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:
M30 – February 2016
Actual submission date:
14/03/2016

Deliverable ID:	**D2.4**
Deliverable Title:	**Components and functions for a decentralized governance and trust framework**
Responsible beneficiary:	7/UME
Contributing beneficiaries:	UME, UNIS, UMU, CEA, DNET
Estimated Indicative Person Months:	10

Start Date of the Project: 1 September 2013 Duration: 36 Months
Revision: Final
Dissemination Level: Public

Document Information

Document ID:	D2.4
Version:	0.15
Version Date:	14. March 2016
Authors:	Yee Wei Law, Colin O'Reilly, Niklas Palaghias, Nenad Gligoric, Benoit Denis, Etienne Gandrille, Christine Hennebert, Carmen Lopez de la Torre, Ignacio Elicegui Maestro, Jose Luis Hernandez Ramos, Jorge Bernal Bernabe, Antonio Skarmeta Gomez
Security:	**Confidential**

Approvals

	Name	Organization	Date	Visa
Project Management Team	Klaus Moessner	UNIS		
Internal Reviewer	Rob Van Kranenburg	RD		
Internal Reviewer	Antonio Pintus	CRS4		

Document history

Revision	Date	Modification	Authors
0.1	24/11/2015	First ToC	UME
0.2	26/01/2016	Addition to sections 4,5,6 and 7	CEA,UMU,UNIS
0.3	26/01/2016	Merging several parallel 0.2 versions into 0.3	UNIS
0.4	27/01/2016	Revision of sections 7.1.1 to 7.1.3	CEA
0.5	08/02/2016	Addition to Section 7.2	UNIS
0.6	10/02/2015	Revision of eq. format in 7.1.1 to 7.1.3	CEA
0.7	12/02/2016	First edit of revision 0.6	UME
0.8	15/02/2016	Integration of Jorge's contribution	UME
0.9	20/02/2016	Additional content	UMU
0.10	24/02/2016	Additional content	UME
0.11	27/02/2016	Additional content	UNIS
0.12	28/02/2016	Additional content	CEA
0.13	29/02/2016	Revised Executive Summary, Introduction and Conclusion. Edited other parts as well. All done with "Track Changes" off.	UME
0.14	09/03/2016	Addressing reviewers comments	UME, UNIS
0.15	14/03/2016	Final clean version	UNIS

Content

Table of Figures

Executive summary

Description of the deliverable content and purpose

The objective of WP2 is to enable more automated forms of discovery and specification of trust relationships between humans and a translation of those to their devices as well as the establishment of reputation of IoT devices/data providers. Tasks 2.1—2.4 have been designed to achieve this objective. The SocIoTal security framework produced by these tasks consists of:

- A Group Management module, which facilitates creation of secure groups using Ciphertext-Policy Attribute-Based Encryption.
- A Key Management module, which provides the mechanisms and protocols for pre-distributing cryptographic keys and establishing secure channels between IoT devices.
- An Authentication module, which facilitates authentication of users based on login passwords, public-key certificates, and anonymous credentials.
- An Identity Management module, which enable users to use different partial identities to access target devices depending on the context.
- An Authorization module, which is responsible for making attribute-based access control decisions, and enforcing them through the use of capability tokens
- A Trust & Reputation Management module, which provides continuous assessment of a user's trustworthiness to satisfy strong security needs.
- A Context Management module, which provides the functionalities to store, query and retrieve contextual information for the purpose of device discovery.

This deliverable D2.4

- defines the functionalities and services provided by the aforementioned components;
- reports on their implementation status and performance results;
- reports on the level of integration of these components so far.

Section 1 - Overview

The SocIoTal project adopts the Architectural Reference Model (ARM) of the IoT-A project [1]. As an instantiation of the security functional group of this model, the SocIoTal security framework features seven modules (see Figure 1):

- **Group Manager**: This module enables information sharing, in a secure and private manner, between groups of entities (Communities or Bubbles) which satisfy a certain set of identity attributes values.
- **Key Management**: This module is responsible for managing cryptographic keys and establishing secure channels between IoT devices.
- **Authentication**: This module verifies if a subject is indeed who or what it claims to be.
- **Identity Management**: This module is responsible for managing the identities of the users and smart objects.
- **Authorization**: This module is responsible for making authorization decisions based on access control policies.
- **Trust & Reputation Management**: This module is for establishing a trusted and reliable environment where users can interact with IoT services while preserving privacy.
- **Context Manager**: This module informs other modules of the current context of user interaction.

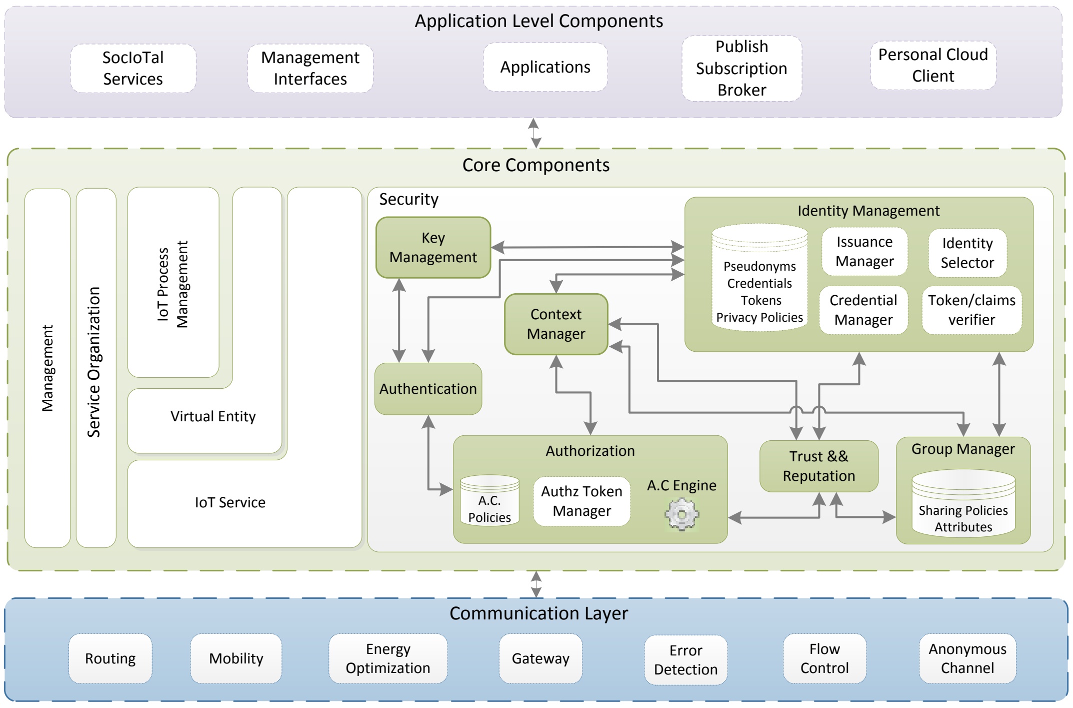


Figure 1: The SocIoTal security framework within the Architectural Reference Model

Each of the modules above contains innovations contributed by SocIoTal, and these innovations are discussed in Section 2—8. In integration, these modules enable decentralized IoT governance and trust management. Here, an overview of how these modules interacts with each other is provided. This overview takes the form of a sample use case adapted from the SocIoTal proposal.

In some hypothetical neighbourhood, a few elderly people live by themselves, without any of their family members close-by. By wearing their smartphone or specialised tracking devices, their current activities are monitored so that in case of emergency (e.g., detected fall or lack of activity for long periods at unusual times), an alarm can be signalled to emergency services and close-by "angel" community members. A motion sensor in the entrance area of the house or interpersonal contact measured through smart phone apps can quickly identify periods of increased isolation. The community watch service is developed by one of the IT developers living in the area and it enables collaborative monitoring of elderly people as well as children by trusted community members. Information from smart phones/tracking devices is processed, cross-correlated with information generated by other relevant sensors (for example cameras showing the area where the person is located at a given moment) and in case of alarms forwarded to the person who can provide the best support—based on the current location and/or community profile. Every access to the sensors and tracking devices is audited and logged and made available in a daily weekly digest in the intuitive user environment. This allowed the senior residents to learn of the trusted members that take interest in their life activities, and one of the senior residents to send out small Christmas gifts to thank those empathic individuals.

In the use case above,

- The *Group Manager* enables the creation of the "Angels" community bubble.
- Anyone can join the "Angels" bubble and is granted access credentials by the *Key Management* module.
- The access credentials are for authentication by the *Authentication* module.
- Any elderly who would like to share their information with members of the "Angles" bubble can take advantage of identity delegation through the *Identity Management* module, where the elderly allows its devices to use the his/her partial identities to act on his/her behalf.
- Every basic membership in the "Angels" bubble is granted with a set of basic rights by the *Authorization* module, e.g., access to social network updates of the elderlies.
- Only the community members who have earned enough score from the *Trust & Reputation* Management module are given more advanced rights, e.g., access to the data streams from the elderlies' health monitoring devices; the higher the trust score of the data subscribers, the more detailed the information they receive.
- The data sent to the subscribers contain contextual information determined by the *Context Manager*.

Thus, every module in the SocIoTal security framework has a crucial role. In the subsequent sections, the latest advances in each of these modules, including details about their implementation and integration status, will be discussed.

Section 2 - Group Manager

The Group Manager functional component of the SocIoTal security framework is based on the use of the *Ciphertext-Policy Attribute-Based Encryption* (CP-ABE) cryptographic scheme [2], in order to enable a secure data sharing mechanism with groups of entities (i.e. bubbles of smart objects). CP-ABE represents the generalization of *Identity-Based Encryption* (IBE) [3], in which the key is associated with a set of identity attributes, while a ciphertext is encrypted under a combination of such attributes. Thus, only smart objects with a CP-ABE key satisfying such combination, will be able to decrypt the information. This cryptographic scheme provides significant features and a noteworthy potential to be exploited in IoT environments. On the one hand, a smart object, acting as a data producer, can decide how its information is disseminated to other entities by encrypting each piece of information with a different combination of identity attributes. Indeed, unlike the use of symmetric-key cryptography, in which groups of entities must be pre-distributed with the same key, a smart object could encrypt each data under a different combination of attributes, allowing the creation of dynamic groups (or subgroups). For example, a smart object could encrypt information so that only the set of objects from the same manufacturer or the same owner could decrypt the information.

The CP-ABE scheme has been combined with other technologies and components for the design and implementation of group sharing mechanism. In this sense, it has been integrated with the FI-WARE IdM, so CP-ABE keys that are generated for users are associated with their identity attributes, which are stored in the Keyrock IdM instance. Moreover, this mechanism enables sharing of encrypted information related to the entities registered in the Context Manager entity, so only users with the proper CP-ABE keys are allowed to decrypt such data. A more detailed description about the main aspects of such mechanism can be found in D3.3 [19].

2.1 Secure Group Communication

The realization of the SocIoTal group sharing mechanism has been carried out through the instantiation of two main components:

1. **Group Manager Server (Attribute Authority)**. The Group Manager Server or Attribute Authority (AA) is a HTTPS server accepting requests for CP-ABE keys generation. CP-ABE keys that are generated by the AA are associated with the attributes stored in the Keyrock IdM.
2. **Group Manager Client**. The Group Manager Client is a HTTPS client, which is intended for making requests to the AA to obtain CP-ABE keys, which are used to share information with a group or bubble of entities in a secure way. Additionally, this component is responsible for making the basic encryption/decryption operations, as well as for sharing information with the SocIoTal Context Manager through the use of NGSI-9/NGSI-10 interfaces.

