is being operated in a genuine manner, a high reputation score can indicate to the SocIoTal platform that the data originating from the device can be “trusted”. On the other hand, if a device is not being used by the genuine user or is being operated in a malicious manner, a low reputation score can inform the system that the data provided should be ignored.
To assign reputation scores, the data generated by the devices can be used to distinguish a genuine user from an imposter or malicious one. In SocIoTal we consider the case of a smartphone enabling the SocIoTal users to share data from sensors in order to build citizen-

centric services. The genuine user is the person who owns the smartphone, this person is also known as the rightful user. An imposter is a person who obtains the smartphone from the rightful owner, by stealing it for example. An imposter may attempt to use the phone to perform a malicious action by trading on the trustworthyness of the genuine user. In order to prevent this, an algorithm can be used to determine whether the current user of the phone is the genuine user.

As soon as a device (e.g., smartphone) from a legitimate user is registered in the SocIoTal platform, a training phase the algorithm aims to create a model of normal usage for the genuine user through training data obtained from the available sensors. Once the model has been obtained, the testing phase (i.e. normal operation) assigns a reputation score to the phone which is obtained by comparing the current data with the model of normal usage by the genuine user. If the current data is similar to the model of normal usage, the reputation score will be higher. However, if the current data is very different to the model this is an indication that an imposter has the phone and consequently the reputation score of the device is reduced. It is essential to identify when an imposter has obtained the phone as quickly as possible in order to update the reputation score of the phone so that actions cannot be conducted using the reputation score of the genuine user. In addition, the phone may “lock” some functions or require additional authentication or a new registration to the platform.

Due to the mobility of a smartphone, one of the key activities is walking. If a smartphone can quickly identify the change from the genuine user to the imposter through the action of walking, the reputation score of the phone can be corrected quickly. In SocIoTal we will then investigate how to compute and update reputation score based on user walking modality identification.

2.4.2 Trust-aware Access Control

According to the SocIoTal security framework [45], the Trust Manager quantifies trust and reputation following the Trust Model described at section 2.4.1. The Trust Manager can be integrated in the SocIoTal Access Control mechanism in order to enable a secure information exchange between trustworthy entities. Thus, the SocIoTal Authorization system allows target devices to take into account trust values about subject’s devices before allowing them to access to its resources. During the token validation process done in the target device, the Authorization system deployed in the device checks that whether the trustworthy values about the subject or requester are not below a specific threshold value, otherwise it does not permit the incoming request. The trust values about devices are computed by the *Trust Manager* component of the framework. The *Trust Manager* can be deployed in the target device for non-constrained devices, or in another entity in case of the device has not enough hardware resources.

The trust-based access control mechanism has been designed to be used in IoT scenarios where smart objects (e.g. smartphones, sensors, actuators, etc.) can maintain social relationships composing different kinds of bubbles (i.e, Personal, Familar, Office or Community). Each bubble is made up by a set of smart objects as well as an Authorization Manager, which is responsible for generating authorization credentials (e.g. it can be deployed on a user smartphone in the case of a personal bubble) for smart objects. Furthermore, each smart object can have a Trust Manager in charge of assessing the trustworthiness degree of an entity. In the case of IoT devices with tight resource constraints (i.e, class 1 devices), the Trust Manager is assumed to be deployed in a more powerful network component.

The trust manager is an important component to assist the authorization control process between smart objects from different bubbles. On the one hand, it is used by the requester smart object to know the most trustworthy target among a set of devices providing the same

service (e.g. weather service). On the other hand, it is employed by the target smart object in order to get the trust value associated to the requester under a specific transaction. This value is used, along with the authorization credential that is previously obtained from the Authorization Manager, in order to make the access control decision.

The required message exchange can been split into four main stages, taking into account the generic steps of a trust and reputation system [46]. This scenario assumes a smart object A tries to get access a service that is provided by other smart object B in another bubble or community. Such process is shown in Figure 5.

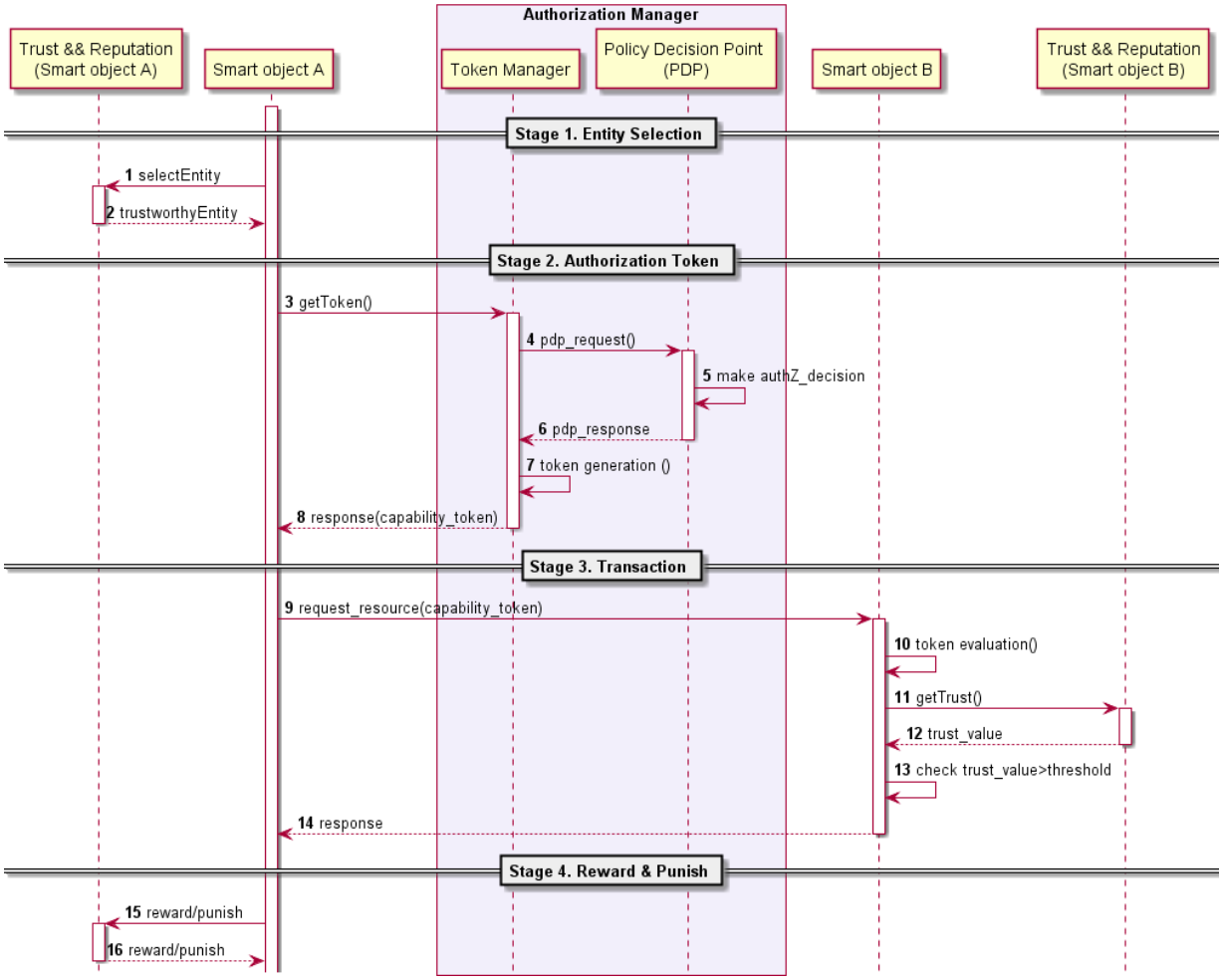

Figure 5. General Trust-aware Authorization process between two Devices

During **Stage 1**, the smart object accesses the Trust Manager in order to know the most trustworthy smart object providing a specific service. In the case of the Trust Manager is a separate component, both entities must establish a security association (e.g. based on PKI) for the communication. For this purpose, it makes a request in which a list of smart objects (being identified by a URI) is attached Smart Objects (in the case of a class 2 or more powerful device, this communication is performed within the smart object). Upon receiving the request, the *Trust Manager* calculates the trust value associated to each of the smart objects within the list, according to the model presented in the previous section. Then, the *Trust Manager* sends response back to the *Smart Object A* with the URI of the most trustworthy smart object; in this case, the smart object B. At this stage, device A could even renounce requesting device B in case the trust value of B was under a predefined threshold.

Before requesting access to device B, the *Smart Object A* needs to obtain an authorization credential **Stage 2**. For this purpose, it contacts with the *Token Manager* (a well-known entity which is trusted by the involved parties), indicating the specific resource and action to be performed on the smart object B. Then, the *Token Manager* sends a request to the *PDP* in order to know if an authorization token must be granted or not to the smart object A. The PDP evaluates the XACML policies using the policy engine and makes an authorization decision. In case of a PERMIT decision, the PDP generates an XACML obligation with a threshold trust value, in order to specify the risk associated with that access control decision. Then, the *Token Manager* generates an authorization credential or token and includes the threshold trust value as a new condition to be verified by the target device, i.e. the smart object B. Before delivering the authorization credential, the *Token Manager* signs the tokens. This signature is validated by the Smart Object B when it receives the token. An example of an authorization token including the trust threshold condition is shown below.

```
{
  "id": "Jd93_jZ8Ls5V0qP",
  "ii": 1412941013,
  "is": "coap://tokenmanager.um.es",
  "su": "aB4wSICIXC1pm2pkW9YMPQyFudc=CPhYdgOAQwc0YgURwP1q02WSv=",
  "de": "coap://smartObjectB.um.es",
  "si": "TqZaXuxZ5dmZU6k3PtiWwI3NrjH=7u5By5OHzl0Otq4TmkrZU2JPd=",
  "ar": [
   {
      "ac": "GET"
      "re": "position"
      "co": [
         {
            "t": 5,
            "u": trust,
            "v": 0.7
         }
      ]
   }
  "nb": 1412941013,
  "na": 1412941456
}
```

Figure 6. Trust-aware Capability Token example

As can be seen in the token example of the Figure 6, the access rights part 'ar' limits the access to the action *GET* over a resource *position*, as long as the trust threshold condition is over 0.7.

Then, during the **Stage 3**, the *Smart Object A* uses the authorization credential for access to a service being hosted on the smart object B. Although it is not showed in the diagram this step also requires authentication prior the key exchange is carried out. This authentication could be done using traditional means or, if supported by the involved devices, using the privacy-preserving authentication based on the anonymous credential system provided by the IdM component of the Framework. Additionally, a session key is generated to exchange securely the capability token that is used to get access to a specific resource. When device B receives the access request, it checks whether the token is valid or not checking the signature and the expiration date. Then, as the access rights are contained in the token, the device B checks them against the requested action, including the conditions described in the token. Besides, since the device have the final say, it can check the current context, like for instance its battery level, before allowing device A to access to its resource.

In addition, once the capability token is evaluated by the smart object B, the access control process at device B takes into account the trust value associated to the requester device, in this case the smart object A. Once the smart object B receives the trust value, it verifies this number is greater than the threshold trust value, which is contained in the capability token. If that condition is fulfilled, the request is accepted and the service is provided to the smart object A. It should be pointed out that the Trust Manager assesses the trustworthiness of devices dynamically upon changes that affect the set of trust criteria. The specific criteria to be taken into account to compute trust are outside of the scope of this document, and they will be covered in D2.3.

Finally, during the **Stage 4** the reward and punish operation takes place. Depending on the satisfaction degree of the smart object A regarding the transaction, a set of evidences about the interaction is sent to its Trust Manager A in order to update the trust value associated to the smart object B. Further details about security concerns (e.g. DoS attacks or malicious nodes) will be given in D2.3.

Section 3 - SocIoTal framework interactions for Privacy and Access Control

This section extends the first design and description of the SocIoTal security framework introduced in Deliverable D2.1 [45]. In the IoT paradigm, unlike the current Internet, interaction patterns are often based on short and volatile associations between entities without a previously established trust link. Furthermore, given the dynamic and distributed nature of IoT, it is necessary to consider more flexible data sharing models in which some information can be shared with a group of entities or a set of unknown receivers and, consequently, not addressable a priori. In addition, due to the pervasive character of envisioned IoT scenarios, the way in which this information is disseminated must consider contextual data where sharing transaction is going to be performed. Consequently, the SocIoTal security framework is not just an instantiation of the security functional group of the IoT *Architecture Reference Model* (ARM) [47] from the IoT-A EU project [48], but it actually extends it by defining additional components in order to address the inherent requirements of IoT scenarios. Figure 7 shows such ARM-compliant architecture, in which the security functional group is detailed. In addition to the five security functional components of ARM, this framework proposes an extension of it with the inclusion of two new functional components: Context Manager and Group Manager. The former has been added in our framework to enable the rest of security components to cope with the pervasive and ubiquitous nature of IoT. The latter has the aim of dealing with more flexible and data sharing models in which a set of entities can be involved.

While D2.1 provides a description of the SocIoTal security framework mainly focused on the Identity Management (IdM) functional component, this document explains the main interactions and the components involved for the realization of the SocIoTal access control system. This also provides a description of the different processes in which the new functional components are employed, as well as their application in order to manage communities and bubbles in a secure and privacy-preserving way. Specifically, for each process, the following sections show the sequence diagram and the *Application Programming Interface* (API), which is required for the context-aware and privacy-preserving identity selection, access control, and group sharing mechanisms. Additionally, an example security policy for each of these processes is also provided.

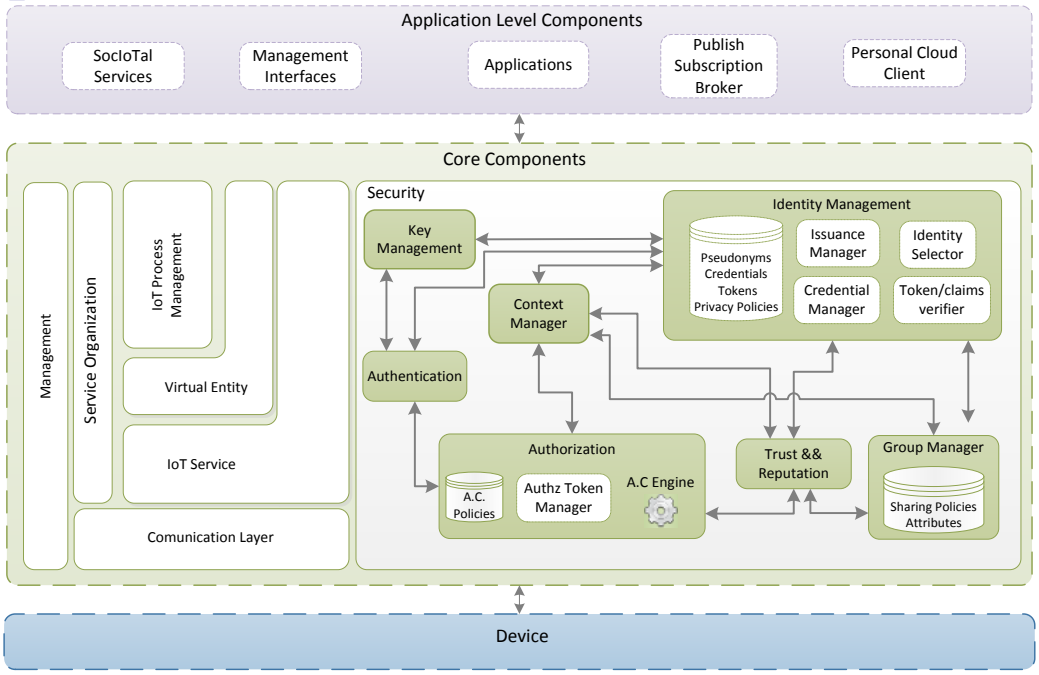

Figure 7. SocIoTal security framework

3.1 Context-aware security framework

3.1.1 Context Model

Taking as a starting point the already mentioned context definition [37] as *"any information that can be used to characterize the situation of an entity…"* and considering also as an entity every item or actor, such as a person, a device, a place or a smart object, providing relevant information within any IoT interaction, this subsection pursues to design a modelling structure to define and characterize every entity belonging to SocIoTal environment, while it serves as the baseline to manage all the generated context information: the SocIoTal Context Model.

In order to provide such context model, SocIoTal first defines and describes an entity together with its context information for, later on, provide to the user with a standardized way to access and manage this information. To achieve this, the IoT approach to virtual entities (as shown in D1.2.1 [49]) and the OMA NGSI Context Management specifications are the references on top of which SocIoTal will build this model.

3.1.1.1 IoT-A Virtual Entities definition

The IoT-A Domain Model [47] presents a Unified Modelling Language (UML) diagram where the entity is the main focus of interactions by humans and/or software agents. The purpose of these models is to create a structure for the respective information and relationships presentation. Within this structure, an entity is a virtualization of "things" in IoT (here so called Virtual Entities -VE) and could be a person, animal, physical location or any other object in the physical world. The ***"entity"*** is the main focus of interactions by the user and software agents in the IoT world and can be considered as the main ***"Context Producer"***. In addition to the profile properties, such as name and identifier, the virtual entity model defined in IoT-A [47], which has already been introduced in SocIoTal [49], includes properties to describe location of an entity, and description elements for features of interest for an entity (features

that can be observed by a sensing mechanism or can be changed/controlled by an actuation process).