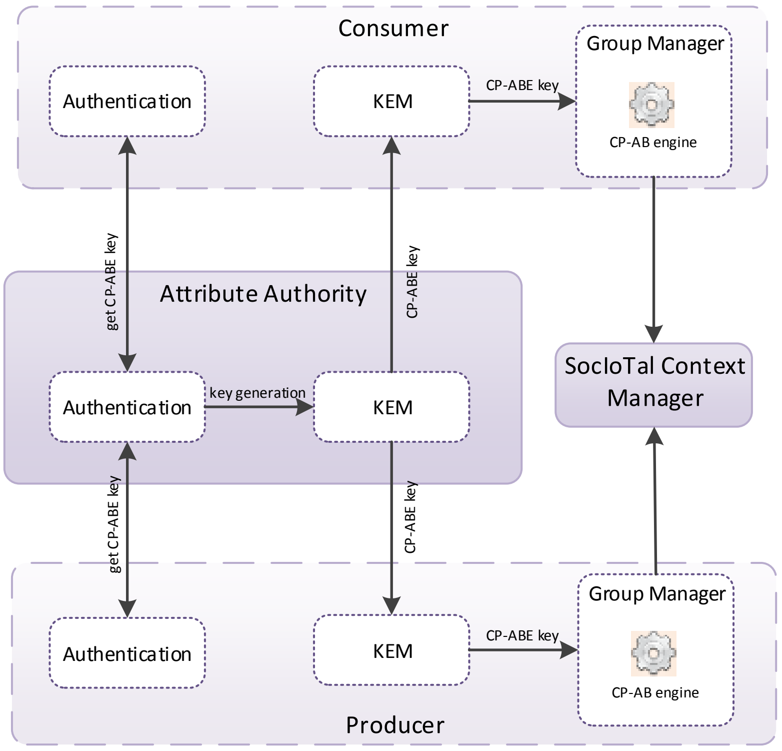


Figure 2: SocIoTal framework interactions for secure group communication

Figure 2 shows the main framework interactions and how they are instantiated for the SocIoTal secure group communication mechanism. It provides a high-level description of the application of this mechanism in the context of the project and how it has been instantiated. Taking into account these components and interactions, Figure 2 shows the SocIoTal secure group communication scenario based on the use of CP-ABE. According to Figure 3, User 1 and User 2 are acting as Group Manager Clients, which are intended to share information via the Context Manager in a secure way. Before this process can be performed, they need to get CP-ABE keys associated to their identity attributes. This process is shown in the figure in messages 1-6 and, for the sake of clarity, it is only displayed for User 1. During this procedure, a user acting as a Group Manager Client, applies for a CP-ABE Key to the AA via a HTTPS request. Then, the AA gets the user ID from her certificate, and it queries her attributes associated to the Keyrock IdM. These attributes are employed to generate the corresponding CP-ABE key and deliver it to the final user. It should be noted that this process could require the generation and delivery of other required cryptographic material, such as the public parameters, which are needed for basic CP-ABE algorithms.

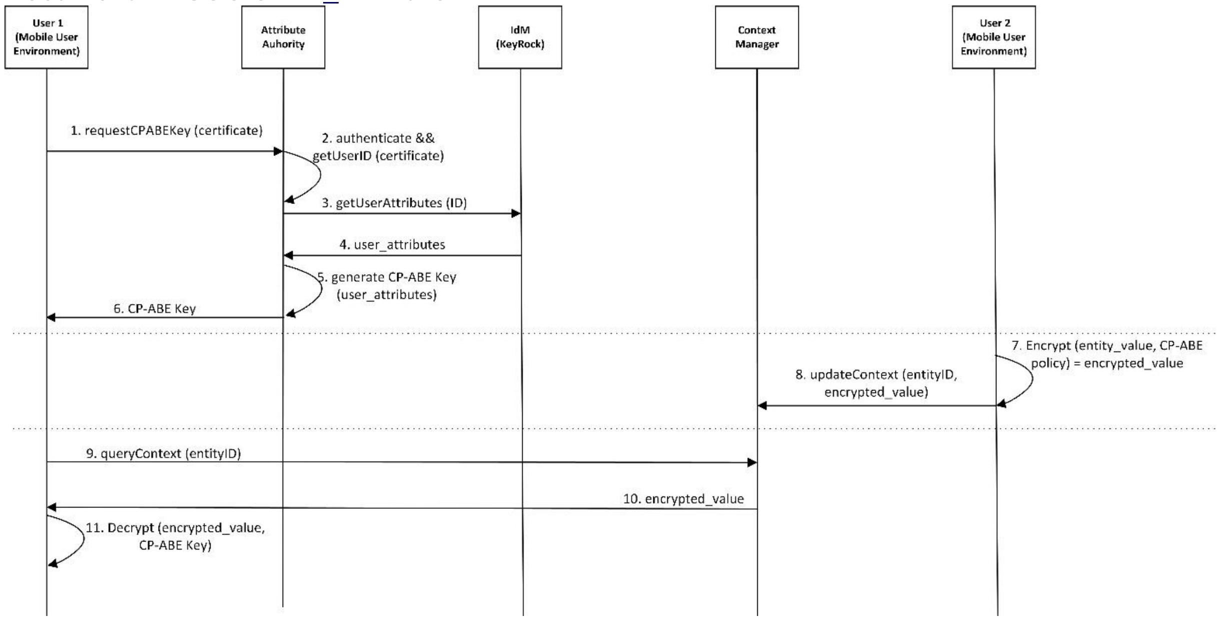

User 1 (Mobile User Environment)
Attribute Auhority
IdM (KeyRock)
Context Manager
User 2 (Mobile User Environment)
1. requestCPABEKey (certificate)
2. authenticate && getUserID (certificate)
3. getUserAttributes (ID)
4. user_attributes
5. generate CP-ABE Key (user_attributes)
6. CP-ABE Key
7. Encrypt (entity_value, CP-ABE policy) = encrypted_value
8. updateContext (entityID, encrypted_value)
9. queryContext (entityID)
10. encrypted_value
11. Decrypt (encrypted_value, CP-ABE Key)

Figure 3: **SocIoTal secure group communication through CP-ABE**

After users are endowed with the required keys and parameters, they are enabled to share encrypted information through the Context Manager by using CP-ABE. In this case, User 2 decides to update a certain entity's attribute *att* by encrypting its value by using a specific combination of identity attributes or CP-ABE policy *pol*. For this purpose, they use the updateContext method from NGSI10 to communicate the new value to the Context Manager. Then, User 1 makes a query to know the new value of *att*. Towards this end, again, they make use of the queryContext method and they get the encrypted value of *att*. Then, by using the CP-ABE key that was previously obtained, they try to decrypt the value. In case its CP-ABE key satisfies *pol*, they will be able to decrypt the *att*'s value. A more detailed description of this process, as well as a set of evaluation results can be found in D3.3 [19].

2.2 Implementation and Integration

The implementation of the Group Manager entities (client and server) has been realized through the deployment of a Java-based CP-ABE library, which was used to implement the core functionality of the sharing scheme. This library, whose details and experimentation results are provided at D3.3, is based on the Java Pairing Based Cryptography library (jPBC) [29], and uses type A pairings, which are built on the supersingular curve y2 = x3 + x over the field Fp for some prime p = 3 mod 4. Let p be the prime order of Fp, and E(Fp), the additive group of points of affine coordinates (x, y) with x, y in Fp, that satisfy the curve equation, q represents the order of the cyclic subgroup of interest in E(Fp). The security level of the scheme depends on the size of primes p and q. It should be pointed out that for this implementation it is considered an 80-bits security level (i.e., |p| = 512, |q| = 160).

The Group Manager Server (Attribute Authority) has been implemented as a Java servlet and deployed in UMU premises [30], which is responsible to generate and deliver CP-ABE keys associated to the identity attributes of the requesting user. These attributes are obtained from the IdM Keyrock instance. Specifically, *setup* and *keygen* CP-ABE algorithms are implemented and deployed in this component.

The Group Manager Client has been realized with a Java library on top the CP-ABE library that allows apply for CP-ABE keys, as well as to share information through the Context Manager in a secure way. This library has been used for the development of the Group Sharing app, which is an Android application intended to enable an information sharing mechanism with a group of entities. In particular, this app makes use of the Group Manager Client library to encrypt data, which is shared through the SocIoTal Context Manager (CM). By running this application, an entity can act as a context producer, encrypting data with a specific CP-ABE policy and publishing this information to the CM. Furthermore, the same entity can act as a context consumer, receiving encrypted notifications or making queries about an entity to the CM and trying to decrypt them with its CP-ABE keys. Figure 4 shows a screen examples about the developed app. The initial screen (on the left side) allows to specify the IP address and port in which to receive notifications from the CM, and the IP address of this entity.

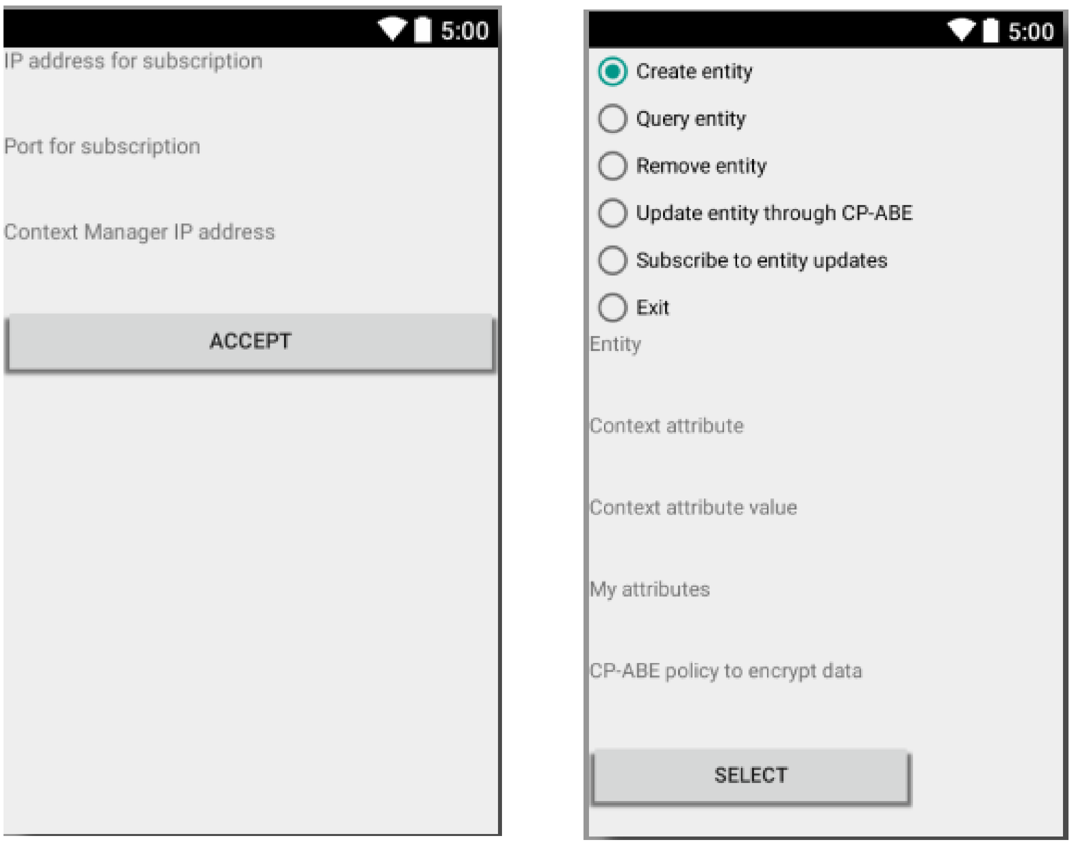


Figure 4: Screen examples of the Group Sharing app

Moreover, the main screen of the app (on the right side) is making reference to the main NGSI operations that can be performed on the CM (e.g. queryContext or updateContext). It should be noted that the *CP-ABE policy to encrypt data* field is intended to specify the combination of identity attributes that will be used to encrypt the value, when the *Update entity through CP-ABE* option is indicated.