In a very simplified way, the model that defines an entity is divided into two main blocks:

- *Entity Definition*, that encloses the information needed to name and identify the entity, as well as the entity type.
- *Set of attributes*, including all the different set of data the entity can somehow offer. These attributes also make up the context of the entity. The attributes are composed by its name, type, metadata (definition of what the attribute includes and how it can be read) and the values themselves.

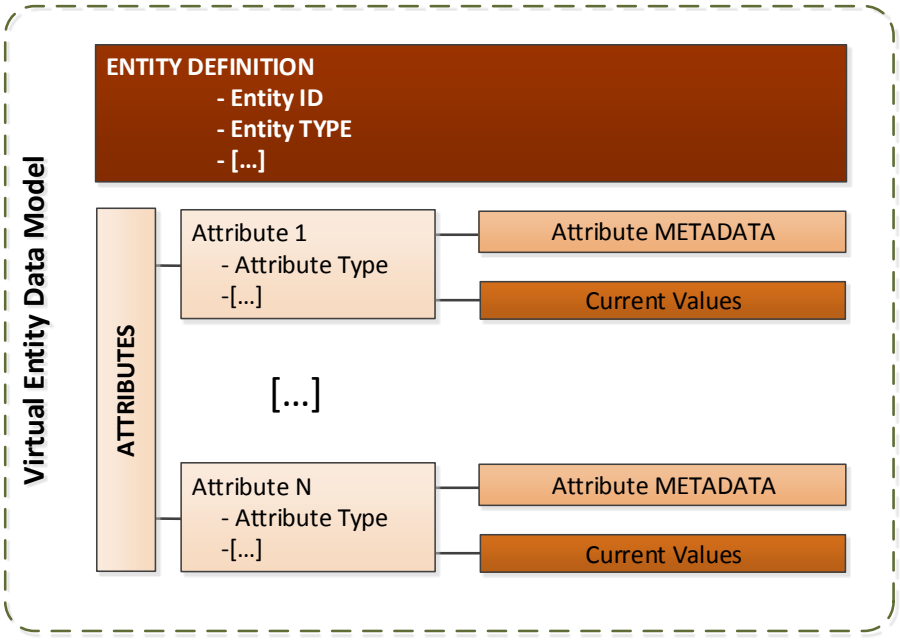

Figure 8. IoT-A simplified Virtual Entity Model

IoT-A, within its Virtual Entity model, introduces also the concept of “association”: whilst the service aspects are captured by the IoT service model [47], which exposes resource functionalities and provides a standardized access mechanism to the resource capabilities, these IoT Services are associated to Virtual Entities through the associations, linking the capabilities of the resource (or resources) mapped in a given VE with its attributes, as shown in Figure 9.

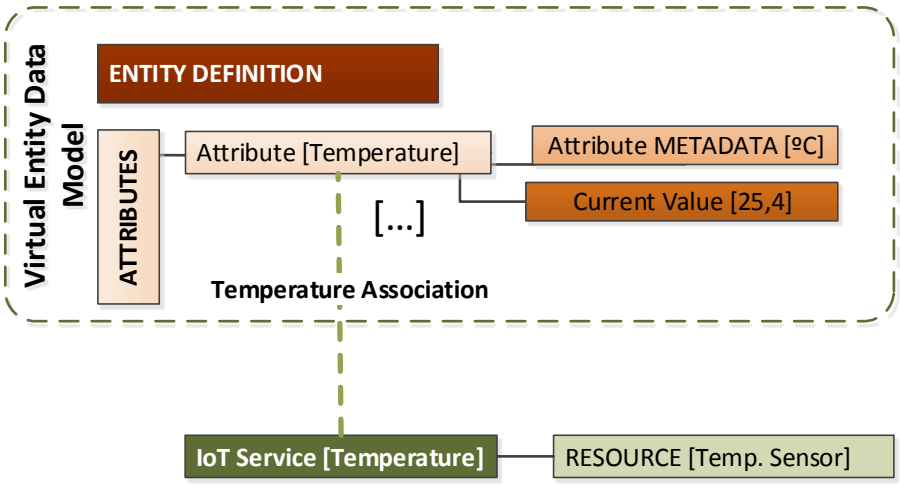

Figure 9. IoT-A associations: VE attribute and related resource IoT Service

3.1.1.2 OMA NGSI Context Management

In OMA Context Management specification [50], Context Entities are entities described by Context Information, so these will be defined through the Context Information Model. Figure 10 gives some examples of entities that can be used as Context Entities.

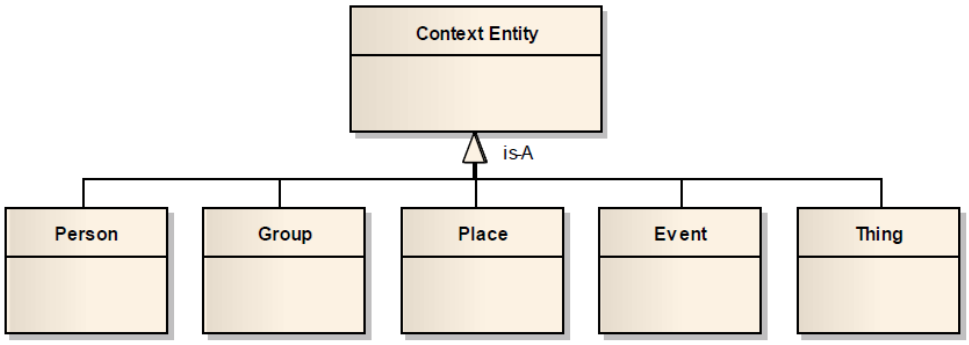

Figure 10. Examples of Context Entities

The Context Information Model details how Context Information is structured and associated to Context Entities in order to describe their situation. In this model, Context Information is organized as Context Elements, which contain a set of Context Attributes and associated metadata (Figure 11).

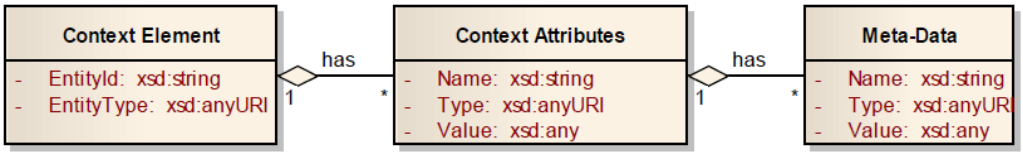

Figure 11. Context Information Model

- ***Context Element (Entity ids and types)***: Context Entities are identified using an entity identifier (EntityId). The optional entity type may be needed when the EntityId does

not contain type information or when the EntityId is only unique per entity type. The entity type is defined as a URI, and thus can be for example an ontology reference (as URL) or a namespace (as URN). The definition of EntityId and Type values is out of scope of this specification.

- ***Context attributes and attribute domains***: a context attribute represents atomic Context Information. An attribute is defined as a set of information, namely a name, a type, a value and a set of associated metadata (e.g. timestamp, expires, source). The attribute value is expressed as any content, including strings or opaque objects represented using standard formats.
  An attribute domain represents the grouping of multiple attributes. Attribute domains allow requestors to specify a set of attributes of interest using a single string as attribute domain name.

The definition of attribute names, attribute types and attribute domains, as well as the different types of entities is out of scope of OMA, depending this on the SocIoTal project.

Together with the Context Information Model, the OMA Context Management specifies interfaces in order to access and manage the context information and the context entities; ***NGSI-9*** and ***NGSI-10*** Interfaces with the following main functions:

- Register and retrieve the availability of Context Entities and/or Context Information.
- Update Context Information in accordance to a specified Context Information Model.
- Query for and subscribe to Context Information about Context Entities.

These specifications will be further presented when defining the SocIoTal Context Manager enabler, within WP3.

3.1.1.3 SocIoTal Context Modelling example

IoT-A Virtual Entities and OMA Context Entities are quite similar ideas in their basis. SocIoTal will use the IoT-A approach to define the entities mapping the different resources and context producers involved in its scenarios and the OMA specifications to implement its associated context management.
Figure 12 shows, according to IoT-A VE data model, a Weather Station entity, as a context producer, taken from SocIoTal D1.2.1 as part of one of the SocIoTal's scenarios and represented by an identifier (VirtualWS01) and three attributes: Temperature01, Pressure01 and Humidity01. These three attributes are described by their corresponding metadata, while their current values (context information) are provided by three resources through their corresponding associated IoT services: WS01_TempServ01, WS01_PressServ01 and WS01_HumServ01.

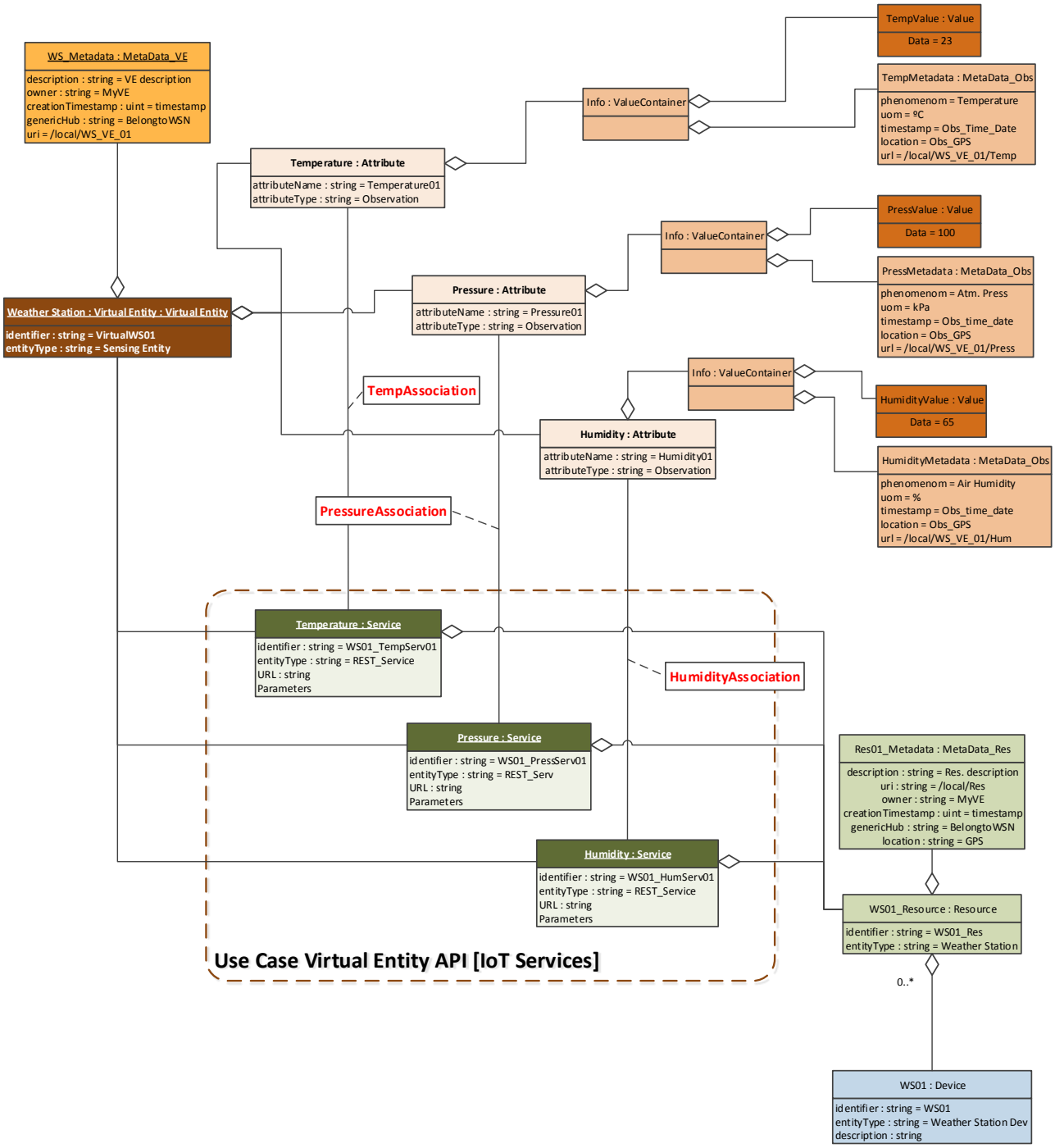

Figure 12. Weather Station VE (IoT-A definition)

Figure 13 presents the JSON implementation, following OMA specifications, of the Context Entity associated to VirtualWS01 Weather Station. For attribute definition and types, SensorML ontology specification [51], as a dictionary, is used as the main reference. Where this ontology falls short, Semantic Sensor Network (SSN) ontology [52] is used as fall-back reference.

```
{
   "contextElement": {
      "id": "SocIoTal:SAN:WeatherStation:VirtualWS01",
      "type": "urn:x-org:sociotal:resource:device",
      "isPattern": "false",
      "attributes": [{
         "name": "AmbientTemperature",
         "value": "25.44",
         "type": "http://sensorml.com/ont/swe/property/AmbientTemperature",
         "metadatas": [{
            "name": "DateTimeStamp",
            "value": "20141030T113343Z",
            "type": "http://sensorml.com/ont/swe/property/DateTimeStamp"
         },
         {
            "name": "Unit",
            "value": "celsius",
            "type": "http://purl.oclc.org/NET/ssnx/qu/qu#Unit"
         },
         {
            "name": "accuracy",
            "value": "0,5",
            "type": "http://sensorml.com/ont/swe/property/QuantitativeAttributeAccuracy"
         }]
      },
      {
         "name": "HumidityValue",
         "value": "59",
         "type": "http://sensorml.com/ont/swe/property/HumidityValue",
         "metadatas": [{
            "name": "DateTimeStamp",
            "value": "20141030T113343Z",
            "type": "http://sensorml.com/ont/swe/property/DateTimeStamp"
         },
         {
            "name": "Unit",
            "value": "percentage",
            "type": "http://purl.oclc.org/NET/ssnx/qu/qu#Unit"
         },
         {
            "name": "accuracy",
            "value": "1",
            "type": "http://sensorml.com/ont/swe/property/QuantitativeAttributeAccuracy"
         }]
      },
      {
         "name": "AirPressureValue",
         "value": "102.3",
         "type": "http://sensorml.com/ont/swe/property/AirPressureValue",
         "metadatas": [{
            "name": "DateTimeStamp",
            "value": "20141030T113343Z",
            "type": "http://sensorml.com/ont/swe/property/DateTimeStamp"
         },
         {
            "name": "Unit",
            "value": "kPa",
            "type": "http://purl.oclc.org/NET/ssnx/qu/qu#Unit"
         },
         {
            "name": "accuracy",
            "value": "1",
            "type": "http://sensorml.com/ont/swe/property/QuantitativeAttributeAccuracy"
         }]
      }]
   }
}
```


Figure 13. Weather Station CE (OMA Context Information Model implementation)

Another example of context producer, related to the SocIoTal envisioned scenario, is the one attached to a citizen mobile phone. By relying on the enablers defined in [38], the produced

context is defined in terms of user indoor localization and presence of F2F relations. The defined Context will have then the following attribute (according to SensorML attribute list):

- Location: providing the position of the device according to specific indoor localization coordinates;
- Person: providing a list (eventually empty) of pseudonyms representing the id of devices for which a F2F relations was detected;
- DateTimeStamp: a temporal reference for when the previous two context attribute were obtained.

A representation of a SmartphoneContext VE, providing such context information, along with the element characterising the produced context, is provided in Figure 14, while the OMA Context Information Model implementation is provided in Figure 15.

As an example, the production of such context information, characterising the three dimensions of context interesting for SocIoTal (i.e., time, space and environment, in terms of F2F relations) could be exploited in case the citizen mobile phone is accessed by an authorized user and device to expose a Noise:Service through its internal microphone sensor. In this case, in order to preserve privacy and allow access to the service only when the detected context matches a given indoor location and no close F2F interactions are taking place, the production of this context is shared and used by the other modules of the Privacy and Security framework (e.g., Context Manager and Group Manager).