2.3 Communities Communication

SocIoTal Communities manager tool allows users to create communities to share information with other users within a safe environment avoiding the leak of information. In [19] and [20] it can be found an initial description of the SocIoTal communities' definition and an updated description will be find in [23]. This section will present a brief summary of the tool with further information about the integration with other blocks of the SocIoTal framework.

The first step to make use of communities' tool is the registration of the user within SocIoTal. This task can be accomplished either through the SocIoTal Communities Manager [22] or directly using the SocIoTal Identity Manager (IdM). Both ways will create a registry in the SocIoTal IdM. The physical user will provide (at least) a name and a password and the IdM will create a new virtual user (or identity) within the platform. A physical user can manage just one or several virtual users or identities through the SocIoTal IdM. In addition to this, when the registration process is done through the Communities Manager, a private community for the user will be automatically created, and the "owner" role will be associated to the user for this community. This processes are presented in Figure 5.

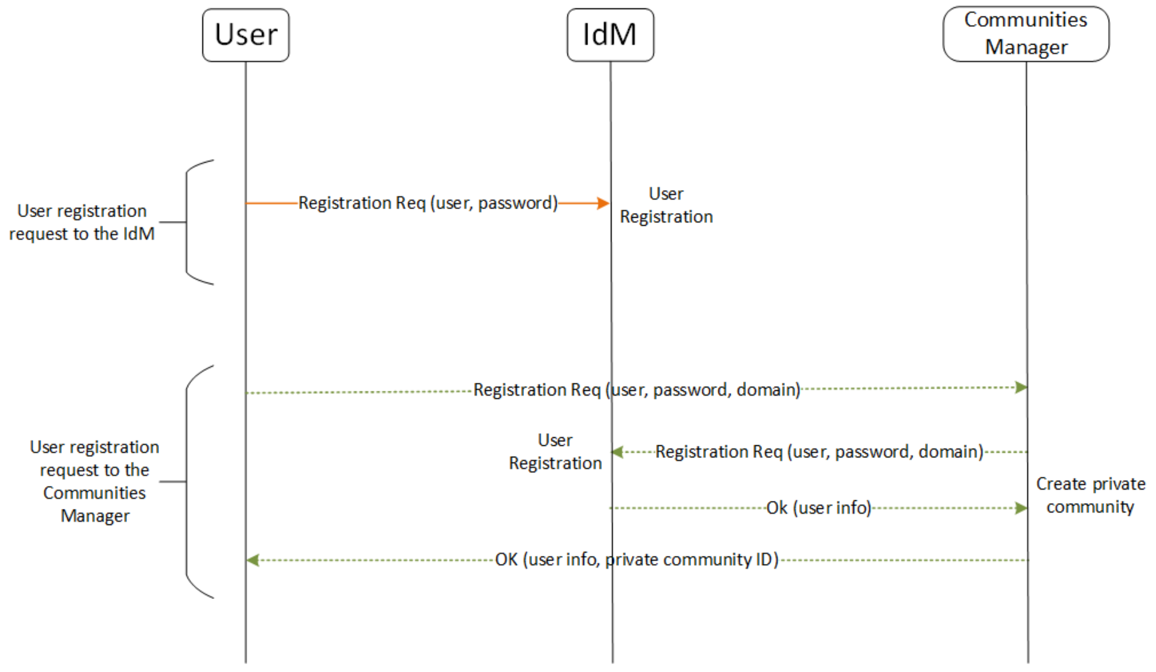

User
IdM
Communities Manager
User registration request to the IdM
Registration Req (user, password)
User Registration
User registration request to the Communities Manager
Registration Req (user, password, domain)
User Registration
Registration Req (user, password, domain)
Create private community
Ok (user info)
OK (user info, private community ID)

Figure 5: User registration process

In order to perform actions over the private community, and the same will happen with other communities, the users will need a key called "Community-Token" that will relate the user, with the community and the role the user plays within that community. Community-Tokens are managed by the Communities Manager. To request this token, the user will be authenticated against the Communities Manager through his/her name (or "id") and password and should point community he/she is requesting the token for. If the authentication is correct, and the user belongs to the pointed community, a Community-Token, identifying the user, role, community and its period of validity will be generated. Later, within SocIoTal Communities Manager, this Community-Token will be required to perform any request related to the management of the communities (create, modify, list members, assign roles, etc.) in order to check the requestor community. SocIoTal Context Manager will also requires and check this Community-Token when registering an entity, identifying, this way, the owner and the community this entity will belong to, and also to perform actions over registered resources, using it to identify the requestor user and his/her community belonging. The info provided by the attached Community-Token let Context Manager know if the requestor is allowed to perform the requested action within the corresponding community. This is: when a user wants to register a resource, they will attach the previously requested community-token related to the community to which they want to add the resource. Besides, a valid community-token will be also needed when a user wants to search for entities registered in the platform within communities or update an entity belonging to a specified owner.

As presented before, the main functionality of the communities' tool is to provide SocIoTal users with a means to create new groups in order to share resources with other users plus offer them mechanisms to identify and authenticate other users and check the communities' membership. This way, SocIoTal communities intend to be close environments, grouping users and entities available only for communities' members. The SocIoTal Communities Manager includes also a role schema, which allows communities' owners to define who can do what (read/write entities, add new users, modify roles, etc.) within their communities.

Once a virtual SocIoTal user has been registered, he/she will be able to create new communities. To do so, at least a community name, a domain and a brief description is required, plus a Community-Token that links the creator user to the selected domain (a community-token can be linked to a community and a domain or only to a domain). The proper way to request tokens are described in SocIoTal Wiki [24]. The next step will be to request a token so as to perform actions within the new community. As a creator of the community, the user will obtain the role of "owner" (by default) and will be able to perform most of the manager actions such as add resources, add or remove other users to the community, etc. This is, the "owner" role provides the user the means to be the manager of the community by using his community-token. It can be highlighted that there can be as many owners as the creator of the community wants, by assigning the "owner" role when adding a user to the community. An example of this process is presented in Figure 6, and an extended information can be found in [19].

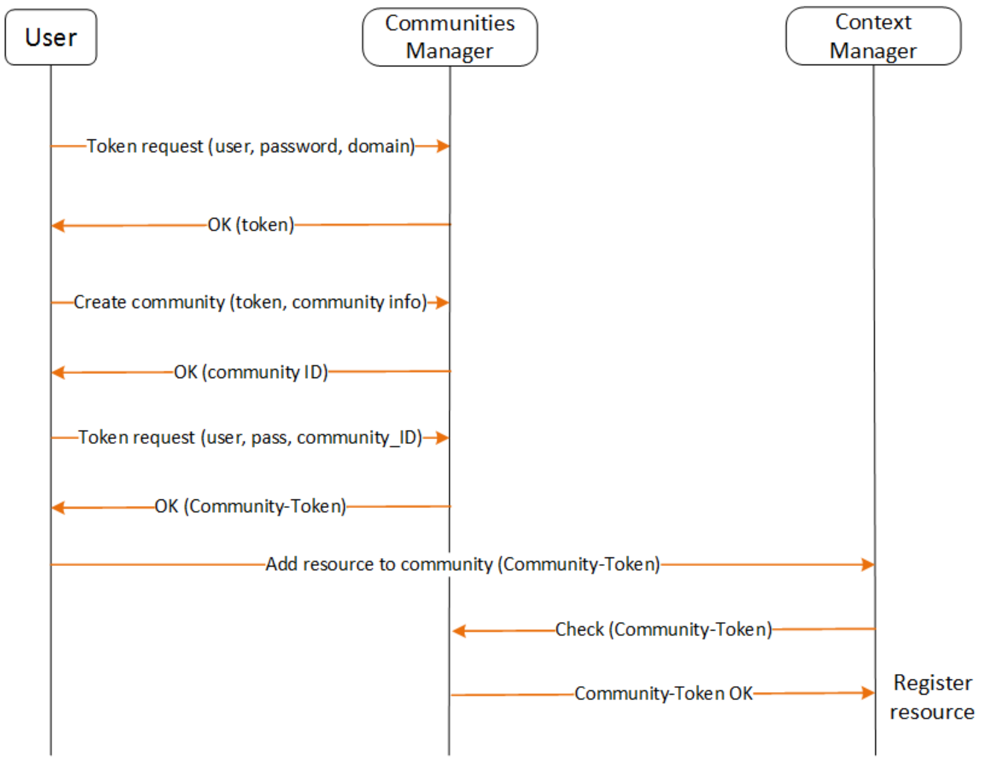

User
Communities Manager
Context Manager
Token request (user, password, domain)
OK (token)
Create community (token, community info)
OK (community ID)
Token request (user, pass, community_ID)
OK (Community-Token)
Add resource to community (Community-Token)
Check (Community-Token)
Community-Token OK
Register resource

Figure 6: Community Creation example

The Communities Manager will be presented to the user through two main interfaces. On the one hand, developers will be able to use the set of APIs available in [22]. They will be able to interact with the platform through the APIs, or to create new services using its functionalities.

On the other hand, the SocIoTal User Environment will integrate the Communities Manager APIs so as the users will be able to interact with the tool in a user-friendly manner. More detailed info related to SocIoTal Communities Manager can be found in D3.2.2 [23].

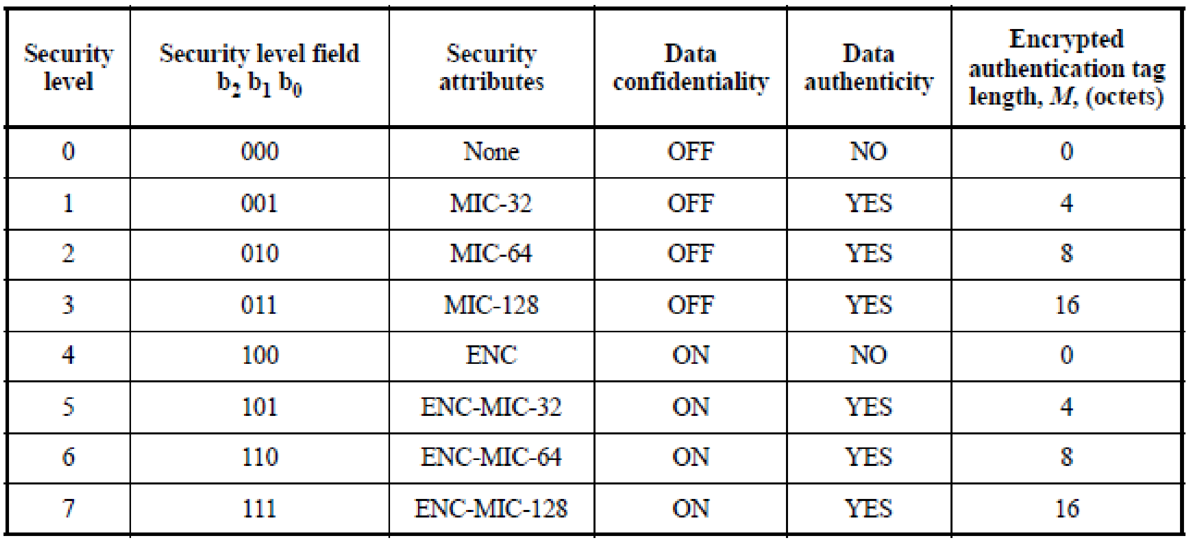
Section 3 - Key Management

The scheme presented in D3.2.1 [27] and protected by the patent (French patent 16 50610) consisting in the generation and the use of pseudonyms to hide both the MAC and the IPv6 addresses of the source and destination of a message sent over a 6LoWPAN meshed network involves the management of:

1) A security Key, called SKey, at the link layer, the IEEE 802.15.4 MAC layer;
2) A shared symmetric key for the pseudonym management, called “privacy” key or LKey, managed at the link layer too.

3.1 Management of the security key at IEEE 802.15.4 MAC layer

Security features are provided by the IEEE 802.15.4 standard since the first version of the standard in 2003 [17]. In its version of 2006 [18], the standard mandates the support of AES-CCM* security suite (ANSI X9.63-2001, and Appendix A of NIST Pub 800-38C). It is a Counter CBC-MAC mode of operation for cryptographic AES block ciphers. It is designed to provide both authentication, integrity and confidentiality. CCM mode is defined for block ciphers with a block length of 128 bits and for CBC-MAC results on 32, 64 or 128 bits. Eight levels of security are defined from no security (level 0), CBC-MAC only (32, 64 or 128 bits) (levels 1 to 3), encryption only (level 4) or both CBC-MAC (32, 64 or 128 bits) and encryption (levels 5 to 7) (see on Figure 7). No specifications are provided by any standard for the secret key distribution and management.



Security level	Security level field $b_2\ b_1\ b_0$	Security attributes	Data confidentiality	Data authenticity	Encrypted authentication tag length, *M*, (octets)
0	000	None	OFF	NO	0
1	001	MIC-32	OFF	YES	4
2	010	MIC-64	OFF	YES	8
3	011	MIC-128	OFF	YES	16
4	100	ENC	ON	NO	0
5	101	ENC-MIC-32	ON	YES	4
6	110	ENC-MIC-64	ON	YES	8
7	111	ENC-MIC-128	ON	YES	16

Figure 7: Security level at IEEE 802.15.4 MAC layer

However, the paper [35] details several vulnerabilities in the specifications of the security of the version 2006 of the standard. These leaks have been resolved in the latest version dating from 2011 [33] and its amendment IEEE 802.15.4e in 2012 [34].

Despite these security recommendations, few hardware and compliant embedded software technologies were implementing the support of the security at the link layer until few months. Contiki OS supports the security feature of the version 2006 of the standard in its latest release Contiki 3.0 from October 2015 and the distribution still under development Contiki 3.x. This newer distribution are supported and compliant with the openMote hardware technology (http://www.openmote.com/).

So, these technology will be used in the Sociotal project to provide of proof-of-concept of the management of the security at the Link layer and the generation and the use of pseudonyms for the MAC and IP addresses of a 6LoWPAN meshed network in order to provide a tunnel-like mode for communications into an IEEE 802.15.4 wireless sensor network.

The openMote meshed network is linked to the Sociotal platform thanks to a local gateway holding an OSGI, that is able to communicate with sensinact gateway by sending POST restful messages (see Figure 22).

The local gateway supports a Debian Linux distribution. The latest kernel of Linux 4.4 from October 2015 provides some tools to manage the security at the IEEE 02.15.4 MAC layer, compliant with the version 2011 of the standard, in the 6LoWPAN stack embedded in the core.

In the following, we present the implementation of the distribution and the management along time of the symmetric secret key, shared by all the motes composing the 6LoWPAN network and the local Gateway. This key is indexed by the time as it is periodically renewed and is called SKey(t_q). The initial key, known by the motes before their deployment is SKey(t_0).

When a mote would like to join the secured network, it wakes up by requesting the current SKey(t_q) to the gateway. The gateway responds by sending SKey(tq) encrypted with the initial SKey(t_0). At receiving, the mote stores the current SKey(t_q) in its RAM memory and is able to communicate securely over the 6LoWPAN network.

3.1.1 Implementation of the security management in Linux 4.4

The latest Linux 4.4 kernel includes the 6LoWPAN stack and features to manage the security at the link layer of the IEEE 802.15.4 standard version 2011. The debian 4.4 and raspBian 4.4 distributions holds these features. The local gateway is implemented as an OSGI running on raspBian 4.4.

The frame includes in the MAC header a field entitled “Security Control field” composed of two sub-fields, the “Security Level” (see Figure 7) and the “Key Identifier Mode” (see Figure 8). Only the two first mode are supported by Contiki, enabling the use of a key determined implicitly (mode 0x00), or the use of a key determined explicitly (mode 0x01) from the Key Index field that determines the chosen key in a key table.

Key identifier mode	Key Identifier Mode field b_1 b_0	Description	Key Identifier field length (octets)
0x00	'00'	Key is determined implicitly from the originator and recipient(s) of the frame, as indicated in the frame header.	0
0x01	'01'	Key is determined from the Key Index field in conjunction with *macDefault-KeySource*.	1
0x02	'10'	Key is determined explicitly from the 4-octet Key Source field and the Key Index field.	5
0x03	'11'	Key is determined explicitly from the 8-octet Key Source field and the Key Index field.	9

Figure 8: Key Identifier mode

The implicit mode is used to register the current key SKey(t_q) linked to the network PANID and to the destination mote MAC address.

But, while the mote has not yet joined the network, its MAC address is not yet registered. So, at the bootstrap phase, the initial key SKey(t_0) is used to encrypt the value of the current key SKey(t_q). This initial key is stored thanks to the explicit key identifier mode (0x01) at the Key Index number 1. It is available for any mote whatever their MAC address and even if it is not yet registered.

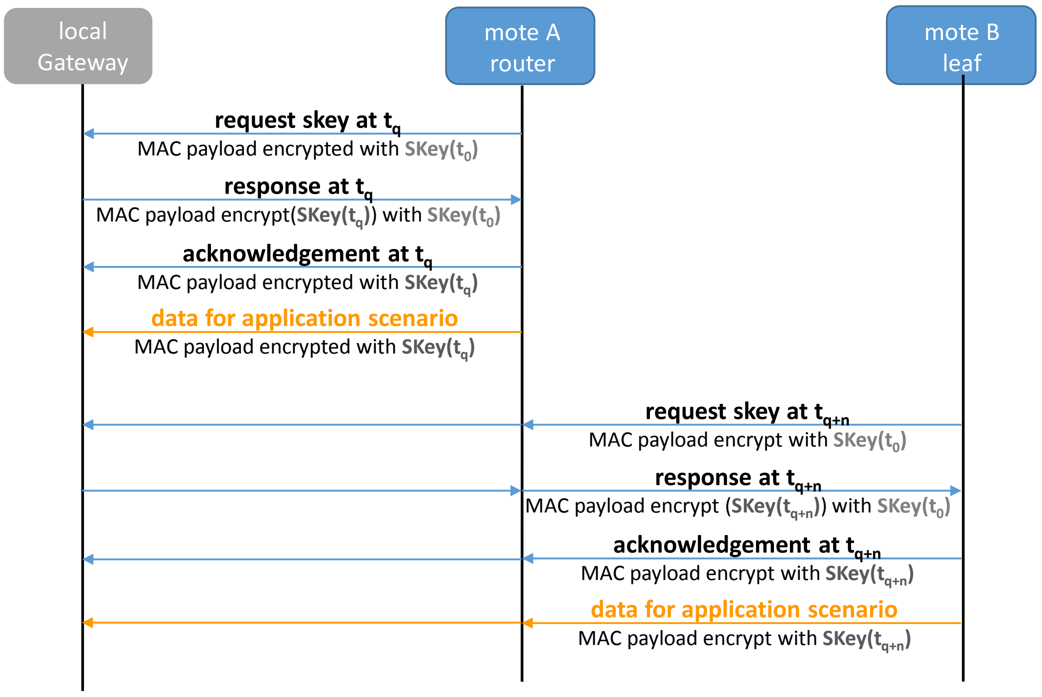

local Gateway
mote A router
mote B leaf
request skey at t_q
MAC payload encrypted with SKey(t_0)
response at t_q
MAC payload encrypt(SKey(t_q)) with SKey(t_0)
acknowledgement at t_q
MAC payload encrypted with SKey(t_q)
data for application scenario
MAC payload encrypted with SKey(t_q)
request skey at t_{q+n}
MAC payload encrypt with SKey(t_0)
response at t_{q+n}
MAC payload encrypt (SKey(t_{q+n})) with SKey(t_0)
acknowledgement at t_{q+n}
MAC payload encrypt with SKey(t_{q+n})
data for application scenario
MAC payload encrypt with SKey(t_{q+n})

Figure 9: Distribution of SKey(t_q) after a join

An extract of the code of the Java API that enables the management of the security key SKey along the time is detailed below.

```
// Outbound (TX) configuration
Wpan.modifyInterface("wpan0")
                .setOutputSecurityLevel(Interface.SecurityLevel.ENC_MIC32)
                .setOutputKeyIdentifier(
                              Wpan.KeyIdentifier.Builder.implicit(0x1234,
"0x4BC1").build()
                )
                .apply();

// Inbound (RX) key management
Wpan.modifyInterface("wpan0")
                .keys()
                .add(new Wpan.KeyItem(
                              EnumSet.allOf(Interface.FrameType.class),
                              EnumSet.noneOf(Interface.CommandFrame.class),
                              Wpan.KeyIdentifier.Builder.index((byte)
0x01).build(),
                              new byte[] {0xca, 0xfe ...}
        ));

// Device management
Wpan.modifyInterface("wpan0")
                .devices()
                //frame counter | panid | short addr | extended addr | security
level exempt | "key mode"
                .add(new     Wpan.Device(0x0,     0x1234,     "0x4BC1     ",
"0xbeefcafecafebeef", false, Wpan.Device.KeyMode.IGNORE));
```

3.1.2 Implementation of the security management in Contiki 3.x

The management of the security in Contiki 3.x is set in the *project-conf.h* file presented below. The file *noncoresec.c* in the Contiki core gets the security values defines in the *project-conf.h* and instantiates the variable in the RAM memory to handle these values. This raises a problem: when the mote switches off because its battery is empty for example, there is no way to connect it to a secure network anymore. The RAM content is lost when the mote switches off and the security variable are set from the *project-conf.h* file during the link.

That is why, we are defining an initial secret key named SKey(t_0) that is stored as a constant in the FLASH memory with the execution code. This key is used by the mote to join the secure network, and even if the mote switches off, SKey(t_0) remains available for a further join.

```
#ifndef PROJECT_CONF_H_
#define PROJECT_CONF_H_

#define CC2538_RF_CONF_CHANNEL 11

#define NETSTACK_CONF_LLSEC    noncoresec_driver
#define NETSTACK_CONF_FRAMER noncoresec_framer

#define LLSEC802154_CONF_SECURITY_LEVEL FRAME802154_SECURITY_LEVEL_ENC_MIC_32

#define NONCORESEC_CONF_KEY {       0x00, 0x01, 0x02, 0x03,
                                    0x04, 0x05, 0x06, 0x07,
                                    0x08, 0x09, 0x0A, 0x0B,
                                    0x0C, 0x0D, 0x0E, 0x0F}

#endif /* PROJECT_CONF_H_ */
```

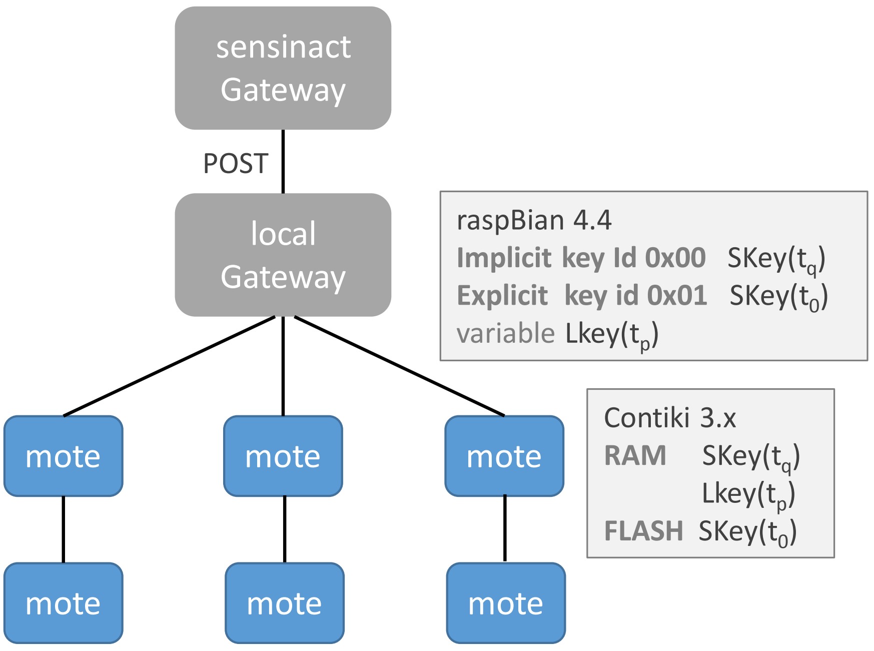


Figure 10: Implementation of the security at the link layer

Figure 10 shows a picture of a possible WSN deployment embedded security keys for the link layer protection.