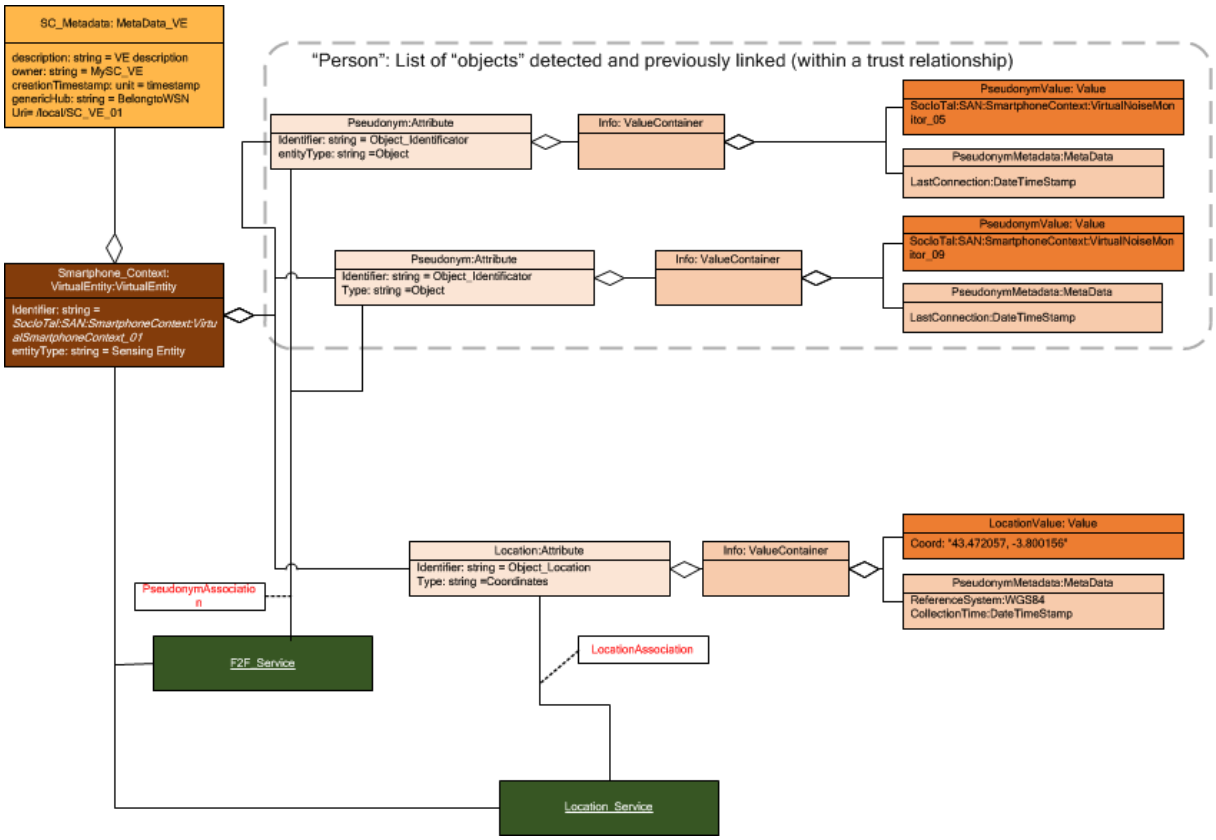

Figure 14. SmartphoneContext VE (IoT-A definition)

```
{
   "contextElement": {
      "id": "SocIoTal:SAN:SmartphoneContext:VirtualSmartphoneContext_01",
```

```
      "type": "urn:x-org:sociotal:resource:device",
      "isPattern": "false",
      "attributes": [{
         "name": "Pseudonym",
         "value": "Device_ID_01",
         "type": "http://sensorml.com/ont/swe/property/pseudonym",
         "metadatas": [{
            "name": "DateTimeStamp",
            "value": "20141030T113343Z",
            "type": "http://sensorml.com/ont/swe/property/DateTimeStamp"
         }]
      },
      {
         "name": "Pseudonym",
         "value": "Device_ID_02",
         "type": "http://sensorml.com/ont/swe/property/pseudonym",
         "metadatas": [{
            "name": "DateTimeStamp",
            "value": "20141112T111123Z",
            "type": "http://sensorml.com/ont/swe/property/DateTimeStamp"
         }]
      },
      {
         "name": "Location",
         "value": "43.472057, -3.800156",
         "type": "http://sensorml.com/ont/swe/property/Location",
         "metadatas": [{
            "name": "WorldGeographicReferenceSystem",
            "value": "WGS84",
            "type": "http://sensorml.com/ont/swe/property/WorldGeographicReferenceSystem"
          },
          {
           "name": "DateTimeStamp",
           "value": "20141124T104543Z",
           "type": "http://sensorml.com/ont/swe/property/DateTimeStamp"
         }]
      }]
   }
}
```

Figure 15. Smartphone Context CE (OMA Context Information Model implementation)

3.1.2 Context manager within the security framework

The role of the Context Manager within the SocIoTal security framework is to provide the context notion to the different architecture modules in order to support their activities. The context can be managed globally in a backend and locally in devices or gateways. SocIoTal security components are able to made security decisions taking into account their local context along with the context coming from the backend. Figure 16 shows the subcomponents of the envisioned Context Manager, which are described in D2.1.

The device Context Manager is envisaged to be NGSI compliant and deal with the local context and be able to obtain context from the global Context Broker. The Context Manager can publish raw context events (e.g. coming from the own sensors and actuators) or elaborated events (as a result of the events processed by the context manager engine) to the backend context broker. The context broker is usually placed in the Cloud or in a Data centre in order to maintain and process context events coming from different devices. In addition to publish events, the Context manager can also accept events from any NGSI compliant Event Producer. Thus, the devices could be notified by the context broker with new context events needed to take security decisions.

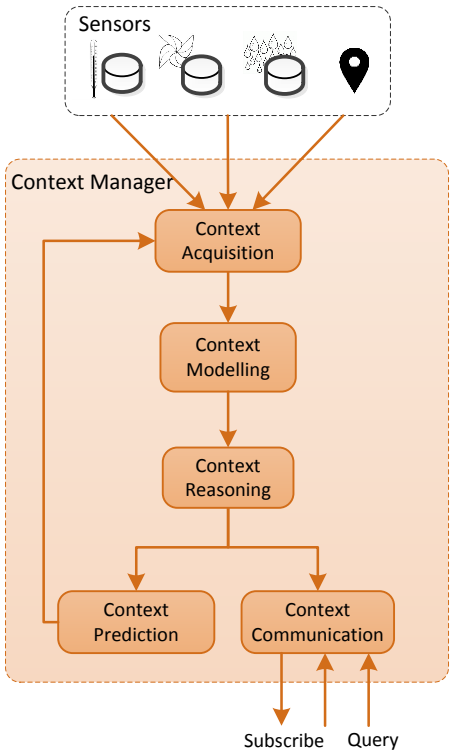

Figure 16. Context Manager subcomponents

The Complex Event Processing [53] technology provides means to processing events across different sources and layers, identifying the most meaningful events, analysing their impact, and taking derived action in real time. It offers a way of filtering, aggregating and merging real-time data from different sources. It allows discard and use value-added data that are relevant for each application. IoT world and concretely sensor network applications (RFID reading, scheduling and control of fabrication lines, air traffic) can be beneficed of this approach.

CEP technology can be used, along with other Context reasoning techniques, in the Context engine placed within the Context Manager. CEP can be used to process the input data and generate an aggregated and smaller set of output data to be delivered in the upper level or to the Context Broker. In the CEP paradigm events are published by Event Producers and subscribed to by Event Consumers. Battery issues, network disruptions can lead IoT devices to be unreachable. The Context Manager could be optionally endowed with a local storage in order to act as a cache and hold last context to be available later on. The Context reasoning mechanisms, including CEP, as well as the way of processing the context events will be described in deep in deliverable D3.2.

CEP allows setting rules to deal with the context, defining event types, and configuring output event recipients. Event type's definitions are envisaged to be compliant with the context model definition defined in previous section, which is based on NGSI. The rules of the local Context engine allow the components of the security framework take security decisions according to the actual context. Different security components, namely, the Identity Management, Authorization component, Group Manager and the Trust && reputation can be subscribed to the context engine and take security decisions depending on the inference of the rules. Thus, each time a rule is activated the security components can be notified with the consequent information derived in the rule. It can be done implementing the corresponding action Listener of such context rule.

Figure 17 shows the way different security components of the security framework can access to the Context Manager to make security decisions accordingly.

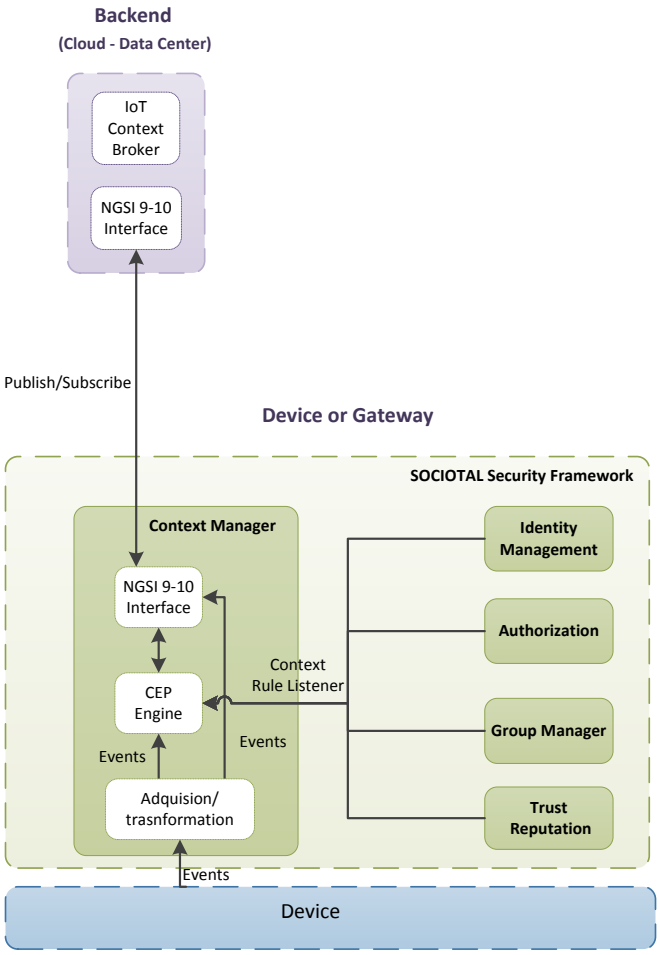

Figure 27. Context Manager main interactions

The device Context Manager can be registered as an NGSI context provider in the backend IoT Context Broker, it can be done by calling the NGSI-9 *registerContext* operation. After context registration, the Context Manager can send events by calling the NGSI-10 *updateContext* method of NGSI interface in the backend. The Context broker acts as event consumer.

Additionally, the device Context Manager can be subscribed to receive context from the IoT Context Broker. It should be noted that the communication with the Context Broker could be subject to security mechanisms to ensure only trustworthy devices interact with it. These considerations will be addressed later on in the project. Thus, the device can take security decisions locally based on the context when it is necessary. In this case, the Context Broker acts as context provider. The main security components, can access to the inferred context data in the local Context engine. It is done by accessing to the activated rules by an action-listener mechanism.

The following sections will show the way each security component handles and uses the context obtained from the context manager.

3.2 Identity Management

The Identity Management component of the security framework is an anonymous credential system that ensures user privacy and minimal disclosure of personal information when accessing IoT services. It is based on already existing implementations of anonymous credential system like Idemix [11] or U-Prove [9] but adapted to IoT scenarios. In order to address the SocIoTal uses cases, where mobile smartphones are usually employed.

Having part of the IdM deployed in end-users smartphones allows to control and manage personal data in the smartphone, defining partial identities and describing rules defining the way its personal information is disclosed according to the context. In this kind of scenarios, users could interact directly with others peers member of communities and bubbles to share information and access each other their IoT services, so that user devices could act as consumers and producers of information. It means that the IdM deployed in the device should be provide means to select the partial identity according to actual context, and run the presentation process to demonstrate its anonymous credentials without revealing unnecessary private information.

The SocIoTal IdM, unlike traditional IdMs such as Fi-ware, addresses a small set of functionalities, focusing on the authentication process and the privacy preserving mechanism that enable users to use different partial identities to access target devices according to the context. Other IdM functionalities used in traditional web contexts, such as user profile management and SSO (Single Sign On), are left in SocIoTal to open existing solutions, which already provide successfully those functionalities.

Figure 18 shows the way the partial identity selection as well as the authentication based on anonymous credential system is done by the SocIoTal IdM. The authentication is done by running the presentation process. The full process is omitted here since it has been already defined in section 4.1.3 of deliverable D2.1.

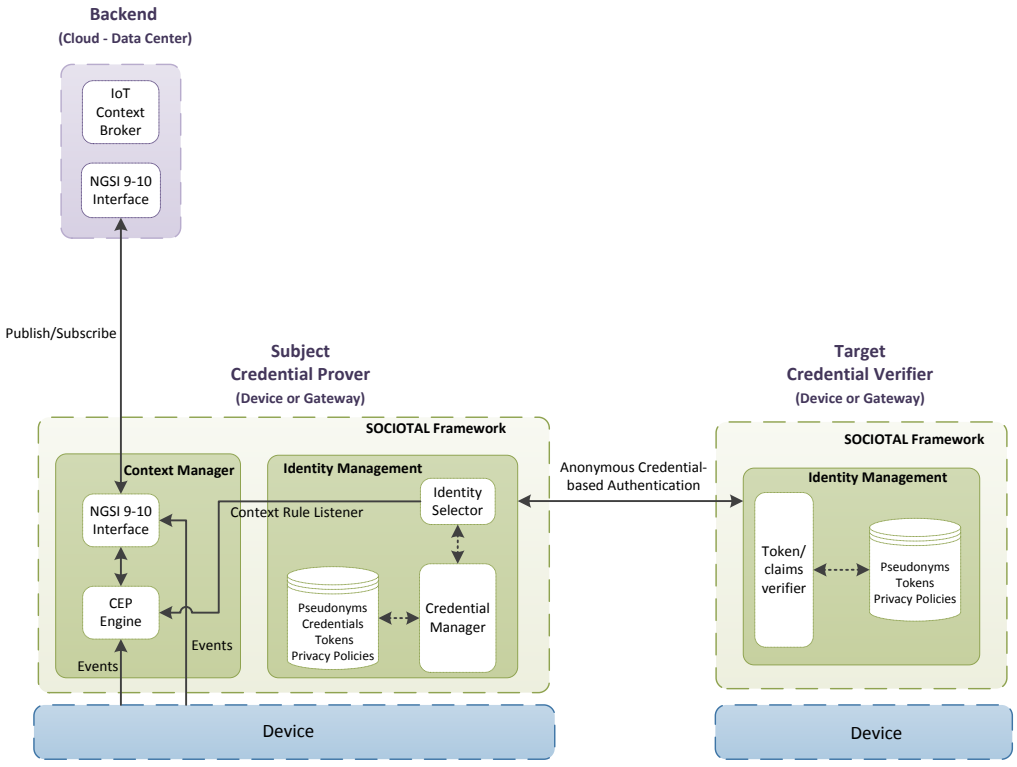

Figure 18. Context Based partial identity selection and authentication

As can be seen in the figure, prior performing the authentication, the user or smart object can have rules defined to choose the most suitable partial identity (i.e. credential) to use in each particular context. To perform the Identity selection, the Identity selector is notified by the context manager each time a particular context has changed. It is done by snooping to the context rules the Identity Selector is interested in.

It should be noticed that the Context Manager could have obtained the context information from both, an external context provider (like the context broker), or taking into account only device local context information. Details about context handling, aggregation, filtering, context inference, and so on will be defined in deliverable D3.2.