3.2 Management of the "privacy" key at IEEE 802.15.4 MAC layer

Section 5 in the deliverable D3.2.1 [27] proposes a new scheme to mask the MAC and IPv6 addresses of the messages exchanges hop-by-hop over the air in a 6LoWPAN network. This avoids many attacks on routing protocols enabled via the analysis of the information brings by the source and destination addresses included in the headers of the eavesdropped packets.

Thanks to this counter-measure, the privacy of the things is ensured by a tunnel-like mechanisms within the wireless sensor network (WSN).

The technique of pseudonym generation and management is based on the use of a secret symmetric key, named LKey, shared by the motes inside the 6LoWPAN network. This key can be deduced from contextual characteristics holding in the system, or it can be derived from a random number in the local gateway. Anyway, LKey should be maintained and renewed along the time and is indexed by the time LKey(t_p). Here, it should be noticed that the time where LKey is renewed t_p is independent of the time where t_q where SKey is renewed.

So, LKey(t_p) is registered by the local gateway as a variable and is propagates to all the mote belonging to the WSN encrypted by SKey(t_q). ach mote receiving LKey(t_p) is then enabled to generate source and destination pseudonym according to the dynamic pseudonym generation scheme presented in D3.2.1 and protected by the French patent 16 50610.

This scheme will uses the two keys at the link layer, in one hand to cipher the MAC payload that contains, among other information, the IPv6 addresses used to route a messages from its original source to its final destination; and in the other hand, to mask the MAC addresses with cryptographic pseudonyms in the packet sent between neighbour motes.

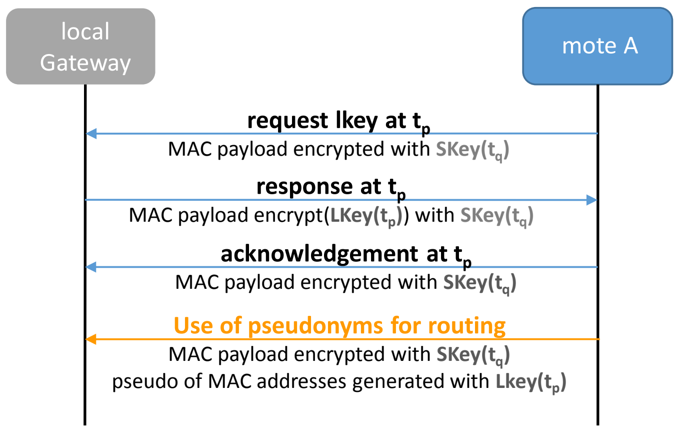

local Gateway
mote A
request lkey at t_p
MAC payload encrypted with SKey(t_q)
response at t_p
MAC payload encrypt(LKey(t_p)) with SKey(t_q)
acknowledgement at t_p
MAC payload encrypted with SKey(t_q)
Use of pseudonyms for routing
MAC payload encrypted with SKey(t_q)
pseudo of MAC addresses generated with Lkey(t_p)

Figure 11: Distribution of LKey(t_p) used to generate the pseudonyms

Section 4 - Authentication

The authentication module verifies if a subject is truly who or what it claims to be. Authentication services are provided by the SocIoTal Identity Management (IdM) system. Besides the conventional password-based and key-based mechanisms, the SocIoTal IdM also supports authentication based on *anonymous credentials*.

4.1 Login password-based authentication

The SocIoTal IdM relies on the Keyrock IdM to perform this kind of authentication. To this aim, the SocIoTal IdM library provides a method that given the username and password authenticates the user against the keyrock and generates an authentication token that can be used afterwards to perform other actions against the Keyrock IdM.

The SocIoTal Keyrock IdM client library provides two main methods for dealing with this kind of authentication.

- The first method is for authenticating a user registered in the Keyrock given the password and username and its domain. This method allows authenticating a registered entity in the IdM KeyRock and generating an authentication token associated to it, using its unique ID, password and domain. If the entity was not registered or the authentication process is wrong, the token is not generated.
- The second method, provided by the Keyrock IdM client, is for validating a given token ID. This functionality is used in SocIoTal by the CapabilityVerifier library (deployed for instance in the Context Manager) to ensure the authentication of a user in a request, that is, to ensure that the user accessing the Context manager, is who he claims to be, in the attached token. This authentication token verification is also used by the Capability Manager to authenticate users requesting capability tokens. Further information about this authentication mechanism can be found in Section 5.

4.2 PKI-based authentication

SocIoTal framework allows users to authenticate using their digital certificates. It requires a Public Key Infrastructure (PKI) where the user's certificates are signed by the Certificate Authority (CA) and validated on the server side. The IdM server performs client authentication based on asymmetric-key cryptographic. This is done actually by the IdM web server that authenticates the client that must possess the private key associated with its public X509 certificate (the server trusts the CA). The server uses the SSL stack to require a valid certificate chain from the client before accepting a connection. This kind of authentication can be used in SocIoTal, for instance, in the IdM client app for Android when accessing the Issuer Server. The authentication based on certificates is also performed during the transaction stage between two entities, where the target entity is endowed with the capability token verifier module. The IdM client app is able to verify that the capability token signed by the Capability Server is issued to a subject that matches the subject of the user's certificate being used during the transaction.

4.3 Anonymous credentials based authentication

This kind of authentication is done by the SocIoTal privacy-preserving IdM, which offers a claim-based solution for authentication. The SocIoTal IdM relies on an anonymous credential system, namely Idemix from IBM [4], to ensure a privacy and minimal disclosure of personal information. The IdM provides means for users to make use of the Idemix credentials for authentication. This kind of authentication based on anonymous credentials was explained in detail in deliverable D2.1 [26]. Implementation and API details can be found in the SocIoTal Wiki. Further information about this authentication mechanism can be found in Section 5.

Section 5 - 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 but adapted to IoT scenarios. In order to address the SocIoTal uses cases, where mobile smartphones are usually employed, a part of the IdM is deployed in the end-users' smartphones.

Having part of the IdM deployed in end-users smartphones allows the end users 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 peer 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 provides the means to select the partial identity according to the actual context, and run the presentation process, which aims 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 to existing open solutions, which already provide those functionalities.

The SocIoTal IdM has been integrated within the FI-WARE IdM, i.e., the Keyrock IdM. The SocIoTal IdM Issuer, which is in charge of generating the Idemix credentials, is able to communicate with the Keyrock to generate the credentials based on the attributes stored in the Keyrock for the user requesting the credential. To this aim, the SocIoTal IdM provides a Java library to interact with Keyrock by means of the SCIM standard [7]. This library provides methods such as add, remove or update users.

Some other components in SocIoTal, beyond the IdM Issuer server, such as the Web User Environment or the community's enabler makes use of the SocIoTal IdM library to manage users' identities.

5.1 SocIoTal IdM components

This section overviews the five main IdM components that have been designed and implemented in the scope of SocIoTal. Implementations details are omitted since they are described in the SocIoTal Wiki.

1. SocIoTal IdM Client: This is client application that allows obtaining Idemix credentials from the Issuer Server. It also allows interact with the Verifier server which can validate the partial identity derived from the credential.

2. SocIoTal-Issuer-Server: This is a web application which allows generating Idemix credentials for clients. The client must be authenticated against the Issuer using a valid certificate. The Issuer also supports the verification functionality.

3. SocIoTal-Verifier-Server: This is a web application, which is able to validate partial identities presented by the client application.

4. SocIoTal-IdM-Enabled-Capability Manager: This is a web application that allows users to obtain capability tokens using their partial identities. In other words, it allows authenticating and demonstrating their attributes by means of Idemix proofs of having a valid credential issued by the Issuer.

5. SocIoTal IdM KeyRock Client: This is a library that provides a basic API for identity management by implementing a client to interact with the FI-WARE KeyRock server.

The authentication process is needed each time the user (or a trusted entity acting on behalf for example the Web User Environment defined in D4.3 [28]) requests a capability token for the user. The SocIoTal IdM is in charge of the authentication process using either a claim-based approach (i.e., based on Idemix) or an On-Line based approach (i.e., relying on Keyrock IdM). Figure 12 shows the main process for both kinds of authentication. In the following, we discuss this process in detail:

Capability Token request using claim-based authentication
The IdM allows Android-based Smart Devices to interact with an Issuer to obtain the credentials and used them as a mechanism to interact with other entities in a privacy-preserving fashion based on a claim-based approach.

Firstly, the user requests an Idemix credential to the IdM Issuer Server. The Issuer in this case authenticates the user with login-password against the Keyrock IdM. It is assumed that the user was previously registered in Keyrock through the Web User Environment. Once authenticated, the Issuer obtains the user profile including its attributes, and checks that the Idemix credential to be generated matches the attributes values in the Keyrock IdM. The Issuance process between the Issuer and Subject is already defined in deliverable D2.1 [26], so the detail is omitted here.

Then, the user requests a capability token and he wants to authenticate and demonstrate its attributes following a claim-based approach. To this aim, the obtained Idemix credential can be used by the user to be authenticated against a target device (or central entity) running the Idemix proving protocol (which demonstrates the derived partial identity and its associated attributes). The IdM proving protocol is defined in deliverable D2.1 [26].

The Capability Manager in this case makes an authorization decision based on user attributes presented in the partial identity, without the on-line intervention of the KeyRock IdM for obtaining the user attributes, since the attributes are cryptographically demonstrated in a credential proof.

Capability Token request using On-Line authentication
For those scenarios where claim-based authentication is not needed, SocIoTal provides an On-Line authentication mechanism based on login-password, which rely on the FI-WARE Keyrock IdM service. In this case, the user wants to obtain a capability token, so he is firstly authenticated through the Web User Environment, which in turn, uses the Keyrock client API to authenticate the user in the Keyrock IdM service. In this step, the user obtains a tokenId that can be used as means for authenticating against the Capability Manager entity in charge of generating the capability tokens.