3.2.1 Sequence Diagram

This section describes the privacy-preserving IdM process about adopting the proper partial identity according to the context, in order to authenticate against a target device using such a partial identity, ensuring minimal disclosure of private data. The sequence diagram shows the overall process:

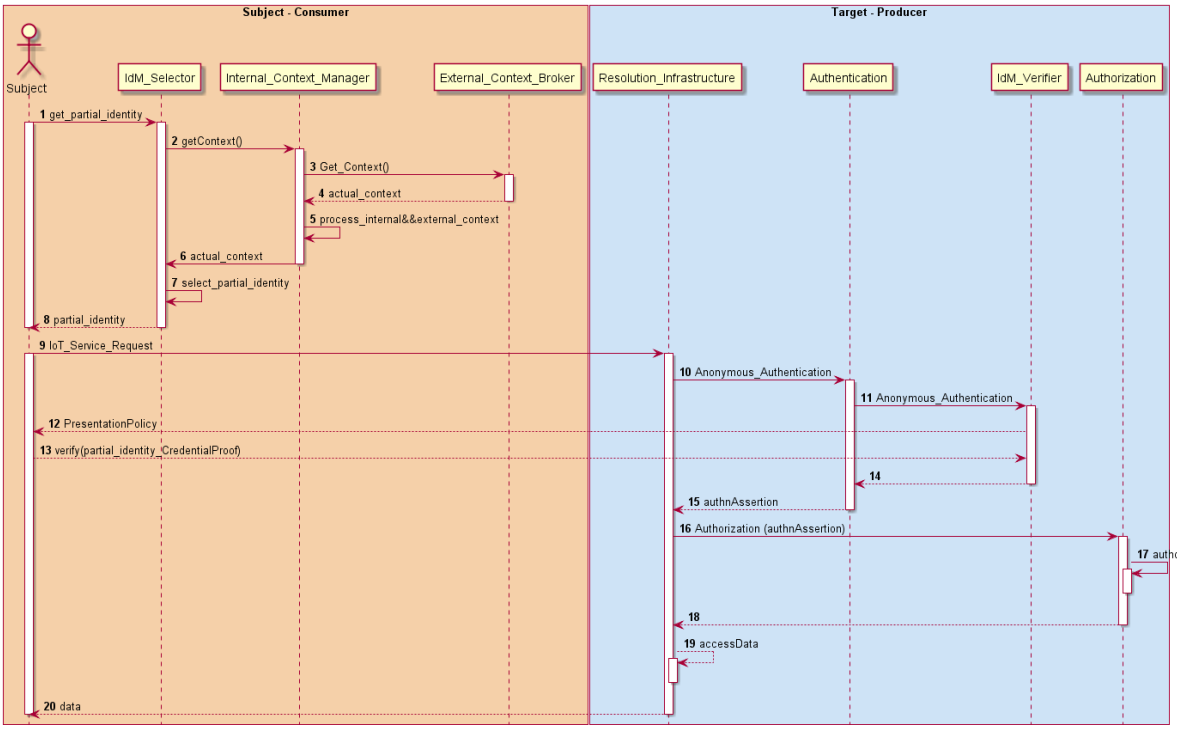

Figure 19. Identity Selection sequence diagram

In a first stage, a subject or consumer wants to access to a protected target device acting as producer. In a first stage the subject wants to select the proper partial identity (i.e. the anonymous credential) to use against the target device, in the actual context. To this aim, the IdM selector on an event of new request, prior running the identity privacy rules to select the most suitable credential, obtains the context relying on the Internal Context Manager of the device (step 2). In this step the context manager checks context values obtained directly from the own device measurements. In addition, the context manager can be configured to access (either subscription or request/response) to an external context broker where context data from other trusted devices and users are maintained. This interaction is based on NGSI standard as explained in previous section. Then, the Context Manager unifies its own context and values coming from the context broker and passes this information to the *IdM Selector*.

The *IdM Selector* evaluates the privacy identity rules, which take into account the context, to choose the best partial identity to adopt in such a situation.

It should be noticed that the partial identity selection process can be done also later on, during the presentation process (step 12), instead of perform it at the beginning. In such a case the Identity Selector could also take into account the requirements from the target device regarding the attributes to be contained in the anonymous credential.

In the second stage (step 7) the subject uses the selected anonymous credential to obtain the data from the target device B. To this aim, the subject must be authenticated using the anonymous credential and following the presentation process described in Deliverable D2.1 section 4.1.3. As a result, in case of a successful authentication, the authorization stage takes place, which makes a decision to allow subject A to access the requested resource/service. It should be noticed that the authorization stage has been simplified in this diagram since it is already described in deep in Sections 2 and 3.

3.2.2 Interfaces Definition (API)

- ***Authenticate*** *(credential): assertion*

  This method allows authenticating users and smart objects based on the provided credentials. The credential can be in form of login/password, shared key, digital certificate. As a result, an assertion is generated to be used afterwards to declare that a specific subject was authenticated successfully by the Issuing authority.

- ***verifyAuthNAssertion*** *(authn_assertion): valid/invalid*

  This method of the Authentication component can be invoked by entities that can verify that an authentication assertion obtained previously is correct.

- ***getCredential*** *(Attributes): credential*

  This method is invoked by a subject to obtain a credential from an Issuer. The credential can be an attribute container representing a real o partial identity. The attributes values to be included in the credential are needed. Credentials and the attributes follow a particular structure given by the credential specification.

- ***createCredentialProof*** *(credential, ProofSpecification):proof*

  This method is use to generate a proof of having credential given a credential. The credential proof structure, which defines the set of attributes and statements the user want to prove, has to be specified. It returns a cryptographic proof of possession of a credential. The credential proof structure follows the Idemix proof specification format.

- ***verifyCredentialProof*** *(credentialProof):valid/invalid*

  This method is used by the verifier (the service being accessed) to ensure that the subject satisfices the required identity attributes. The credential proof of the subject to be validated is passed as parameter. The method returns a Boolean indicating if the proof is valid or not.

- ***createPseudonym*** *(assertion, context): pseudonym*

This method is used to generate a pseudonym to a certain subject. Before obtaining a pseudonym, the subject should have been previously authenticated, that is why the authentication assertion obtained in the authentication is needed to create the pseudonym.

- ***resolvePseudonym*** *(pseudonym):identityID*

  This method is used to resolve the real identity ID, given a pseudonym.

3.2.3 Privacy Policies

The Identity Management component is equipped with an Identity selector sub-module that enables users to manage its partial identities and choose automatically the proper partial identity according to the context. The Identity Selector is a rule-based tool that allows describing the behaviour of the user device when accessing certain IoT services, under certain circumstances. This module can be used, in turn, by high level graphical applications that can facilitate users to manage their partial identities in different context and discharge them with the task of defining low level rules. The rules can be deployed and executed either in the Identity selector component or directly by the Context Engine which holds and manages the context used in the rules.

The Identity selector is being designed from scratch in the scope of the project, and will be in a close relationship with the context manager definition in WP3, since rules will rely on the context data model in order to describe the behaviour, that is, choose the proper credential in a particular context.

The Identity Selector allows users to choose the partial identity with respect to the current situation. It includes the target IoT service to be accessed, the current partial identity, the community or bubble, the target service and the context (like position, time…)**.**

Although it is initially out of the scope of the project, it is worth mentioning that an identity negotiation process could be considered in case the IoT service requires more private identity data from the user than he wants to disclose in the actual situation. This conflict can be solved with a negotiation between this service and the user. In this regard, a comparison from the service's and user's security and privacy policy could be tackled to suggest the best partial identity to adopt.

The following piece of code shows an example of a high level rule definition that defines the partial identity to be enforced, taking into account the context, the current community as well as the target service. These rules could be defined by users through an automated tool or manually. As can be seen, the consequent of the rule indicates to the Identity Selector which partial identity has to be deployed (among the ones already defined by the user). To this aim, there is a mapping between the partial identities defined in the rules of the Identity Selector module and the credentials that represent such partial identities that are managed by the Credential Manager.

```
IF
   location=x AND time=y AND targetService=z AND community=c

THEN
   adoptedCredential="partialIdentityA"
```

3.3 Access Control

The Authorization functional component of the SocIoTal security framework is based on a combination of access control models and techniques, which were previously described in section 2. In order to accomplish with the main features of the proposed system, contextual information is a key aspect to be considered when making access control decisions. In the same way that the Context Manager was considered for partial identities selection, this component is expected to be deployed into IoT devices (e.g. smartphones) in order to drive the access control logic to protect resources being accessed. According to the SocIoTal access control system, a subject entity gets a capability token in order to get access data from a target device. This token is usually generated by an *Issuer* entity which makes access control decisions that are embedded into the token. Therefore, when the subject entity tries to get access to a resource being hosted in a target entity, it provides the capability token previously obtained. Such token can contain contextual restrictions to be locally verified when the token is evaluated by the target device. At that moment, the target device can use contextual information from its local Context Manager, as well other data stemming from the IoT Context Broker deployed in the Backend. This process is shown in Figure 20, in which context information from the Context Manager is used by the target device when verifying the capability token.

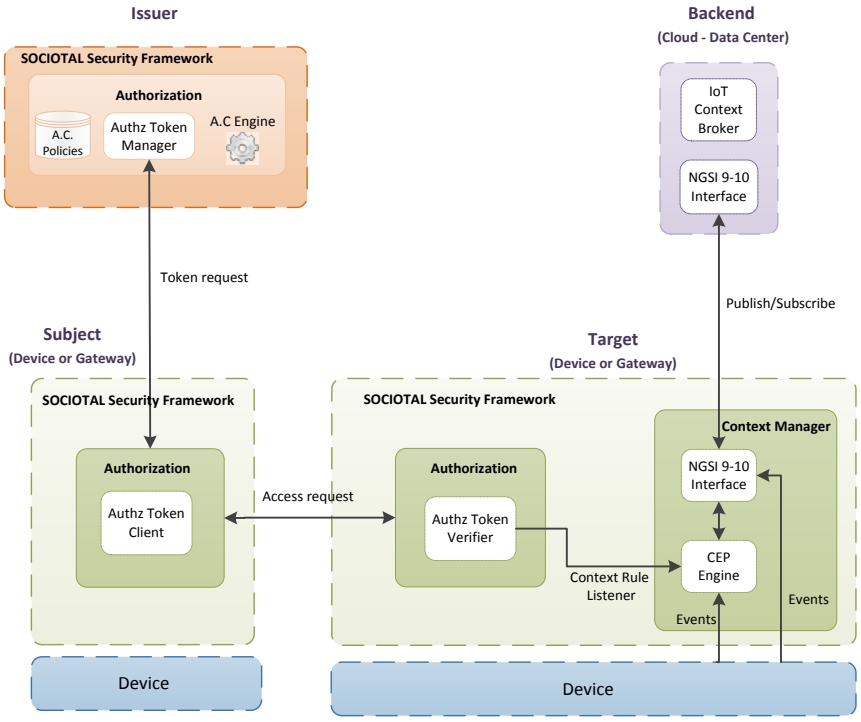

Figure 20. SocIoTal context-aware access control

Moreover, in this scenario another authorization process can be carried out, in this case between the target device B when trying to access to the Centralized Context Manager placed in the Backend. Thus, the first time the device B wants to subscribe to the Centralized Context Manager (to be notified later on about context information), the device B is required to present its capability token against the Authz component of the Backend. The token verifier sub-component of the Auth verifier checks the token prior allowing the device B to get subscribed to the Broker.

3.3.1 Sequence Diagram for non token-based approach

The generic sequence diagram for controlling the access to an IoT service using the context-based authorization system is shown in Figure 21. This process is based on the access control interactions from D2.1., in which capability tokens are not employed.

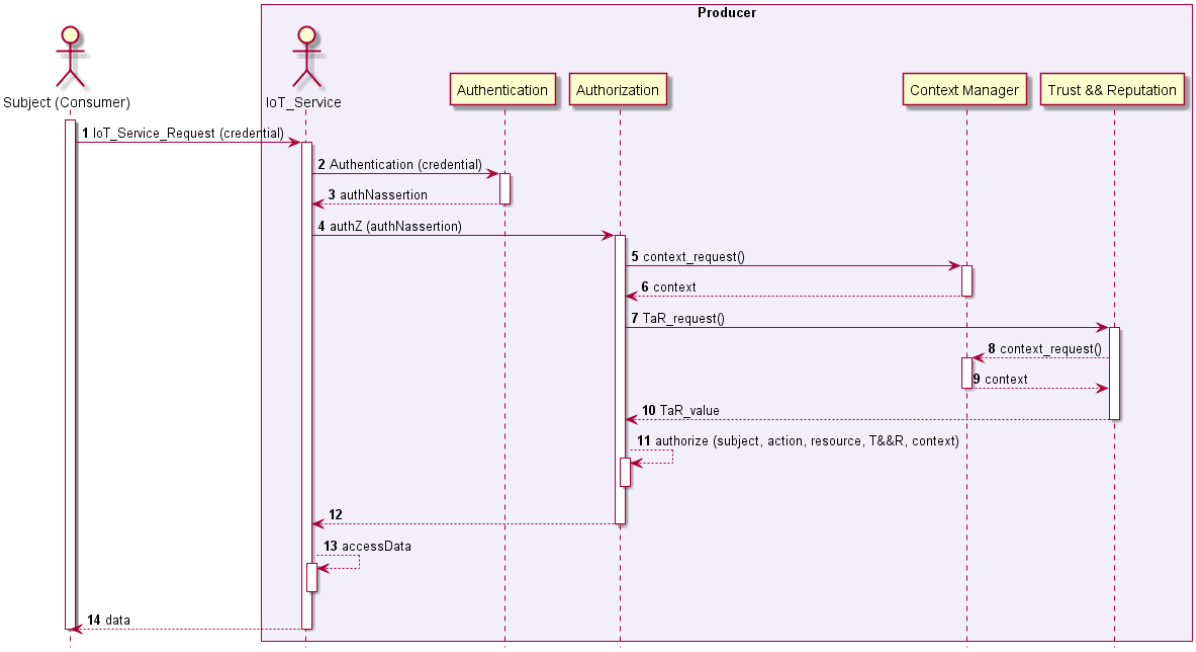

Figure 21. Non token-based access control sequence diagram

- In a first stage, a Subject, (user or smart object) acting as consumer, requests to get access to an IoT Service or virtual entity (producer).

- Then, the subject is redirected to the Authentication component of the IoT Services, where it is authenticated. According to the previous section, this process can be based on the use of anonymous credentials which can be employed by the subject to be anonymously authenticated in order to preserve its privacy.

- After a successful authentication process, the Authentication component generates an authentication assertion to be used during the transaction an optionally afterwards, depending on the validity period of it.

- In the same process, and making use of the assertion previously generated, an authorization process is carried out in order to make an authorization decision about that subject to access the IoT service.

- To make the authorization decision, the Authorization component firstly obtains the contextual information that is required in the authorization policies to make the decision. The type of information to be considered will depend on the IoT Service being accessed and it will provided by the Context Manager. Moreover, the Authorization component could optionally request the trust and reputation values related to the requester subject in order to avoid untrustworthy devices to get access to the IoT Service. Again, some contextual information can be needed to calculate trust scores for the subject.

– Then, the Authorization component evaluates the access control policies and makes an authorization decision that is sent back to the IoT Service and finally delivered to the subject.

3.3.2 Sequence Diagram for capability token-based approach

The following sequence diagram in Figure 22 shows the use of the SocIoTal access control system based on capability tokens which are obtained before the access to an IoT service.

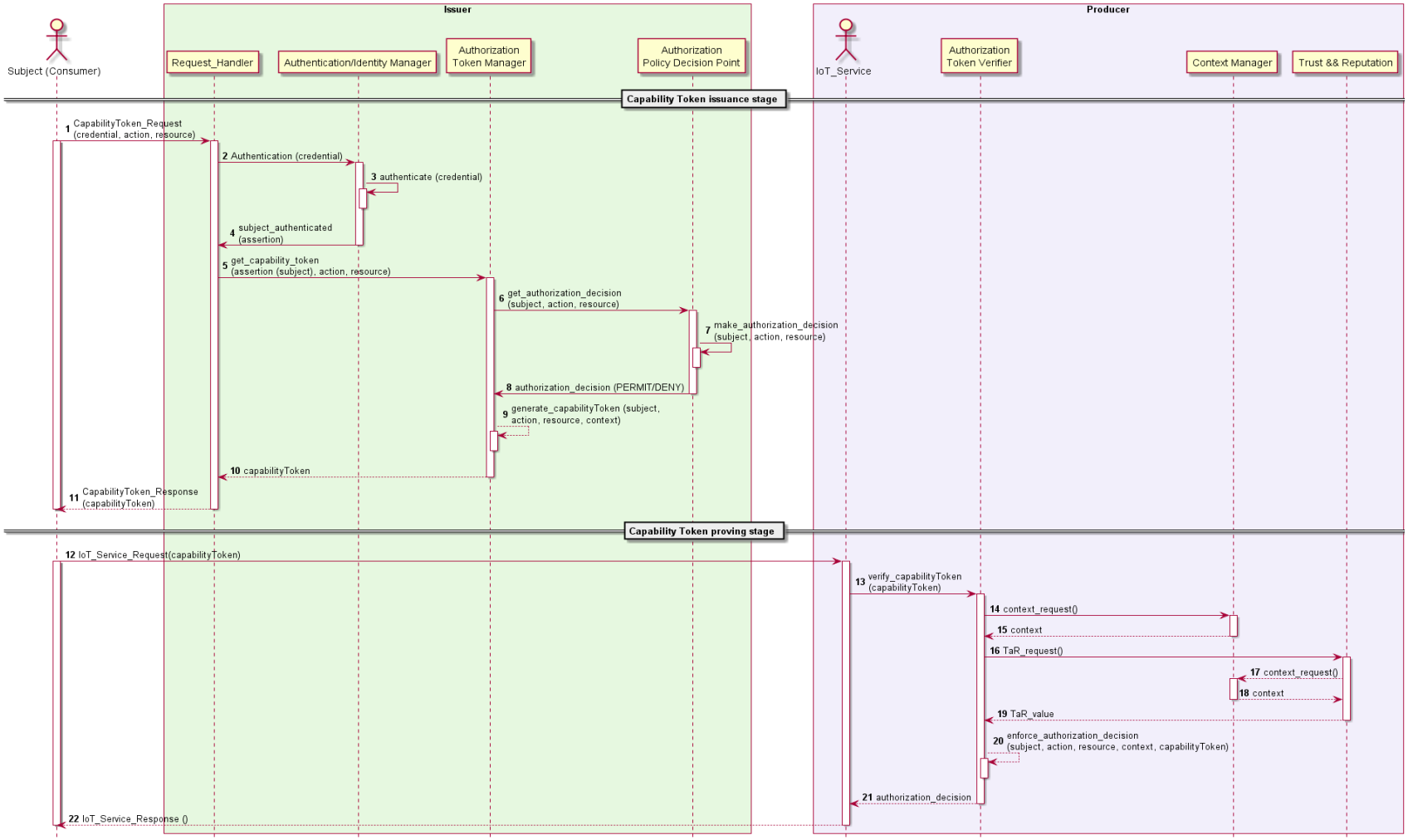

Figure 22. Capability token-based access control sequence diagram

– Firstly, during the "Capability Token issuance stage" a subject (user or smart object) acting as consumer, requests a capability token to an Issuer entity in order to get access to a service being provided by an IoT service or virtual entity (producer). This request contains the target resource and action, as well as a credential to identify the subject.

– The Issuer entity, through the Authentication Manager (or Identity Manager) authenticates the user according to the mechanism which were considered in D2.1 (e.g. based on X.509 certificates or anonymous credential systems).

– After a successful authentication process, the Authentication component generates an authentication assertion to be used during the authorization process. Therefore, the Request_Handler component generates a request in order to get a capability token to be delivered to the subject.

– Then, the Authorization Token Manager component generates a request to the Authorization Policy Decision Point to know if the requester subject is authorized to perform the request action over the target resource.

– The Authorization Policy Decision Point evaluates a set of access control policies in order to get an authorization decision for such request. This evaluation can be based on contextual conditions (e.g. time) which drive the decision.

– Once the authorization decision is made, the Authorization Policy Decision Point sends it to the Authorization Token Manager. Then, in case of an affirmative decision, this component generates a capability token in which the privilege is embedded. According to the DCapBAC model, this token may optionally contain a set of conditions to be locally verified by the IoT Service.

– The token, which is generated by the Authorization Token Manager is finally delivered to the Subject.

– Then, during the "Capability Token proving stage", the subject makes use of the capability to get access to an IoT Service.

– When the IoTService receives the request along the capability token, it contacts to the Authorization Token Verifier in order to validate the token. Firstly, this entity makes a request to get contextual information from the Context Manager (e.g. location of the subject). After that, it evaluates the capability token along the contextual information in order to grant or deny the requested action. Finally, the obtained response is delivered to the subject.

3.3.3 Interfaces definition

The main API functions to realize the envisioned access control mechanisms are:

- ***addAuthzPolicy*** *(authz_policy)*

  It adds an *authz_policy* to a specific policy set.

- ***getAuthzPolicy*** *(authz_policy_id) : AuthzPolicy*

  It returns the *authz_policy* corresponding to a given *authz_policy_id*.

- ***updateAuthzPolicy*** *(authz_policy)*

  It updates the authorization policy corresponding to a given *authz_policy_id*.

- ***deleteAuthzPolicy*** *(authz_policy_id)*

  It deletes the authorization policy corresponding to a given *authz_policy_id*.

- ***make_authorization_decision*** *(subject, action, resource,) : authorization_decision*

  This function aims to check the permission of a subject to perform an action over a resource. Such process is based on the evaluation of XACML policies.

- ***generate_capabilityToken*** *(subject, action, resource, conditions) : capabilityToken*
  In case of a "Permit" authorization decision, this function is used to generate a capability token. This contains the privilege associated to the subject, action and resource which were used in the previous function. In addition, a set of contextual conditions can be included in the token, specifiying certain access restrictions (e.g. regarding context or trust values) to be locally verified at the target device (producer)

- ***enforce_authorization_decision*** *(subject, action, resource, authz_token, context): Permit/Deny*
  This function is used to check the validity of an authorization token. Specifically, it should check at least:
  - The token is valid and have not expired
  - The requested action matches a specific privilege in the token
  - Conditions are fulfilled
  - Cryptographic verification (e.g. issuer's signature)

3.3.4 Access Control Policies

According to the previous description, the SocIoTal access control system will based on the usage of XACML policies for making authorization decisions. The definition of these policies will be based on the requirements of the use cases considered, as well as on the definition of groups of smart objects (i.e. communities and bubbles). Thus, Figure 23 shows an example of XACML policy to be deployed by the access control system of SocIoTal. Firstly, using the *Target* element, it is indicated that the policy will be only applied for access control requests which are intended to obtain the position of a user from the bubble B. The policy has a *Rule* requiring the subject belongs to the bubble A. Furthermore, in order to provide a more accurate and comprehensive access control mechanism, the *Obligations* element specifies that, in the case of a "Permit" decision, the PEP (in this case, the user U) must ensure that the trust value associated to the requester smart object is greater than 0.7. In addition, other obligations may be specified related to the context in order to be verified by the target device upon receiving an access request.

```
<Policy "SocIoTal_example">
  <Target>
      <Resource>
          <AttributeValue DataType="String">bubbleB/userU/position</AttributeValue>
          <ResourceAttributeDesignator DataType="String" AttributeId="resource-id" />
      </Resource>
```

```
        <Action>
            <AttributeValue DataType="String">get</AttributeValue>
            <ActionAttributeDesignator DataType="String" AttributeId=" action-id "
        </Action>
    </Target>
    <Rule “Permit">
        <Target>
            <Subject>
                <AttributeValue DataType="String">bubbleA</AttributeValue>
                <SubjectAttributeDesignator AttributeId="subject-id" DataType="String"/>
            </Subject>
        </Target>
    </Rule>
    <Obligations "trust" FulfillOn="Permit">
        <Apply “trust-greater-than-or-equal">0.7</Apply>
    </Obligations>
    <Rule "Deny">
</Policy>
```

Figure 23. XACML policy example for the SocIoTal access control system

Moreover, according to the sequence diagram explained in Section 3.2.2, in the case that the XACML authorization decision is "*Permit*", a capability token will be generated and then used to access the service (in this case, the position of a specific user from bubble B) by the subject entity. This authorization token contains the privilege obtained, and an associated condition to the trust value obligation by using the token representation which was specified in section 2.2.1. Additionally, an overview of the SocIoTal access control system application for a secure and effective management of communities and bubbles will be given in section 4.2.

3.4 Secure Group Sharing

The Group Manager component of the SocIoTal security framework is based on the use of the *Ciphertext Policy Attribute Based Encryption* (CP-ABE) [54] cryptographic scheme in order to enable a secure data sharing mechanism with groups of entities (i.e. communities and bubbles of smart objects. The following figure shows how contextual information from the Context Manager of a subject entity (acting as an information producer) is used by the Group Manager to select a specific CP-ABE policy to encrypt a specific information. Specifically, the Group Manager contains a set of *Sharing Policies* how the information is disseminated according to contextual data. These policies are evaluated by the *Group Sharing engine* before information is disseminated by the subject. The result of the evaluation of these policies is, in turn, a CP-ABE policy, which is employed by the *CP-ABE engine* to encrypt the information with that policy. After the information is encrypted and disseminated, the Group Manager of a target entity (acting as a consumer) will try to decrypt it with CP-ABE keys related to its identity attributes through its *CP-ABE engine*.

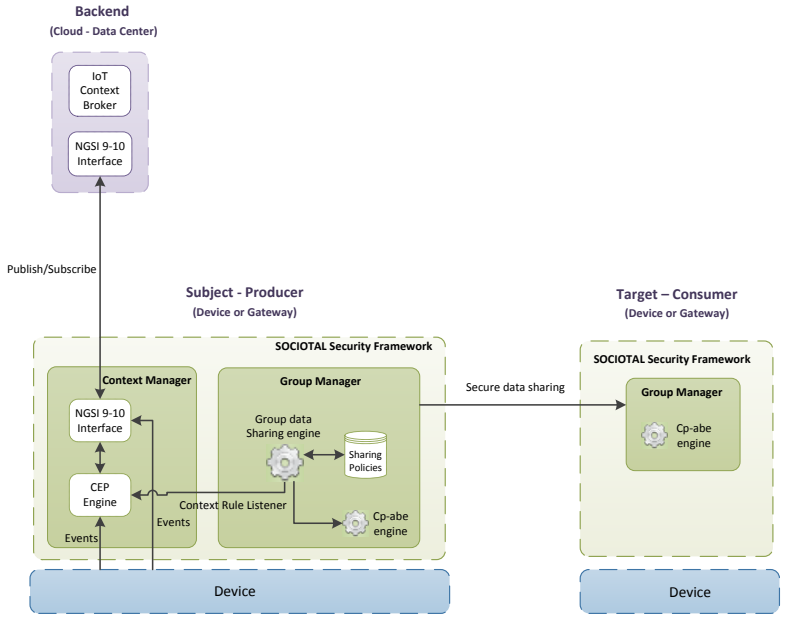

Figure 24. SocIoTal context-aware group sharing

Figure 24 shows an overview of the main components of the Security Framework required when the secure group sharing takes place between a producer and consumer. As can be seen, in the producer side the Group manager evaluates the sharing policies taking into account the Context handled by the local Context Manager, which in turn, can interact with the Context broker. As a result of the sharing policies evaluation, a CP-ABE policy is selected to encrypt the data, which is then delivered securely to the consumer(s).

3.4.1 Sequence Diagram

According to the previous section foundations and the use of CP-ABE scheme, Figure 25 shows the sequence diagram for the Secure Group Sharing mechanism. At high-level, three entities are considered for this scenario:

- The Subject role, acting as a *consumer*, which is intended to get access to a specific data.
- An IoT Service role, acting as a *producer*, which will be in charge of disseminating data in a secure way. Such entity consists of other components of the SocIotal security framework which will perform the task of determining a specific CP-ABE policy to encrypt data.
- Additionally, the Attribute Authority, which is responsible for generating CP-ABE keys for potential consumers. This entity will be usually deployed in a separate component

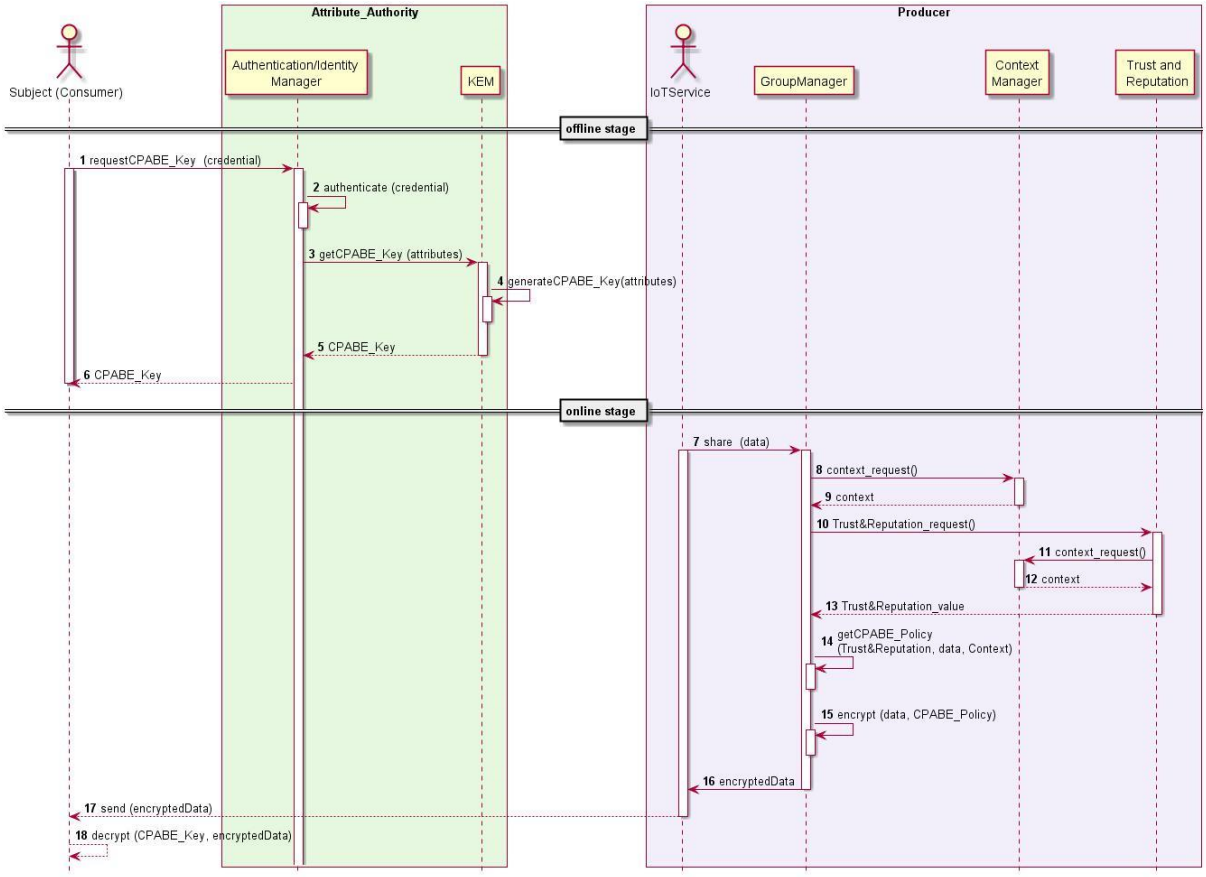

Figure 25. Secure Group Sharing sequence diagram

The description of the sequence diagram is as follows:

- Firstly, during an offline phase, the subject wishes to get a CP-ABE key associated with a set of attributes of their identity. These attributes can be specified in the form of an authentication token or credential issued by a trusted entity. For this purpose, the subject presents it credentials to the Authentication/Identity Manager depending on the authentication mechanism employed (e.g. based on X.509 certificates or Idemix). One the subject is successfully authenticated, the KEM component generates a CP-ABE key according to the identity attributes which were proved by the subject during the authentication stage.
- The data sharing is performed during an online phase. First, when the *IoTService* decides to disseminate (push) a certain data, it notifies the Group Manager component to perform this dissemination in a secure way for a set of potential consumers.
- The Group Manager is the component responsible for encrypting data. The decision on which CP -ABE policy to use for this data is determined by three factors:
  - The type of the data to be pushed (e.g. location, identity, etc.)
  - The current context of the IoTService which is obtained according to the previous section.
  - Optionally, the values of trust and reputation associated with these policies (actually representing groups of smart objects), which are obtained by requesting the *Trust && Reputation* component.
- When these values have been determined, a set of sharing policies which are in the *sharingPolicies* database of the GroupManager are evaluated. The result of this evaluation will be a CP-ABE policy which will be used to encrypt the data.

– Finally, a subcomponent *CPABEEngine* inside the Group Manager is responsible for encrypting the data with the policy obtained in the previous step.
– Once the data has been encrypted, it is returned to the IoTService actor which finally pushes the encrypted data. Then, in case the *CPABE_Key* obtained in the offline phase satisfies the *CPABE_Policy* used to encrypt, the subject can get access to data

3.4.2 Interfaces Definition (API)

The envisioned API functions for the use of a CP-ABE schema to realize group communication security are:

- **addSharingPolicy** (sharing_policy, cpabe_policy)
  It adds a *sharing_policy* to a policy set, specifiying the corresponding *cpabe_policy* to be used in case such *sharing_policy* is successfully evaluated.

- ***getSharingPolicy*** *(sharing_policy_id) : SharingPolicy*
  It returns the *SharingPolicy* corresponding to a given *sharing_policy_id*.

- ***updateSharingPolicy*** *(sharing_policy_id)*
  It updates the sharing policy corresponding to a given *sharing_policy_id*.

- ***deleteSharingPolicy*** *(sharing_policy_id)*
  It deletes the sharing policy corresponding to a given *sharing_policy_id*.

- ***getCPABEKey*** *(attributes) :* $CPABEKey_{attributes}$
  It returns a CP-ABE key associated to a specific set of *attributes*. A mechanism to prove the entity possesses such set of attributes is required (e.g. X.509 certificates or anonymous credential systems)

- ***getCPABEPolicy*** *(sharingPolicies, data, trust&reputation, context) : CPABEPolicy*
  Based on the set of predefined *sharingPolicies,* trust and reputation values and the current *context* in which sharing transaction is going to be done, it evaluates the sharing policies and returns a CP-ABE policy that will be used by *encryptData* function.

- **share** (data)
  It first obtains the context and the trust and reputation values. Then it calls the *getCPABEPolicy* method to obtain a CPABEPolicy. Afterwards, it calls the *encryptData* operation and finally *push* (or publish in case of a publish/subscribe scenario) the encrypted data.

- ***encryptData*** *(data, CPABEPolicy) : ciphertext*
  It takes the *data* to be encrypted, and a *CPABEPolicy* representing subsets of attributes which are allowed to decrypt *data*. It returns *ciphertext* containing *CPABEPolicy*.

- ***decryptData*** *(ciphertext,* $CPABEKey_{attributes}$*): data*
  It takes the *ciphertext* to be decrypted, and a $CPABEKey_{attributes}$ representing a secret key with an associated set of attributes. It returns *data* in the case $CPABEKey_{attributes}$ satisfies the *CPABEPolicy* which is contained in *ciphertext*.