Upon a request, the Capability Manager validates in an On-line fashion the attached Keyrock authentication token against the Keyrock service, using the SocIoTal Keyrock client API. Then, once authenticated, the Capability Manager obtains the user attributes required to check the

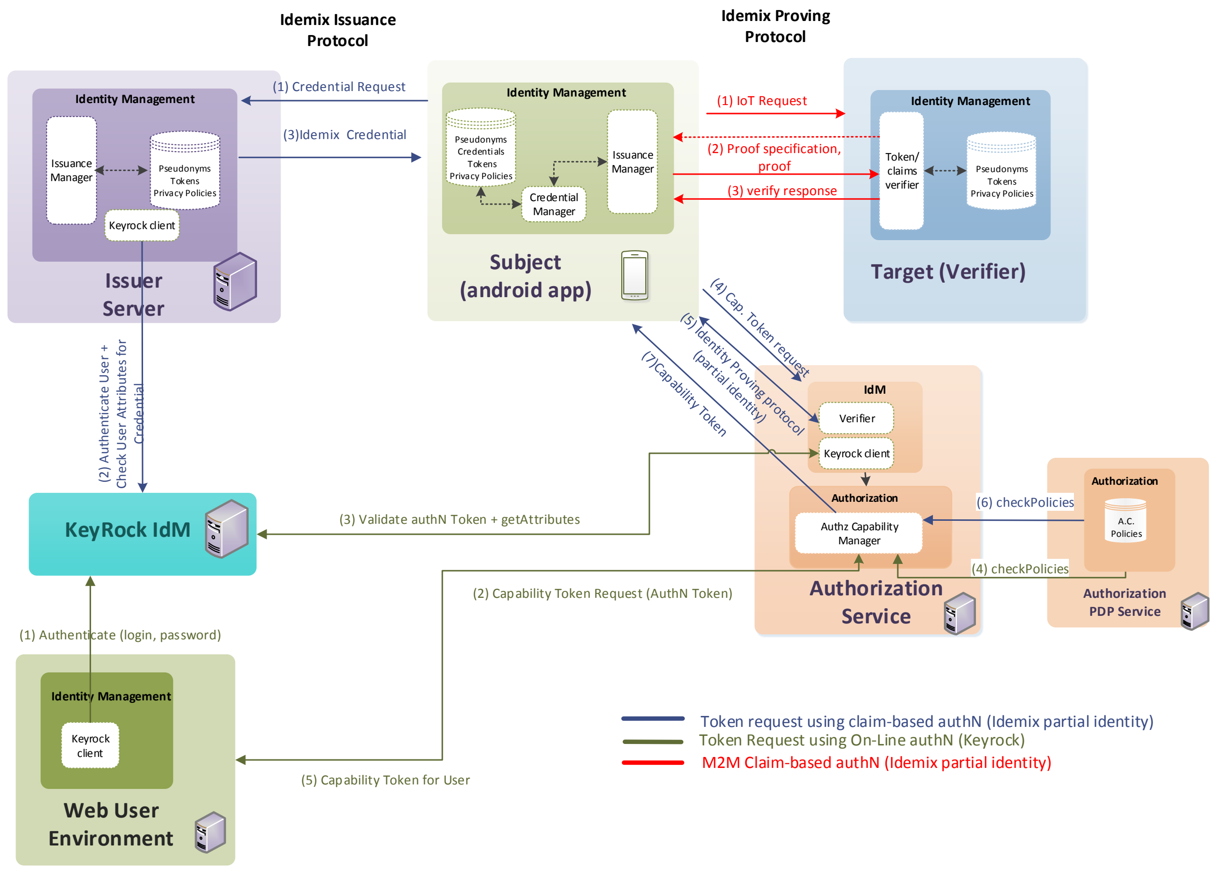

Idemix Issuance Protocol
Idemix Proving Protocol
Identity Management
Issuance Manager
Pseudonyms Tokens Privacy Policies
Keyrock client
Issuer Server
(1) Credential Request
(3)Idemix Credential
Pseudonyms Credentials Tokens Privacy Policies
Issuance Manager
Credential Manager
Subject (android app)
(1) IoT Request
(2) Proof specification, proof
(3) verify response
Token/ claims verifier
Target (Verifier)
(2) Authenticate User + Check User Attributes for Credential
KeyRock IdM
(3) Validate authN Token + getAttributes
(4) Cap. Token request
(5) Identity Proving protocol (partial identity)
(7)Capability Token
IdM
Verifier
Authorization
Authz Capability Manager
(6) checkPolicies
(4) checkPolicies
A.C. Policies
Authorization PDP Service
Authorization Service
(2) Capability Token Request (AuthN Token)
(1) Authenticate (login, password)
Web User Environment
(5) Capability Token for User
Token request using claim-based authN (Idemix partial identity)
Token Request using On-Line authN (Keyrock)
M2M Claim-based authN (Idemix partial identity)

Figure 12: Interactions between SocIoTal IdM components

authorization policies by the Policy Decision Point (PDP) and make a decision about whether the user can obtain the requested capability token or not.

M2M claim-based authentication
The communication between a Subject and the Verifier is done basically by performing the presentation process based on Idemix. It includes the interactions between the Subject device, which wants to prove the possession of certain attributes in its partial identity (i.e., in this issued credential), and the Verifier, which wants to authenticate the Subject's SmartObject.

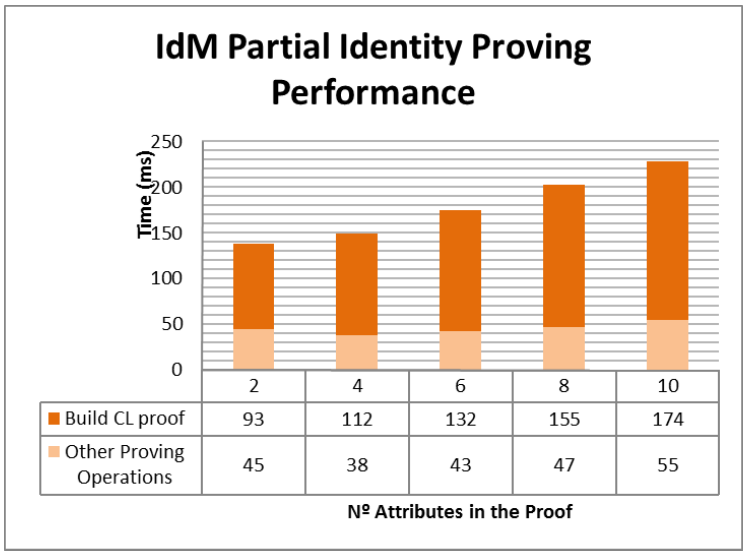
5.2 IdM performance results

An evaluation of the performance of the SocIoTal IdM in generating and validating partial identities is presented here. The evaluation results include the times required to deal with different operations, but the most important results are those related to the Idemix cryptographic proving protocol. The Idemix protocol is carried out by the subject device to prove that it is in possession of a valid credential, while anonymity is preserved. In the testbed, the Idemix proof that is sent to the target device, contains the pseudonym (the same included in the capability token), as well as an incremental amount of unrevealed attributes (different amount in different tests). The implementation relies on the Idemix Java library. It is worth mentioning that the testbed uses Android SDK 1.7 and the RSA secret key length is limited to 1024 bits (i.e., 80-bit security level), in order to compare the results with previous alternatives.

Figure 13 and Figure 14 sum up the performance times obtained in the Identity Management to validate the Idemix testbed. It shows the time required by the subject to build the proof (i.e., build the partial identity). The total build proof operations made in the subject side are split into 2 different series, in order to be able to show the times required to build the Camenisch-Lysyanskaya (CL) proof, which are the heaviest tasks in the whole proving protocol. The series labelled “Other proving operations” encompasses, among others operations, the time required to

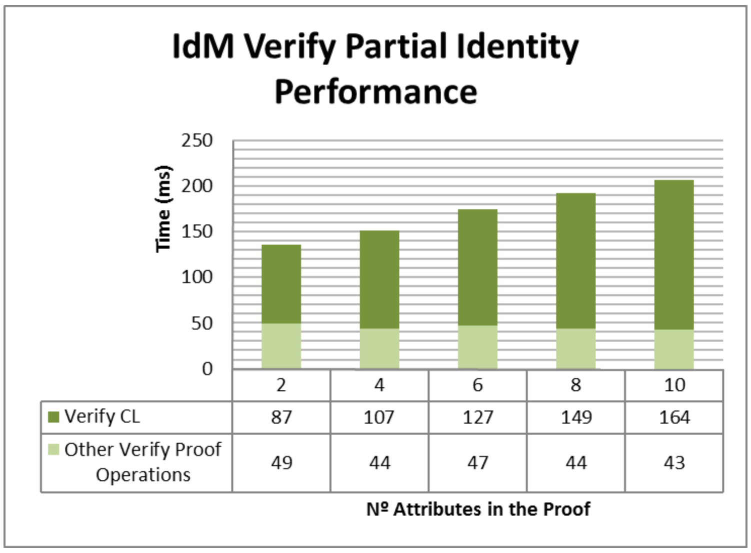
- load the credential previously obtained,
- initiate the verification process to obtain the nonce, parse and validate the proof specification, as well as
- generate the challenge.

The X-axis in both charts represents the amount of attributes used in the proof. The CL proof generation requires more computation time, since the proof contains a higher amount of attributes. The time required to perform other operations are usually steady across the different tests. Notice that the results does not include the network delay.



	2	4	6	8	10
🟧 Build CL proof	93	112	132	155	174
▨ Other Proving Operations	45	38	43	47	55

Figure 13: IdM Partial Identity Proving performance



	2	4	6	8	10
🟩 Verify CL	87	107	127	149	164
▨ Other Verify Proof Operations	49	44	47	44	43

Figure 14: IdM Verify Partial Identity Performance

On the other hand, the Verify Proof operation, which is carried out by the target device, is shown in Figure 14. The total time required during the proof verify task shows the time required to verify the CL signature. As it was predictable, the time increases as the partial identity includes more attributes, that is, the amount of attributes in the proof is higher. As can be seen, according to both charts, the verification operation compared to the build proof operation requires a slightly less computation time.

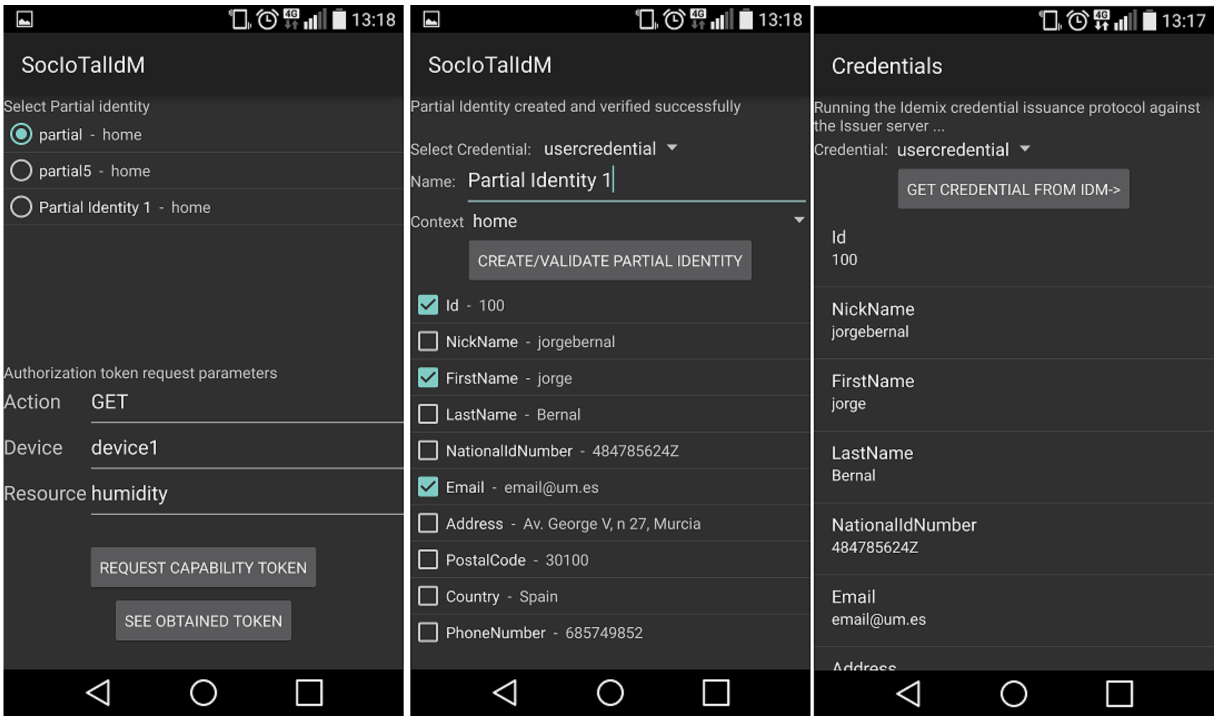
5.3 Implementation and Integration

The SocIoTal IdM is composed of a set of implemented libraries and components explained below:

Identity Management app

It is an Android application that allows obtaining Idemix credentials from the Issuer Server. It also allows interact with the Verifier server which can validate the partial identity derived from the credential.