3.4.3 Sharing Policies

According to the previous description, the Group Manager component has a set of group sharing policies which are evaluated before information is disseminated and intended to be shared with a community or bubble of smart objects. These policies specify particular contextual conditions which are needed to satisfy a certain sharing policy. The result of these policies is, in turn, a CP-ABE policy indicating the set of entities which will be enabled to decrypt the information to be shared. As for the Identity Selector, the group sharing policies can be defined by making use of a rule-based tool that allows describing the context information. This subcomponent can be also used by high level graphical applications that can facilitate users the definition of rules to be integrated into the set of sharing policies. The relationship between sharing policies and CP-ABE policies is shown Figure 26. Both sets of policies are within the Group Manager component of the device. According to it, when a sharing policy is successfully evaluated, the resulting CP-ABE policy is used to encrypt the information to be shared, and consequently sent to the CP-ABE engine subcomponent. In the case of two or more sharing policies are successfully evaluated, the most restrictive CP-ABE policy could be selected.

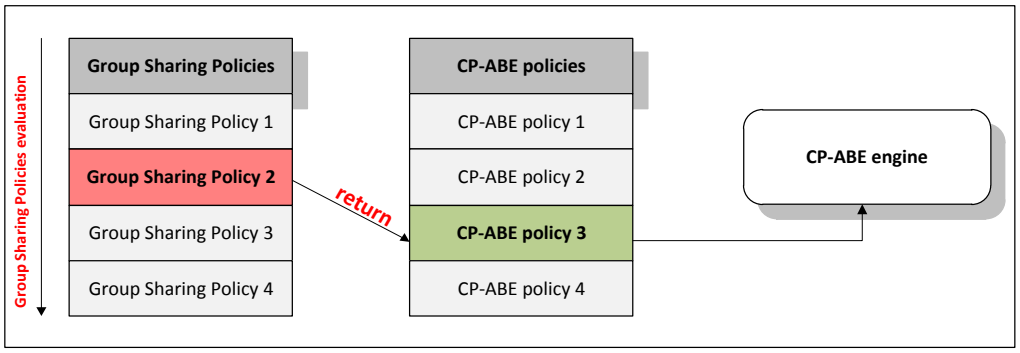

Figure 26. Relationship between sharing policies and CP-ABE policies

The following piece of code shows an example of a high level rule definition that defines a specific sharing policy considering the definition of bubbles and the use of CP-ABE policies. Specifically, the antecedent of the rule indicates a set of contextual condition (e.g. location and time), while the consequent specifies the CP-ABE policy which will be used under those contextual conditions. In this case, the CP-ABE policy is based on the attributes “bubbleA” and “friend”.

```
IF
   location=x AND time=y

THEN
   CP-ABE policy=”bubbleA AND friend”
```

Section 4 - Managing Access Control and Privacy in SocIoTal communities and bubbles

SocIoTal envisions the need of a security management among groups (communities and bubbles) of smart objects wishing to share information with each other. Based on the main security approaches which were presented in D2.1, and the access control system presented in section 2, below the application of these mechanisms for a secure and privacy-preserving management of Sociotal communities and bubbles is provided.

This section focuses on the authorization mechanisms to handle communities and bubbles. For a general overview about the management of communities and bubbles, please refer to Deliverable D2.1.

4.1 Access Control in communities and bubbles

Under SocIoTal foundations, each person or smart object can belong to different trust bubbles. Each of these bubbles represents a group of people and/or smart objects that communicate under the same security policies. In addition, different types of bubbles can be defined according to the relations between people and/or smart objects. For example, a personal bubble can be composed of smart objects belonging to the same owner. According to section 2, the access control mechanism used in SocIoTal is a combination of different techniques in order to accommodate the requirements of security and privacy into IoT scenarios. Figure 27 shows a high level scenario of the application of the SocIoTal access control system for a secure and effective management of communities and bubbles. Such process is considered to be deployed on IoT scenarios where smart objects (e.g. smartphones, sensors, actuators, etc.) can maintain relationships composing different kinds of bubbles (e.g. personal or family bubble).

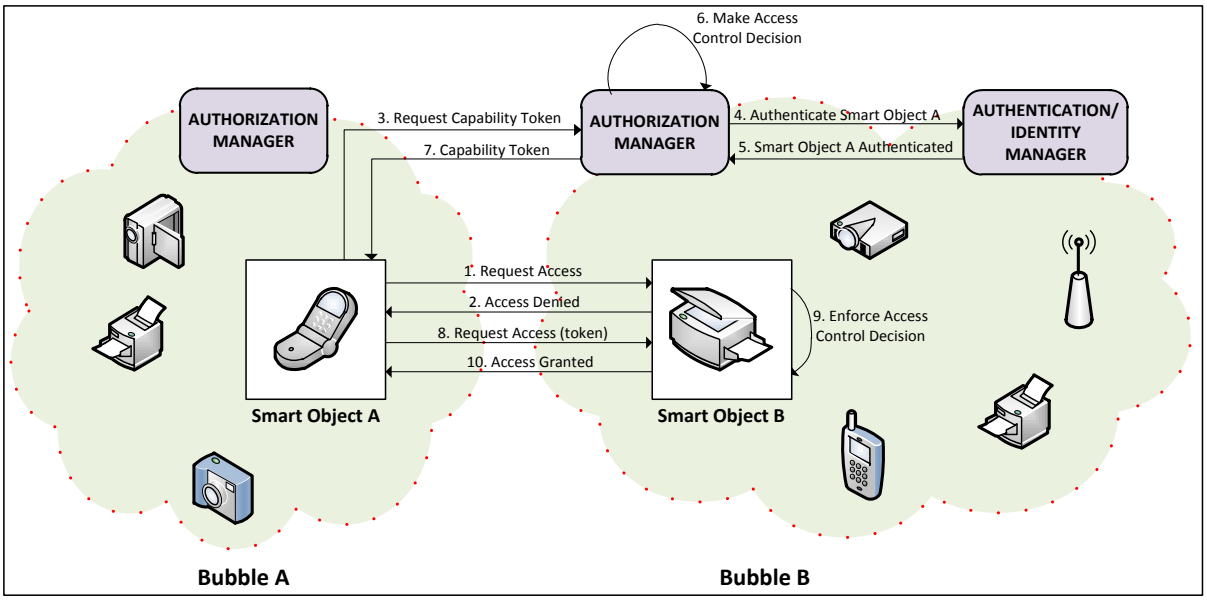

Figure 27. Access Control for SocIoTal communities and bubbles

According to Figure 27, each bubble is made up by a set of smart objects, along with an *Authorization Manager* (e.g. it can be deployed on a user smartphone in the case of a personal bubble), which is responsible for generating authorization credentials for smart objects. Furthermore, each bubble has an *Authentication/Identity Manager*, which is in charge of assessing the legitimacy of a requester smart object. This scenario assumes a

smart object A tries to get access a service that is provided by other smart object B in a different. At this point, it should be pointed out that the components of a community or bubble are equipped with capability tokens, which are employed for a secure communication with the rest of components of the bubble. For an *inter-bubble* communication, as in the case of Figure 27, during an *offline* stage, the smart object A (from the bubble A) contacts with the Authorization Manager (from the bubble B) in order to get an authorization credential (i.e. a capability token) to get access to smart objects from bubble B. Before the authorization process is performed, the Authentication/Identity Manager verifies that the requester smart object is who it claims to be. This process can be based on traditional authentication mechanisms (eg, based on login / password or X.509 certificates). Optionally, anonymous credential systems (e.g. Idemix) could be used in order to preserve the privacy of the smart object. It should be noticed that the set of devices belonging to the same bubble, trust the Authentication/Identity and Authorization Manager, and they are able to establish security context with them. Once the smart object A is successfully authenticated, the Authorization Manager evaluates the XACML policies using the policy engine and makes an authorization decision. In case of a PERMIT decision, the PDP generates a capability token with the set of privileges associated to the smart object over bubble B. An example of capability token for this scenario is shown in Figure 28.

```
{
  "id": "8Lwc4k_0naQkPbA9",
  "ii": 1415174456,
  "is": "authorizationManager@bubbleB.org",
  "su": "zNwS5FetB4rwzSKsWwSBAxm5wDa=JgLjHU8zSnmeSFQgSG9HhdsJrE8=",
  "de": "bubbleB",
  "si": "SbUudG4zuXswFBxDeHB87N6t9hR=PBQqCN3gpu7nSkuPzDk7kaR3dq1=",
  "ar": [
     {
        "ac": "GET",
        "re": "temperature"
     }
     {
        "ac": "POST",
        "re": "light"
     }
  ],
  "nb": 1415174456,
  "na": 1415175569
}
```

Figure 28. Capability token example for SocIoTal communities and bubbles

As can be seen in the token example above, the "device" field makes reference to the set of smart objects composing the bubble B. After the smart object A has received the token, it makes us of it to get access to a resource being hosted on the smart object B. When the smart object B receives the access request, it checks whether the token is valid or not checking the signature and if the token has expired. Then, as the access rights are contained in the token, the device B checks them against the requested action, including the conditions described in the token (if any). If these conditions are satisfied, the request is accepted and the service is provided to the smart object A.

4.2 Secure Data Sharing in communities and bubbles

The realization of IoT scenarios with entities composing dynamic communities requires the definition of appropriate mechanisms in order to design a scalable and distributed security solution for the envisioned use cases. Unlike current Internet, in such dynamic coalitions, IoT

interaction patterns are often based on short and volatile associations between entities without a previously established trust link. Providing basic security properties to such data exchange is a paramount security issue that also needs to be properly addressed by allowing more flexible sharing models (beyond the classic request/response approach), as well as fleeting and dynamic associations between entities, while privacy of involved entities is still preserved.

Figure 29 shows the scenario in which a specific smart object disseminates information to make it visible only to a specific set of entities. This process is based on the CP-ABE cryptographic scheme, which is used to allow secure communication between objects belonging to the same bubble. In this case, a smart object A (from the bubble A) tries to get access to data being shared in the bubble B. It is assumed that smart objects in a bubble X maintain at least one CP-ABE key associated with the attribute "bubbleX" that allows them to exchange information in a secure way. Thus, the smart object A needs to obtain a CP-ABE key associated with the same attribute in order to get access to data being shared among objects of bubble B.

According to Figure 29, each bubble has a Group Manager, as an entity responsible for generating CP-ABE keys to allow secure sharing transactions. Therefore, during an *offline* stage, the smart object A contacts with the Group Manager from bubble B in order to get a CP-ABE key to get access to the information being disseminated among smart objects from bubble B. Likewise for the access control process, before the key generation process is carried out, the Group Manager verifies that the requester smart object is who it claims to be. This process can be based on traditional authentication mechanisms (eg, based on login/password or X.509 certificates). Optionally, anonymous credential systems (e.g. Idemix) could be used in order to preserve the privacy of the smart object. Once the smart object A is successfully authenticated, the Group Manager generates and delivers a CP-ABE key which is associated to the attribute “bubbleB”. In addition, this key could be associated to other identity attributes which were proved during the authentication process (e.g. attributes in an Idemix proof), enabling the composition of *sub-bubbles* according to different combinations of identity attribute values. After the smart object A has received the corresponding cryptographic key, during an *online* stage, it can makes use of it in order to decrypt the information which is disseminated by smart objects in bubble B.

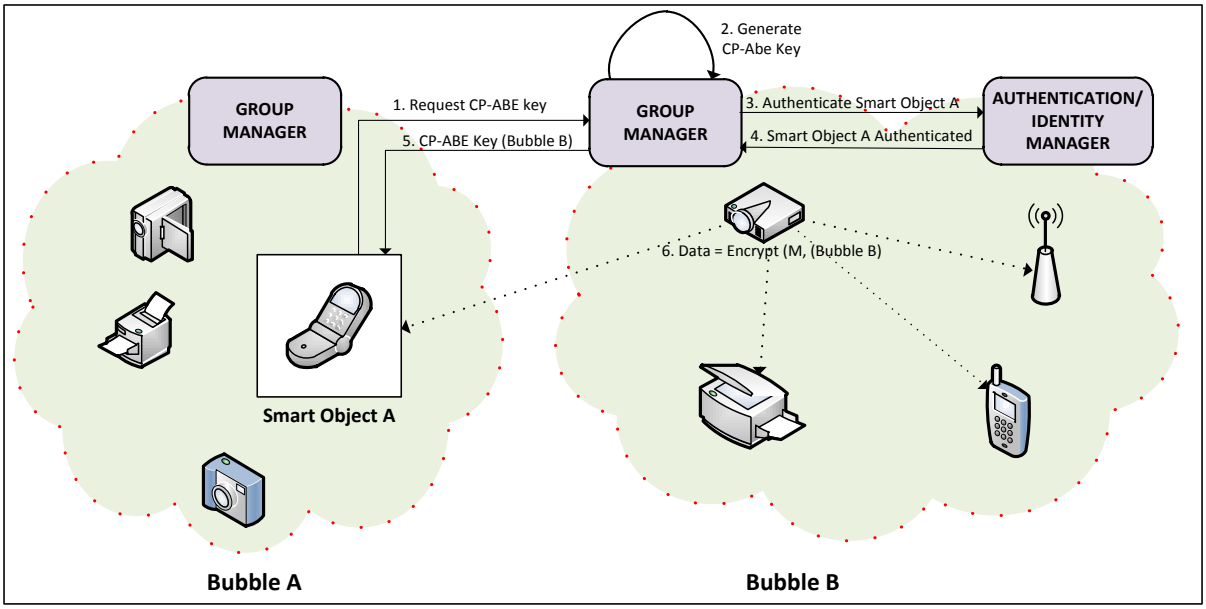

Figure 29. Secure Data Sharing for SocIoTal communities and bubbles

In addition to bubbles or communities which can be statically defined, given the pervasive and dynamic nature of the scenarios envisioned by SocIoTal, it is necessary to consider the application of security mechanisms to cope with the requirements of the so-called *opportunistic bubbles*. An opportunistic bubble is a kind of dynamic sharing group that is not registered as community statically anywhere. Unlike the previous approach, this kind of bubbles leverages opportunistic contact and ad hoc connection among devices, simulating the way people communicate in the physical world. Opportunistic bubbles are formed spontaneously, particularly based on physical proximity, and using short-range communications technologies without infrastructure. Due to the inherently mobile nature of smart objects (such as mobile phones), this model has an important interest to be exploited in IoT. For example, in a real life scenario, a user can create an opportunistic network of mobile phones when he goes into a restaurant to share information with other people who satisfy a specific combination of identity attributes.

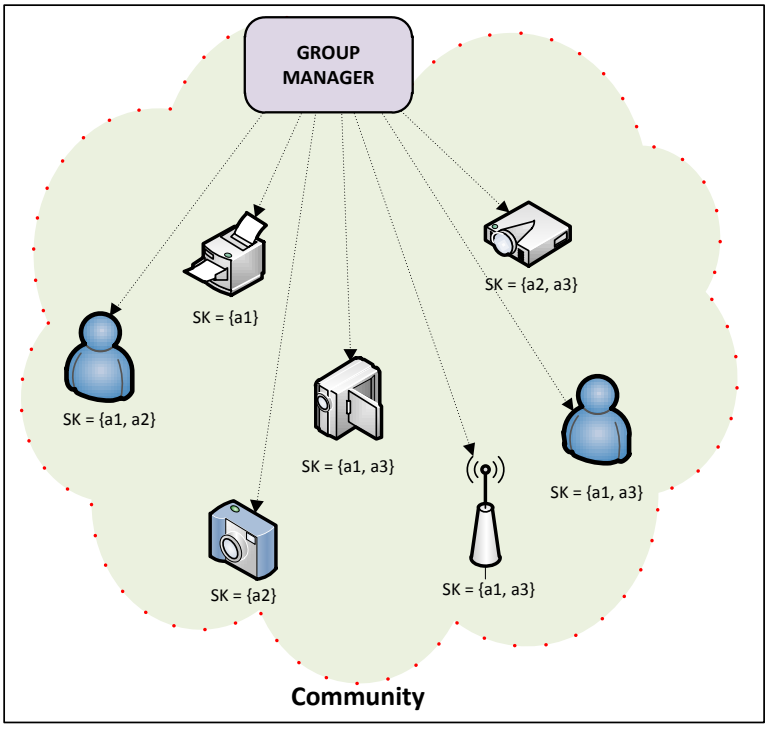

Figure 30. CP-ABE keys provisioning in Communities

Likewise the previous approach, before the realization of a secure data sharing between members of an opportunistic bubble, it is necessary that they previously acquire CP-ABE keys. This process, which has been previously explained, involves a Group Manager responsible for generating keys associated with specific sets of identity attribute values. According to Figure 30, this phase can be carried out in an offline stage in a community of people and/or smart objects. Once these entities within the community have received the CP-ABE keys they can create opportunistic bubbles, as it is shown in Figure 31.

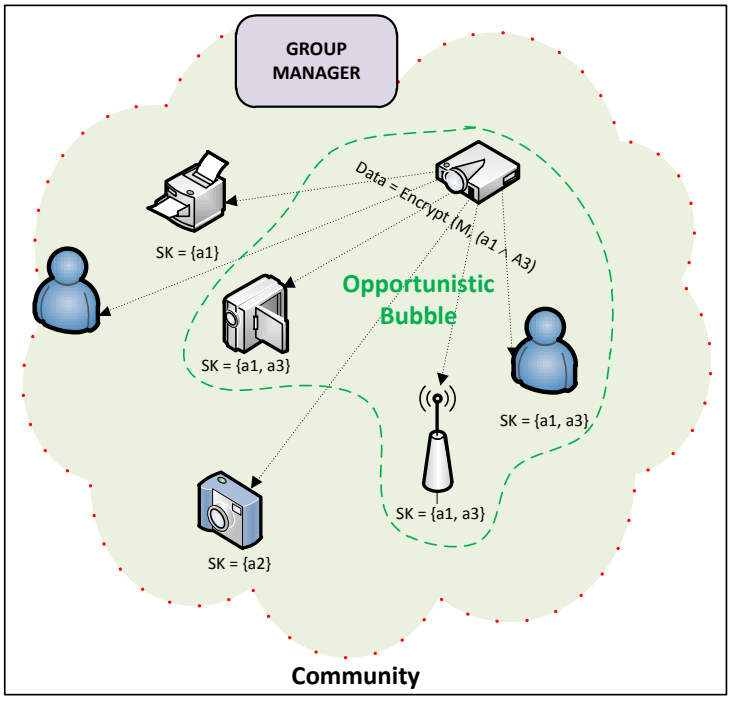

Figure 31. Secure Data Sharing for opportunistic bubbles

This creation of opportunistic groups of entities is given by the fact that an entity can encrypt some data using a combination of attributes to make it visible only to a set of entities whose keys satisfy this combination. Unlike traditional cryptographic schemes (e.g. based on symmetric group keys), in this approach there is no need to generate new keys to enable a secure sharing between subgroups of entities. Indeed, information can be encrypted under different CP-ABE policies and decrypted by the same CP-ABE key. In this way, during the sharing process, any third party is involved, enabling a secure and ad hoc communication between smart objects.

As an example, Figure 32 shows a scenario in which the Secure Group Sharing mechanism is applied. Specifically, the scenario is based on the publish/subscribe sharing model, and the use of CP-ABE scheme. As shown in Figure 32, four different entities are required: a set of data producers (which publish information), a set of data consumers (showing interest on specific kinds of information), the SocIoTal broker (storing data from producers and delivering to corresponding consumers), and an Attribute Authority (in charge of generating CP-ABE keys according to a set of identity attributes).

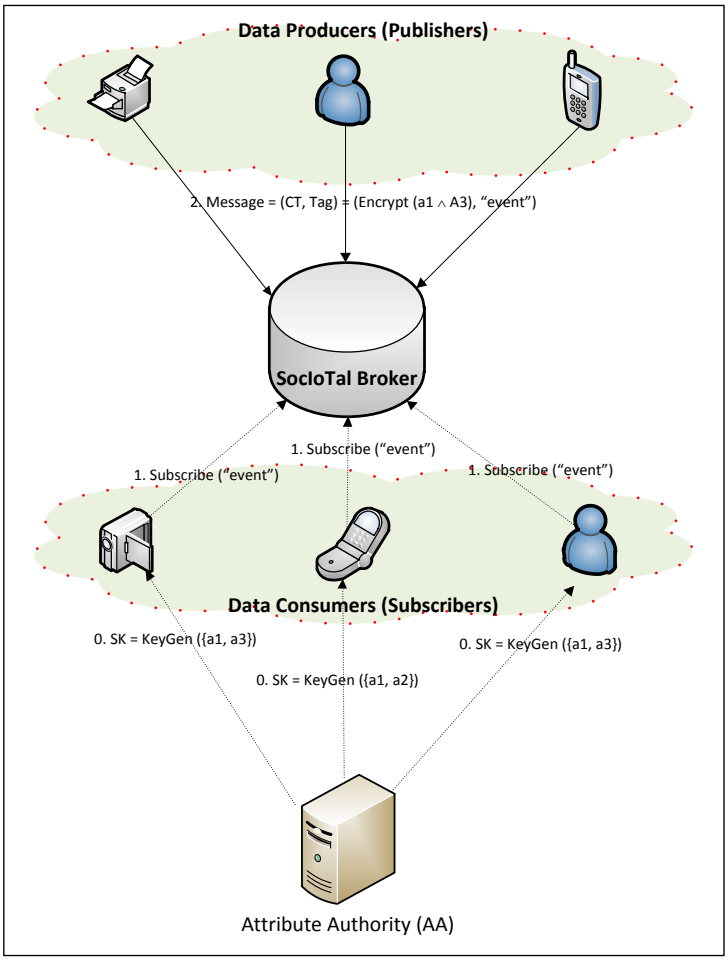

Figure 32. SocIoTal secure group sharing

In this scenarios, users are able to share information each other through a SocIoTal broker accessible by the consumers. During the offline stage, once the cryptographic keys are delivered by an Attribute Authority (AA) to data consumers, they show interest in a specific event and, consequently, they subscribe themselves on the SocIoTal broker. This entity could be deployed in a separate component, such REST interface or in the cloud. Then during the online stage, after data is encrypted with a specific CP-ABE policy (that is derived from sharing policies evaluation), the resulting ciphertext is attached with a specific tag (e.g. "event"), and sent to the SocIoTal broker. Consequently, this entity can match the received tag with the interest specified previously by data consumers (i.e. subscribers). However, while this information is disseminated to the set of subscribers that showed interest on "event", only those entities (i.e. a group) with CP-ABE keys satisfying the CP-ABE policy used to encrypt data, will be able to decrypt the information.

Section 5 - SocIoTal Security Framework in FI-WARE platform

The SocIoTal security framework addresses most of the security topics and challenges emerging from the IoT paradigm. However, covering in deep each security and privacy aspect can be an overwhelming task to accomplish, and therefore, it is worth relying on existing solutions that already handled some of these issues successfully.

Many of the security and privacy aspects have been already addressed and implemented in other platforms like Fi-ware and Butler. SocIoTal envisages taking advantage of these solutions and implementations reusing and extending partially some of them to have better chances to succeed, avoiding implementing solutions from scratch, which is highly time consuming and out of the scope of the project. This approach allows the SocIoTal security concerns to focus on innovation about privacy-preserving aspects for IoT, instead of waste time on traditional security solutions (also required by the framework). Moreover, the SocIoTal dissemination and liaison with other project are also benefited of this integration with other platforms.

In this context, this section provides a first insight to enforce part of the SocIoTal security solutions over a well-known and tested, existing security platform like Fi-ware. Namely, it extends both of its main security enablers, the Access Control Generic Enabler and the Identity Management Generic enabler.

5.1 Extending the Fi-ware Access Control with SocIoTal AC

5.1.1 Fi-ware Access Control generic enabler overview

In order for the reader to have a better understanding of the integration of Fi-ware and SocIoTal access control systems, this subsection gives an overview of the Fi-ware IdM Access control enabler (AC GE) [55]. It should be noticed that the content of this subsection has been taken from the specification of the enabler in the Fi-ware wiki [56].

The Fi-ware Access Control generic enabler provides a mean for controlling access to resources that are available on a given FI-WARE Instance. Those resources are owned by users of the FI-WARE Instance, either end-users (data that belongs to each end-user) or by application or service providers (API operations exported by a given application or service, which may well be a FI-WARE GE). Such access will be typically requested by client applications or services (including FI-WARE GEs) acting on behalf of another user. It would require approval from the resource owner and may be restricted by security policies that are either global to the FI-WARE Instance, or defined for application/services or for the end-users (both resource owners and end-user on behalf of whom access is requested), as well as the organizations that end-users belong to, if any.

The Fi-ware basic concepts come from the OAuth 2 [8] and XACML 2.0 [57] standards illustrated by the sample scenario below. The OAuth standard is supported by the Identity Mangement GE essentially, the OAuth Authorization endpoint and Token endpoint in particular. The Access Control GE is only a consumer of OAuth tokens, therefore it only supports validation of OAuth tokens and getting authorization info from the token, such as user attributes. The XACML standard is only supported by the Access Control GE.

From the user and the client application's perspective, the main interactions are the same as described in the OAuth 2.0 standard, section 1.2. From the Resource Server's perspective, the OAuth token validation and API request authorization may be delegated to the PEP, so that the integration effort on the Resource Server is minimal.

The main Fi-ware Access control interactions are illustrated by Figure 33.

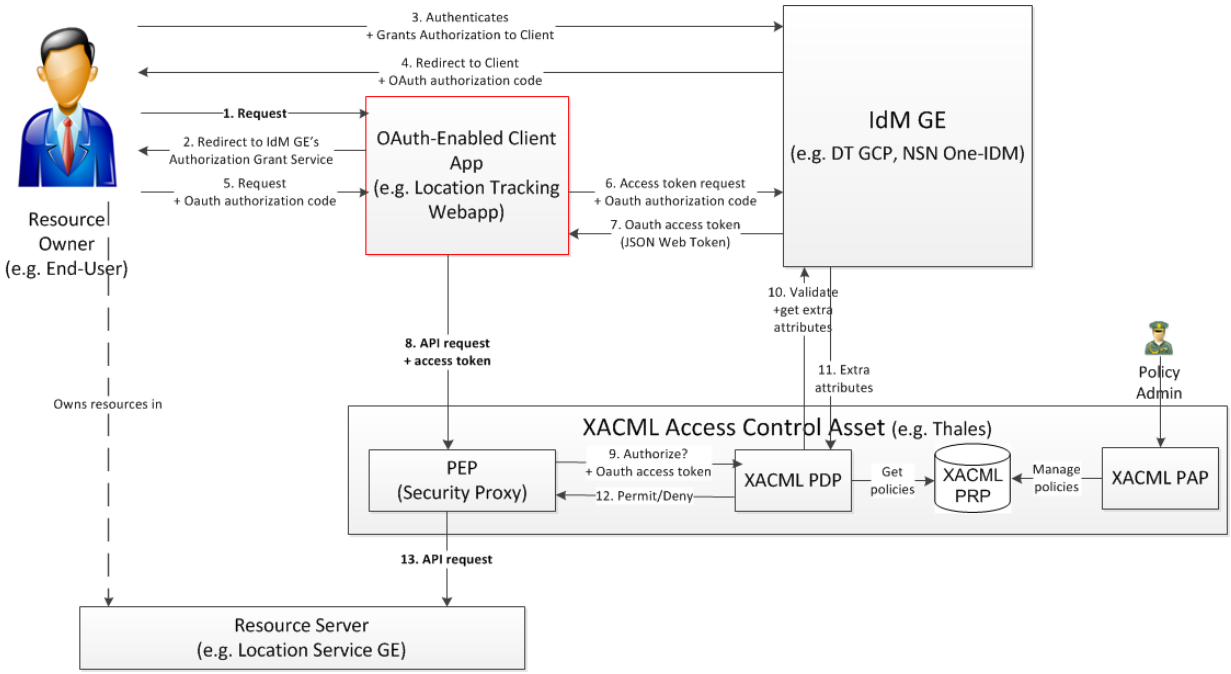

Figure 33. Fi-Ware Access Control Main Interactions [55]

Fi-ware Access Control design principles:

- The architecture complies with the OAuth and XACML standard.
- All APIs (IdM OAuth APIs, XACML Policy Management API, XACML PDP API, Resource Server APIs) are RESTful APIs.
- Format of access control policies managed by the PAP and enforced by the PDP is defined by the XACML standard.
- The PDP computes access decisions based on policies according to the XACML standard rules for policy evaluation (XACML engine).
- The PDP and PAP should support multi-tenancy.
- There are several options for the PEP deployment:
    1. Use a HTTP Proxy. In this case, the owner of the Resource Server must provide the necessary information to the PEP proxy owner for integration with the Resource Server.
    2. Built/Embedded in the ResourCe Server. In this case, the PEP implementation is the responsibility of the owner of the Resource Server.
- In all cases, the PEP would have to integrate with the IdM Generic Enabler for OAuth token validation, and the XACML PDP for requesting authorization decisions.

5.1.2 Fi-ware Access control enabler Extension with SocIoTal AC system

This section provides the main ideas and guidelines to extend Fi-Ware Access control enabler to be able to use the SocIoTal capability-based access control system explained in section 2.2.

Extending the Fi-ware Access control enabler will allow SocIoTal users and devices, which make use of the privacy-preserving partial identities, to access to Fi-ware protected resources by means of the their SocIoTal capability tokens.

Thus, SocIoTal users and devices could, for instance, be given access to the publish/subscribe Fi-ware context broker, which holds context environment measurements coming from sensors and actuators. To this aim, after a successful authentication using the SocIoTal IdM anonymous credential system (or another traditional authentication procedure), the Fi-ware extended Access Control System could endow users with Sociotal capability tokens, in addition to the traditional OAuth token provided by Fi-ware AC GE.

This extension would also allow that capability tokens obtained from another Sociotal capability manager could be understandable by the Fi-ware Access Control enabler, providing a compatible environment between both solutions.

Additionally, this extension will improve the Fi-Ware Access Control Enabler to cover scenarios it couldn't reach otherwise. Fi-Ware users could take advantage of the SocIoTal capability tokens obtained in order to access afterwards directly to target IoT devices, which are able to validate the access control tokens. This will speed up the process avoiding going thought a centralized Policy Decision Point every time users want to interact with the target devices.

This proposal will require two main extensions in the Fi-ware AC GE. On one hand, it would require extend the Fi-Ware Policy Enforcement Point (PEP) to be able to understand the SocIoTal capability tokens. On the other hand, it would require extending the Fi-ware XACML Access Control asset endowing it with the SocIoTal access token manager in charge of generating the tokens.

It should be noticed, that a detailed software integration design has not been detailed yet, since it has still not been decided whether it will be implemented in the scope of the project or not. In case the extension is finally implemented, future project deliverables (like D2.3) would describe the implementation and extension details.

5.2 Extending the Fi-ware Identity Management with SocIoTal IdM

The Fi-ware IdM generic enabler [58] has been designed to deal with scenarios where users have their identity information centralized in their home attribute provider or identity provider. In this user-centric approach, the users are requested to perform consent previous releasing their attribute manager to a client (usually a web application). On the other hand, Sociotal focuses on IoT scenarios where there cannot be a centralized attribute provider and users hold with them their identity attributes (in their partial identity credential) which are released to the target accessed services according to user privacy preferences.

SocIoTal IdM system unlike Fi-ware IdM enabler is based on the usage of anonymous credentials to guarantee privacy preserving, while ensuring, at the same time, minimal disclosure of personal data information when accessing to resources. In Fi-ware the authentication process is based on OAuth v2 tokens used later on to access to resources. The authentication process performed at the Fi-Ware IdM can be extended and enhanced to perform the authentication based on partial identities and using the SocIoTal privacy-preserving IdM mechanism.

5.2.1 Fi-ware IdM generic enabler overview

In order for the reader to have a better understanding of the Fi-ware and SocIoTal possible integration, this section gives an overview of the Fi-ware IdM General enabler (IdM GE) [58]. It should be noticed that the content of this subsection has been taken almost verbatim from the specification of the enabler in the Fi-ware wiki.

The IdM GE is mainly used for authorising foreign services to access user personal data stored in a secure environment. Hereby usually the owner of the data must give consent to access the data; the consent-giving procedure also implies certain user authentication. The Fi-ware IdM GE is used in multiple scenarios spanning from Operator oriented scenarios towards Internet Service Providers (ISP). End users benefit from having simplified and easy access to services (User Centric Identity Management).

Fi-ware IdM GE specifies the roles and responsibilities of the actors required to achieve integration between the IdM-System on the one hand side and services, as well as product management and product presentation on the other hand.

Generically, three responsibilities can be identified in the Fi-ware IdM GE [58]:

- The **IdM-System** is responsible for identity management processes such as login, logout, registration and the like and for customer management processes such as customer data management.

- A **service provider** is responsible for any process that has to do with service provisioning to an user. This responsibility encompasses things like evaluating the users entitlements based on authorisation data as provided by the IdM-System, serving the services which the user is entitled to use, and providing UIs and service-related information and processes to the user. The service provider, sharing a certain service level agreement with the IdM provider, is realizing the service sold to the customer of the partner.

- A **product provider** is responsible for any process to do with defining and offering sellable products to the partners' customer. Typically, the product provider runs a storefront such as a web-store or a landing page presenting the available (subscription) products to the potential customers. The product provider will typically be the IdM-System partner, and is also responsible for all customer communication as well as for all legal aspects of the customer relation.