The following figure shows some screen captures of the SocIoTal Identity Management App. The first one (starting from the left) shows the main window used for requesting a capability token with minimal disclosure using a selected partial identity to demonstrate the attributes. The second one is the partial identity layout, in which users can create partial identities selecting the attributes from the ones that he has in his credential. The third screen shows the credential layout where users can obtain an Idemix Credential from the SocIoTal IdM Issuer.


13:18
SocIoTalIdM
Select Partial identity
[✓ partial - home]
partial5 - home
Partial Identity 1 - home
Authorization token request parameters
Action GET
Device device1
Resource humidity

13:18
SocIoTalIdM
Partial Identity created and verified successfully
Select Credential: usercredential
Name: Partial Identity 1
Context home

☑ Id - 100
☐ NickName - jorgebernal
☑ FirstName - jorge
☐ LastName - Bernal
☐ NationalIdNumber - 484785624Z
☑ Email - email@um.es
☐ Address - Av. George V, n 27, Murcia
☐ PostalCode - 30100
☐ Country - Spain
☐ PhoneNumber - 685749852

13:17
Credentials
Running the Idemix credential issuance protocol against the Issuer server ...
Credential: usercredential

Id
100
NickName
jorgebernal
FirstName
jorge
LastName
Bernal
NationalIdNumber
484785624Z
Email
email@um.es
Address

Issuer Server

This is a web application implemented with Java servlets and XML-RPC which allows generating Idemix credentials for clients. Communications are done by HTTPS. The client must be authenticated against the Issuer using a valid certificate. The Issuer also supports the verification functionality.

Verifier Server

This is a web application, also implemented with Java servlets and XML-RPC, which is able to validate partial identities presented by the client application.

IdM-Enabled Capability Manager

This is a web application that allows users to obtain capability tokens using their partial identities. In other words, it allows authenticating and demonstrating their attributes by means of Idemix proofs of having a valid credential issued by the Issuer.

IdM KeyRock Client

This is a Java library that provides a basic API for identity management by implementing a client to interact with the FI-WARE KeyRock server. To carry out such communication, the SCIM 2.0 and Identity API v3 interfaces provided by this IdM are used. For further implementation details, readers are referred to the SocIoTal wiki [24].

Section 6 - Authorization

The SocIoTal access control system is designed 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 authorization credentials in the form of capability tokens that are described in D2.2 [31]. Then, such tokens are used by smart objects to get access to services being provided by other IoT entities.

The design and implementation of this system have been realized through the convergence of technologies, and adapted to be integrated with other SocIoTal components. On the one hand, it has been integrated with the FI-WARE IdM, so authorization decisions are based on the identity attributes that are stored in the Keyrock IdM instance. On the other hand, part of this system is integrated in the SocIoTal Context Manager, which is also responsible for evaluating capability tokens to allow or not a specific NGSI action over a certain entity. Furthermore, it has been adapted to SocIoTal scenarios enabling the process to be carried out directly (by using user certificates) and through the Web User Environment without the need of such certificates.

The SocIoTal Authorization scenario consists of the following main entities:

1. **Capability Client**: performs requests to the Capability Manager to obtain capability tokens, which are used to perform actions over entities that are registered with the Context Manager.
2. **Capability Verifier**:,a server receiving access requests from Capability Clients. Such access requests contain a capability token, which is evaluated by the Capability Verifier in order to deny or grant the requesting action. This entity makes use of the Capability Evaluator functionality, which is intended to validate capability tokens.
3. **Capability Manager**: a server accepting requests for capability tokens generation. Additionally, this entity acts as a client requesting authorization decisions to the Policy Decision Point.
4. **Policy Decision Point (PDP)**:It is a server that accepts XACML requests to make authorization decisions. The PDP is contacted by the Capability Manager before generating a capability token for the Capability Client.
5. **Policy Administration Point (PAP)**: It is a web application responsible for managing the access control policies. It provides the functionality so users can define XACML policies in a user-friendly way. The PAP has a GUI to facilitate the generation of policies.

- Figure 15 shows a scenario for the case where users possess X.509 certificates. Firstly, a user (e.g. through the Mobile User Environment) acting as a Capability Client, tries to get a capability token from the Capability Manager with her certificate, by specifying the NGSI action and the entity for which the token must be generated. Then, the Capability Manager authenticates the user and it obtains her ID from the user’s certificate. This ID is used to get the identity attributes that are associated to such user and stored in the Keyrock IdM. Once the Capability Manager obtains these attributes, it sends a XACML request to the PDP in order to ascertain that the user (based on her identity attributes) is authorized to perform the NGSI action over the specified entity. If so, the Capability Manager generates a capability token associating the privilege to the public key of the user’s certificates. Finally, this token is delivered to the user, who can make use of it to perform an action over an entity that is in the Context Manager.

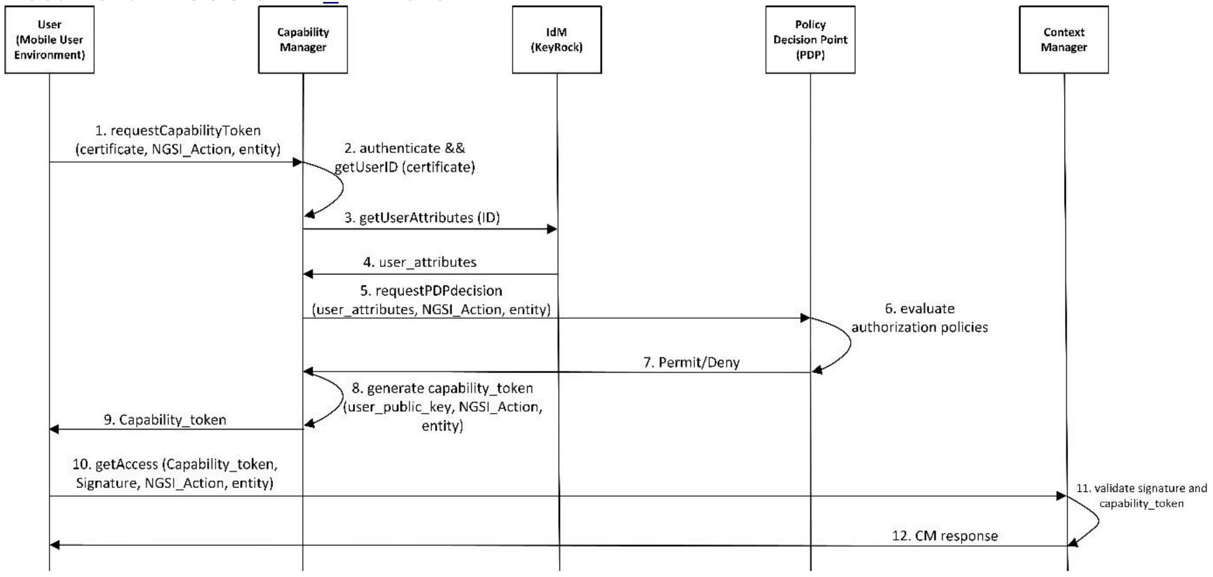

User (Mobile User Environment)
Capability Manager
IdM (KeyRock)
Policy Decision Point (PDP)
Context Manager
1. requestCapabilityToken (certificate, NGSI_Action, entity)
2. authenticate && getUserID (certificate)
3. getUserAttributes (ID)
4. user_attributes
5. requestPDPdecision (user_attributes, NGSI_Action, entity)
6. evaluate authorization policies
7. Permit/Deny
8. generate capability_token (user_public_key, NGSI_Action, entity)
9. Capability_token
10. getAccess (Capability_token, Signature, NGSI_Action, entity)
11. validate signature and capability_token
12. CM response

Figure 15: SocIoTal authorization scenario through the use of certificates

- Figure 16 shows a scenario for the case where users do not have a certificate and they use the Web User Environment (WUE) to apply for capability tokens and to access the Context Manager. In this case, a user accesses the WUE by entering her login password. These data are used by the WUE to get the user ID, which is employed to obtain an authentication token from the Keyrock IdM. Then, a user tries to get a capability token through the WUE (acting as a Capability Client) by indicating a specific action and entity. This information, together with the user ID and the authentication token ID, is used to request the token from the Capability Manager. Then, the Capability Manager gets the authentication token from the Keyrock IdM and validates the token. Furthermore, it uses the user ID to recover the identity attributes associated with the user from the Keyrock IdM. As in the previous case, the Capability Manager obtains these attributes, and sends an XACML request to the PDP to check if the user (based on her identity attributes) is authorized to perform the NGSI action over the entity being specified by the user. If so, the Capability Manager generates a capability token associating the privilege to the user ID and the public key of the WUE. An example of this token is shown in Table 1 (for a more detailed description, see D2.2 [31]).