The general structure of a FI-WARE IdM GE is sketched in the following figure:

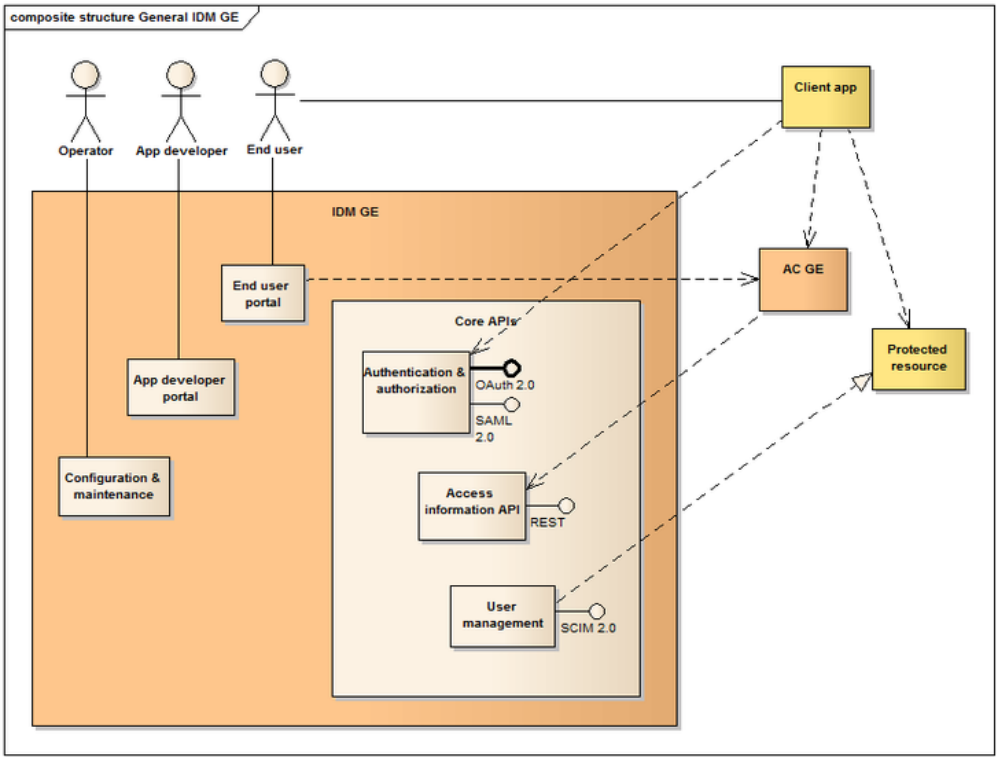

Figure 34. Fi-Ware IdM overall Structure [X]

An IdM GE offers Configuration and maintenance functionalities for FI-WARE Operators. This may be offered as a GUI, configuration files, and standards based API (e.g. JMX [X]) or any combination of these. Third party Application developers may register their applications with the IdM GE using these functionalities. This registration process must be implemented through a Web user interface called Application developer portal. The registration should include operator review and approval. End users register to FI-WARE using the End user portal. This is also implemented as a Web user interface. End users may also review and modify their personal data and maintain their privacy settings using this portal. Access to the end user portal should be controlled by the FI-WARE Access Control GE (AC-GE). Other FI-WARE components may access the IdM GE through Core APIs.

Each IDM GE implementation must contain an authentication and authorization API. This component must support OAuth 2.0 [8] for authorization. At least the authorization code grant (4.1) and resource owner password credentials grant (4.3) flows must be supported. Optionally implicit grant (4.2) may also be supported. IDM GE implementations should also allow configuring trusted FI-WARE components to access protected resources using (4.4) client credentials grant. For access token formats at least bearer tokens must be supported. If OAuth 2.0 is used for authentication, then OpenID Connect [59] should be used. It defines a simple identity layer on top of OAuth 2.0. Optionally other Web authentication standards (e.g. SAML [7] or OpenID [6]) may also be supported. Access control decisions are made by access control policies residing in the AC GE. Policies may require additional information from the IDM GE to make the decision, including, but not limited to token parameters and status, user groups/roles, client settings, user privacy preferences. These may be retrieved from the IDM GE using the Access information API. Optionally IDM GE may provide an API for user and role management. If such an API is provided then it should be compliant with SCIM 2.0 [60]. Only mandatory SCIM operations and features must be supported (there are very few optional features).

5.2.2 Fi-Ware IdM extension with SocIoTal IdM

The Fi-ware Identity Management system [58] has been designed to deliver a multi-tenant user and profile management solution that allows Enterprises to manage consumers of their (Web based) services in the Cloud securely. In Fi-ware, the Identity Manager is the central component that provides a bridge between IdM systems at connectivity-level and application-level. It encompasses a large number of aspects involved with users' access to networks, services and applications, including secure and private authentication from users to devices, networks and services, Authorisation & Trust management, User Profile management, Single Sign-On (SSO) to service domains and Identity Federation towards applications.

However, the IoT paradigm has usually also to deal with scenarios where users and devices access directly to target devices without having a central user repository that manages user profiles and provides user attributes to client web applications. For this reason, the SocIoTal IdM, unlike the Fi-ware IdM, address a small set of functionalities, focusing on the authentication process and the privacy preserving mechanism that enable users to use different partial identities to access target devices according to the context. Other IdM functionalities used in traditional web contexts, such as user profile management and SSO, are left in SocIoTal to open existing solutions like Fi-ware, which already provide successfully such means. Thus, the SocIoTal privacy preserving IDM, could be used along with the Fi-ware IdM to complement each other, when the scenarios require a central IdM where users can manage their profiles and give consent to web application clients, but at the same time it is wanted to allow users to make use of their partial identities to prove their identity attributes.

In Fi-ware the authentication process is based on OAuth v2 tokens where the PEP Proxy expects all the requests to have a header `x-auth-token` containing a valid access token from the IDM. The authentication process performed at the Fi-Ware IdM could be extended and enhanced to perform the authentication using the SocIoTal privacy-preserving IdM mechanism which is based on anonymous credential system.

Users and IoT devices can employ their anonymous credentials to obtain the OAuth tokens required by Fi-ware PEP to access to resources, ensuring minimal disclosure of personal data. In this regard the Fi-ware IdM enabler could be enhanced to generate the OAuth token required later on by the Fi-ware PEP, but in this case, the tokens are generated as the result of the authentication carried out by the SocIoTal IdM system, performing the identity verification process based on anonymous credential system.

To this aim, the integrated IdM should have installed the SocIoTal IdM Verifier module in charge of performing the proof presentation process, which is able to validate the possession of certain attributes or statements in the presented partial identity. The proof presentation process is detailed in Section 4.1.3 of Deliverable D2.1 "*IoT Communities and Identity Management*". Once authenticated by the SocIoTal IdM, the Fi-ware IdM implementation would continue as usual generating the OAuth token.

It should be noticed, that a detailed software integration design of this integration has not been detailed yet since it has still not been decided whether it will be implemented in the scope of the project or not. In case the extension is finally implemented, future project deliverables (like D2.3) would describe the implementation and extension details.

Section 6 - Conclusion

Under the main objectives of security SocIoTal goals, as well as the definition of the ARM-compliant security framework presented in D2.1, this deliverable has provided the definition of the main interactions of such framework in order to show how access control and privacy are addressed by SocIoTal. Specifically, an integral access control system has been provided as a combination of different authorization technologies and tools in order to enable a suitable solution for IoT environments. Such system is based on the use of XACML access control policies, which are employed to generate capability tokens, considering contextual information and trust values to drive access control decisions.

Furthermore, this document has also provided the SocIoTal Context Model based on the specification of virtual entities as well as the OMA NGSI Context Management specification. This model is deployed by the Context Manager functional component of the SocIoTal security framework and used by other security functional components in order to realize a context-aware security paradigm for the IoT. Thus, the main interactions for context-aware identity management, access control and group sharing mechanisms has been provided.

In addition, starting from the reference definition of communities and bubbles given in D2.1, the document provided an explanation about how the proposed security solutions can be used for an effective and secure management of groups of smart objects. While this work is part of the Task 3.3. of the project, this document has supplied the main notions in order to enable a secure group sharing based on the proposed SocIoTal access control system, as well as a mechanism making use of the CP-ABE cryptographic scheme.

Finally, this deliverable has offered the main foundations for integration with the well-known Fi-WARE platform. Specifically, the extension of such platform to support the envisioned security solutions (as part of the SocIoTal security framework) has been provided. This integration is intended to provide SocIoTal security tools to a greater users and developers community.

Section 7 - References

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

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

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

[4] Vincent C. et a. NIST Special Publication 800-162. Guide to Attribute Based Access Control (ABAC) Definition and Considerations http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf

[5] From ABAC to ZBAC: The Evolution of Access Control Models http://www.hpl.hp.com/techreports/2009/HPL-2009-30.pdf

[6] OpenID Authentication 2.0, Finalized OpenID Specification (December 2007)

[7] SAML (Security Assertion Markup Language): http://docs.oasis-open.org/security/saml/v2.0

[8] D. Hardt. "The OAuth 2.0 Authorization Framework". IETF RFC 6749, 2012.

[9] U-Prove Cryptographic Specification V1.1. Revision 3. Microsoft Corporation Authors: Christian Paquin, Greg Zaverucha, December 2013. Available at http://research.microsoft.com/en-us/projects/u-prove/

[10] Baier, D., Bertocci, V., Brown, K., Woloski, M., & Pace, E. (2010). *A Guide to Claims-Based Identity and Access Control: Patterns & Practices*. Microsoft Press.

[11] Specification of the Identity Mixer Cryptographic Library. Version 2.3.40 IBM Research, Zurich January 30, 2013. Available http://www.zurich.ibm.com/idemix/

[12] J. Camenisch, and A. Lysyanskaya, A. "Signature schemes and anonymous credentials from bilinear maps". In *Advances in Cryptology–CRYPTO 2004* (pp. 56-72). Springer Berlin Heidelberg.

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

[14] Fi-WARE FP7 EU project. Available http://www.fi-ware.org/

[15] Butler FP7 EU project (ID 287901). Available: http://www.iot-butler.eu/

[16] IoT@Work FP7 EU project. Available http://www.iot-at-work

[17] S. Gusmeroli, S. Piccione, and D. Rotondi, A capability-based security approach to manage access control in the internet of things, Math. Comput. Model. 58(5–6) (2013), pp. 1189–1205.

[18] Deering, S. E. (1998). Internet protocol, version 6 (IPv6) specification.

[19] G. Zhang and W. Gong, *The research of access control based on UCON in the internet of things*, J. Softw. 6(4) (2011), pp. 724–731.

[20] R. Sandhu and J. Park, *Usage control: A vision for next generation access control*, in *Computer Network Security*. MMM-ACNS2003, LNCS,Vol. 2776,V. Gorodetsky, L. Popyack, andV. Skormin, eds., Springer, Berlin, Heidelberg, pp. 17–31, 2003.

[21] P.N. Mahalle, P. Thakre, N.R. Prasad, and R. Prasad, *A fuzzy approach to trust based access control in internet of things*,Wireless Communications,Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), 3rd International Conference on, Atlantic City, NJ, USA, 2013, pp. 1–5.

[22] J. Liu,Y. Xiao, and C.L.P. Chen, *Authentication and access control in the internet of things*, Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW), Macau, China, IEEE, June 2012, pp. 588–592.

[23] Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., & Ylonen, T. (1999). *SPKI certificate theory*. IETF RFC 2693, September.

[24] J. Dennis and E. Van Horn, *Programming semantics for multiprogrammed computations*, Commun. ACM 9(3) (1966), pp. 143–155.

[25] M. Naedele, *An access control protocol for embedded devices*, Industrial Informatics, 2006 IEEE International Conference on, IEEE, 2006, pp. 565–569.

[26] P.N. Mahalle, B. Anggorojati, N.R. Prasad, and R. Prasad, *Identity driven capability based access control (ICAC) for the internet of things*, Proceedings of the 6th IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bangalore, India, IEEE, December, 2012, pp. 49–54.

[27] J. L. Hernández-Ramos, A. J. Jara, L. Marín, and A. F. Skarmeta, "Dcapbac: Embedding authorization logic into smart things through ecc optimizations," International Journal of Computer Mathematics, no. just-accepted, pp. 1–22, 2014.

[28] Crockford, D. (2006). The application/json media type for javascript object notation (json).

[29] Shelby, Z., Hartke, K., Bormann, C., & Frank, B. (2014). RFC 7252: The Constrained Application Protocol (CoAP). *Internet Engineering Task Force*.

[30] OASIS Public Review Draft. JSON Profile of XACML 3.0 Version 1.0. May 2014: http://docs.oasis-open.org/xacml/xacml-json-http/v1.0/csprd03/xacml-json-http-v1.0-csprd03.pdf

[31] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," Computer Networks, vol. 57, no. 10, pp. 2266–2279, 2013.

[32] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the ip-based internet of things," Wireless Personal Communications, vol. 61, no. 3, pp. 527–542, 2011.

[33] H. Yu, J. He, T. Zhang, P. Xiao, and Y. Zhang, "Enabling end-to-end secure communication between wireless sensor networks and the internet," World Wide Web, pp. 1–26, 2013.

[34] M. Jones, J. Bradley, and N.Sakimura, *JSON Web Token (JWT)*, OAuth Working Group, Internet Engineering Task Force (IETF), work in progress, draft-ietf-oauth-json-web-token-11, July 2013. Available at http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-11.

[35] S. Li, J. Hoebeke, F. Van den Abeele, and A. Jara, *Conditional observe in CoAP*, Constrained resources (CoRE) Working group, Internet Engineering Task Force (IETF),work in progress, draft-li-core-conditionalobserve-04, June 2013. Available at http://tools.ietf.org/html/draft-li-core-conditional-observe-04.

[36] C. Jennings, J. Arkko, and Z. Shelby, *Media types for sensor markup language (SENML)*, NetworkWorking group, Internet Engineering Task Force (IETF), Work in Progress, draft-jennings-senml-10, October 2012. Available at http://tools.ietf.org/html/draft-jennings-senml-10.

[37] Dey, A. K. (2001). Understanding and using context. Personal and ubiquitous computing, 5(1), 4-7

[38] SocIoTal Deliverable. D3.1.1 Device centric enablers for privacy and trust, WP3 – Privacy-aware communication

[39] SocIoTal Deliverable. D1.3.1 First version of API Specification, WP1 – Socially-aware citizen centric architecture and community APIs

[40] mojNS Android application on Google Play, https://play.google.com/store/apps/details?id=com.eu.smartsantander.participatorysensing.mojns accessed on 11.07.2014.

[41] A. Ignjatovic, N. Foo, and C. T. Lee, An analytic approach to reputation ranking of participants in online transactions," in The 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Volume 01, (Washington, DC, USA), pp. 587{590, IEEE Computer Society, 2008.

[42] A. Flanagin, M. Metzger, R. Pure, and A. Markov, User-generated ratings and the evaluation of credibility and product quality in ecommerce transactions," in System Sciences (HICSS), 2011 44th Hawaii International Conference on, pp. 1{10, IEEE, 2011.

[43] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar, Quality control in crowdsourcing systems: Issues and directions," Internet Computing, IEEE, vol. 17, no. 2, pp. 76{81, 2013.

[44] Mohammad Allahbakhsh, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Boualem Benatallah, "Robust evaluation of products and reviewers in social rating systems", World Wide Web 2013

[45] SocIoTal Deliverable. D2.1 IoT Communities and Identity Management, WP2 – Decentralised governance and trust framework

[46] Marti, S., & Garcia-Molina, H. (2006). Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, *50*(4), 472-484.

[47] Bassi, A., Bauer, M., Fiedler, M., Kramp, T., Kranenburg, R., Lange, S., & Meissner, S. (2013). *Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model*. Springer.

[48] IoT-A FP7 EU project. Available http://www.iot-a.eu

[49] SocIoTal Deliverable. D1.2.1 First version of SocIoTal Architecture, WP1 – Socially-aware citizen centric architecture and community APIs

[50] NGSI Context Management. Open Mobile Alliance (OMA). May 2012.

[51] Open Geospatial Consortium "OGC SensorML: Model and XML Encoding Standard", Version: 2.0.0, Ed: Mike Botts, Publication Date: 2014-02-04

[52] Compton, M., Barnaghi, P., Bermudez, L., GarcíA-Castro, R., Corcho, O., Cox, S. & Taylor, K. (2012). The SSN ontology of the W3C semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web, 17, 25-32

[53] Luckham, D. (2002). *The power of events* (Vol. 204). Reading: Addison-Wesley.

[54] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007, pp. 321–334.

[55] FI-WARE Access Control Generic Enabler. http://catalogue.fi-ware.org/enablers/access-control-tha-implementation

[56] FI-WARE wiki. https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Welcome_to_the_FI-WARE_Wiki

[57] Moses, T. eXtensible Access Control Markup Language (XACML) Version 2.0 OASIS Standard Feb. 1, 2005 OASIS Open. *Source: http://docs. oasis-open. org/xacml/2.0/access—control-xacml-2.0-core-spec-os. pdf see also http://www. oasis-open. org/committees/tc—home. php*.

[58] FI-WARE IdM Generic Enabler. http://catalogue.fi-ware.org/enablers/identity-management-keyrock

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

[60] SCIM. System for Cross-Domain Identity Management: Core Schema. Network Working Group. Internet-Draft. P. Hunt, et. al.. draft-ietf-scim-core-schema-13, November 14, 2014