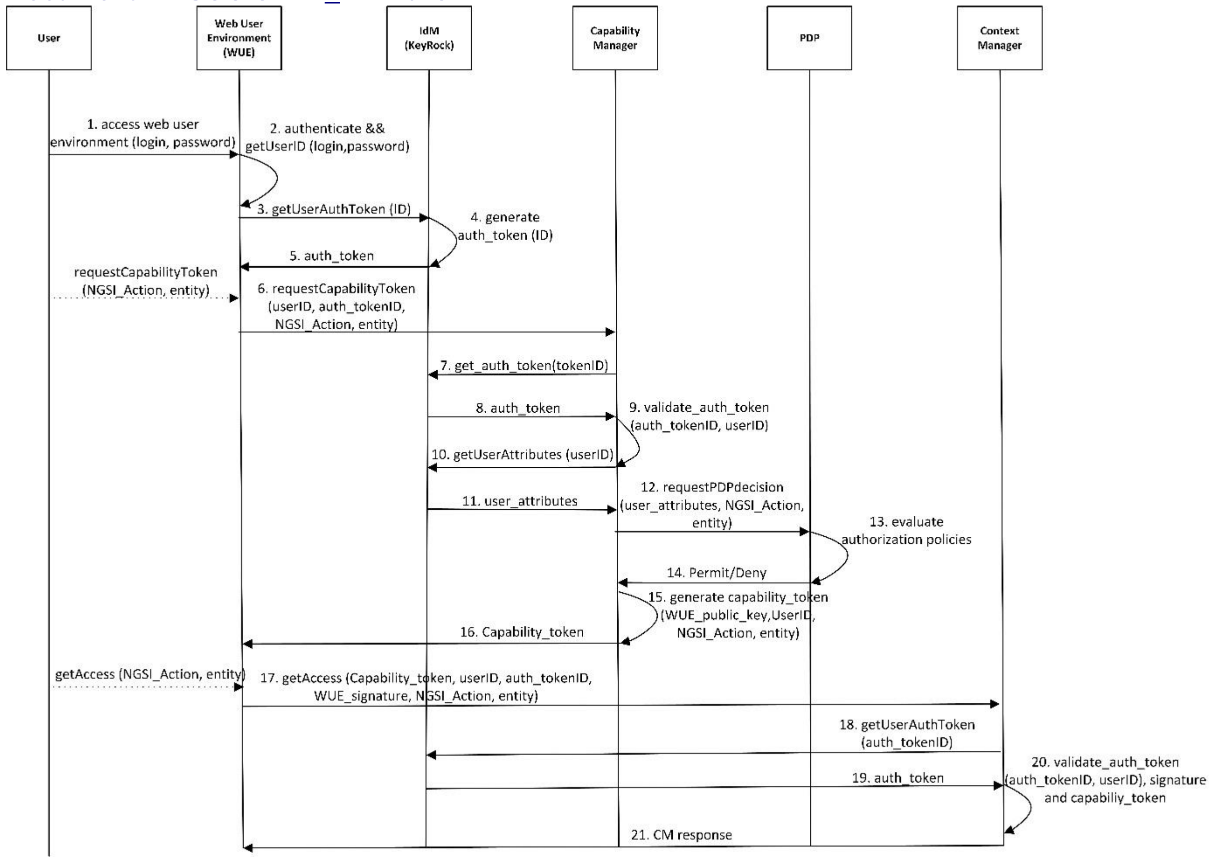

User
Web User Environment (WUE)
IdM (KeyRock)
Capability Manager
PDP
Context Manager
1. access web user environment (login, password)
2. authenticate && getUserID (login,password)
3. getUserAuthToken (ID)
4. generate auth_token (ID)
5. auth_token
requestCapabilityToken (NGSI_Action, entity)
6. requestCapabilityToken (userID, auth_tokenID, NGSI_Action, entity)
7. get_auth_token(tokenID)
8. auth_token
9. validate_auth_token (auth_tokenID, userID)
10. getUserAttributes (userID)
11. user_attributes
12. requestPDPdecision (user_attributes, NGSI_Action, entity)
13. evaluate authorization policies
14. Permit/Deny
15. generate capability_token (WUE_public_key,UserID, NGSI_Action, entity)
16. Capability_token
getAccess (NGSI_Action, entity)
17. getAccess (Capability_token, userID, auth_tokenID, WUE_signature, NGSI_Action, entity)
18. getUserAuthToken (auth_tokenID)
19. auth_token
20. validate_auth_token (auth_tokenID, userID), signature and capabiliy_token
21. CM response

Figure 16: SocIoTal authorization scenario through the Web User Environment

```
{
  "id": "n73kkkihqq218aj20r99f3qdkm",
  "ii": 1455796439,
  "is": "capabilitymanager@um.es",
  "su": "d96aa41f9293bb95843c8632f059e561eaddb00beb1054c636ee3fbc1c0ce64a",
  "del": "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqu...",
  "de": "SocIoTal:IoTWeek:WeatherStation:Dev_001",
  "si": "hLSvWg8dmGw84HYL78tOgpmNkCJuzkPoIMHjd8jf3djdslm...",
  "ar": [
    {
      "ac": "queryContext",
      "re": "*"
    }
  ],
  "nb": 1455796439,
  "na": 1455807439
}
```

Table 1: Capability Token example for interactions through the WUE

After this process is complete, a user tries to get access to the Context Manager by using the token previously obtained. Therefore, the WUE, acting on behalf of the user, gets the token associated with the action and entity being specified by her, and it generates a request to the Context Manager by using such token, as well as the user ID and the authentication token ID, which were obtained during the previous process. After receiving the request, the Context Manager tries to get the authentication token associated with the user to verify if she is actually authenticated. If so, it validates the capability token and sends the response to the WUE, which can display the message to the requesting user.

6.1 Trust-Based Authorization (TAcIoT)

The Trust-aware Access Control for IoT (TAcIoT) implements the SocIoTal Trust Model defined in deliverable D2.3, in order to enable a secure information exchange between trustworthy entities. This mechanism has been considered to be deployed on IoT scenarios where smart objects (e.g. smartphones, sensors, actuators, etc.) can maintain social relationships composing different kinds of bubbles (i.e., Personal, Family, Office or Community). According to Figure 17, each bubble is made up of a set of smart objects, along with 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 has a Trust-aware Access Control for IoT (TAcIoT) Trust Manager, which is 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), this entity is assumed to be deployed in a more powerful network component.

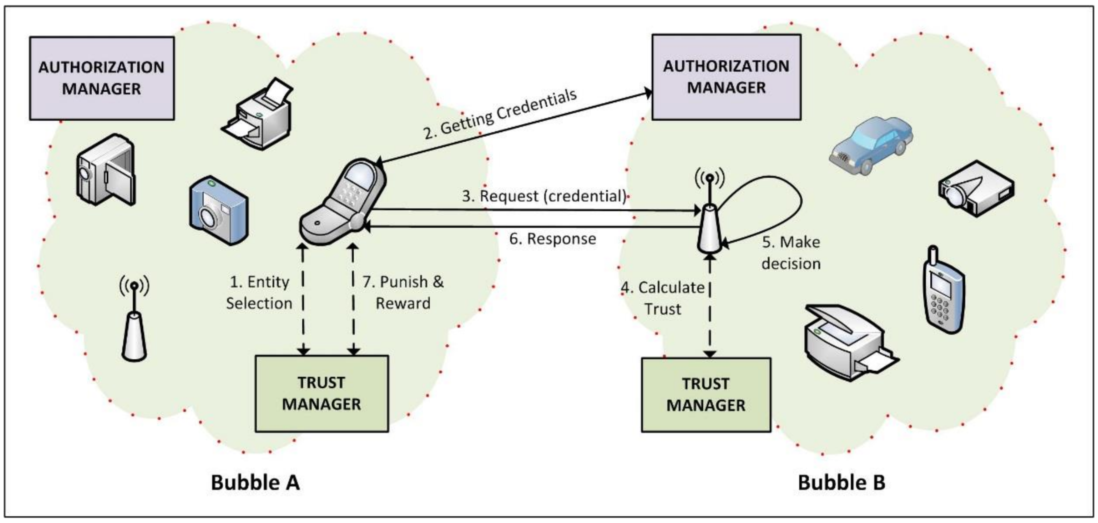


Figure 17: Sample scenario for TAcIoT

In the proposed trust-aware access control system, an **intra-bubble** communication happens when a smart object attempts to access another smart object that is part of the same bubble. Figure 17 shows the interactions at high level in the case of an **inter-bubble** communication between smart objects from different bubbles. Under this scenario, the purpose of the TacIoT Trust Manager is twofold. 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. On the other hand, it is employed by the **target** smart object in order to get the trust value associated with 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 main involved components is provided:

- **Smart object.** It is a device (e.g. a smartphone, printer, camera, sensor, etc.) that can act both as a CoAP client and a CoAP server offering services (e.g. temperature, location, etc.) in an IoT environment.
- **Trust Manager.** It is the component implementing the proposed trust model. In the case of a smart object with tight resource constraints (i.e., class 0 or class 1 device), the Trust Manager is a separate network element. In the case of more powerful smart objects (at least class 2 devices), the Trust Manager is a part of the devices.

- **Authorization Manager**. It is responsible for generating and sending authorization tokens to smart objects. Additionally, it is composed of two subcomponents; the Policy Decision Point (PDP), which is in charge of making authorization decisions based on a XACML engine, and the Token Manager, which generates authorization credentials according to the authorization decisions.

6.2 TAcIoT implementation and testbed features

TAcIoT implements the trust model defined in deliverable D2.3 [25]. The trust model together with the authorization solution has been implemented as part of the Authorization and Trust Manager components of the SocIoTal security framework. The TAcIoT Trust Manager implementation is able to quantify trust property expectations of devices based on historical evidences, calculate trust dimensions' values from the expectations and compute trustworthiness based on such trust dimensions. The Trust Manager has been implemented in Andorid SDK and tested in Android Platform 2.3.3 (API level 10) to cover a wide variety of Android devices with moderate hardware capabilities. Nonetheless, it also works properly in actual platforms like Android 4.4. It features an interface to get trust values about a device or set of devices as well as an interface to feed the Trust Manager with new evidences about another device (reward-punish operation). It uses CoAP as the application layer protocol to enable constrained devices to get the trust values and provide new evidences. To deal with fuzzy trust quantification, the TacIoT Trust Manager implementation relies on the jfuzzylite library [32], which is a lightweight and open-source fuzzy logic control library for Android.

The proposed trust-aware access control mechanism was validated on a real testbed, in which the main components of TACIoT were instantiated. According to the mechanism, three main entities are required in order to carry out the main functionality of the proposed approach. First, the TacIoT Trust Manager was deployed in a smartphone with ARM Cortex A8 Processor 1Ghz and 512 MB RAM. Furthermore, the Authorization Manager was developed on a common computer with an Intel Core i5 processor with 2.27 GHz and 4GB RAM. Moreover, given the heterogeneous nature of IoT scenarios, the functionality of the smart objects was implemented in two different hardware components. On the one hand, two resource-constrained smart objects were implemented on a JN5148 mote equipped with Contiki OS. It provides a low power 32-bit load and store RISC with a programmable CPU speed which can be set to 4, 8, 16 or 32 MHz. Furthermore, it has a unified memory architecture with 128 kbytes of ROM, 128 kbytes of RAM and a 32-byte One Time Programmable (OTP) eFuse memory. On the other hand, two more powerful smart objects were considered and developed on the same hardware for the TacIoT Trust Manager.

6.3 TACIoT evaluation results

In order to demonstrate the feasibility of TACIoT, we executed different tests of the main stages of the scenario proposed. As mentioned above, these tests were performed considering two scenarios according to the hardware features of the smart objects. Figure 18 shows the performance times for each of the main tasks required at each of the four main stages in constrained devices. Similarly, Figure 19 shows the achieved times for the non-constrained devices. For a better understanding of the graphs, series are grouped in a different bar for each stage and arranged in a top-down order, following the order indicated in the legend of the graphs. Notice that the vertical axis has a different time scale in both charts since the non-constrained devices require more time to accomplish the different operations.

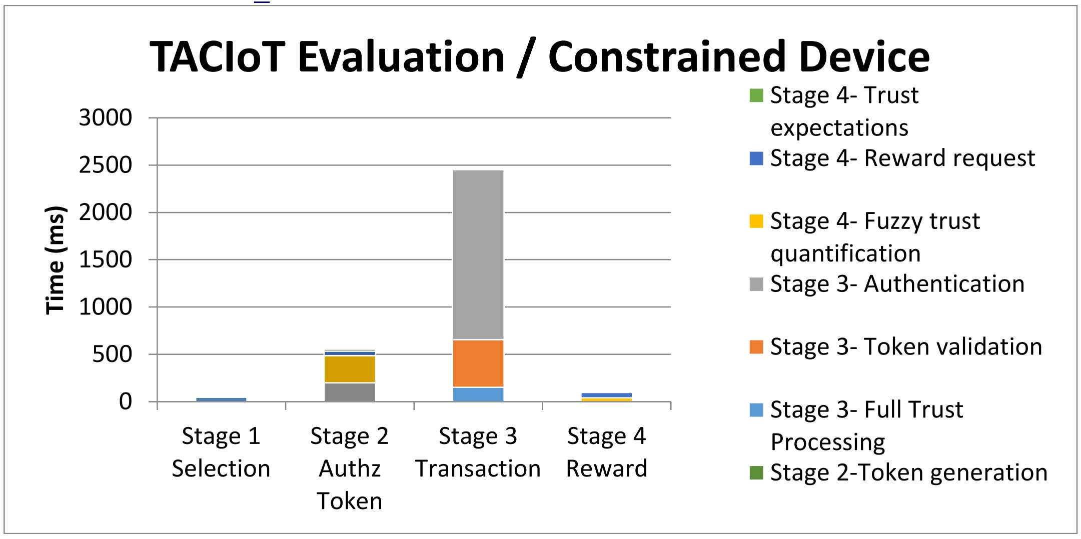


Figure 18: TacIoT Evaluation Constrained device

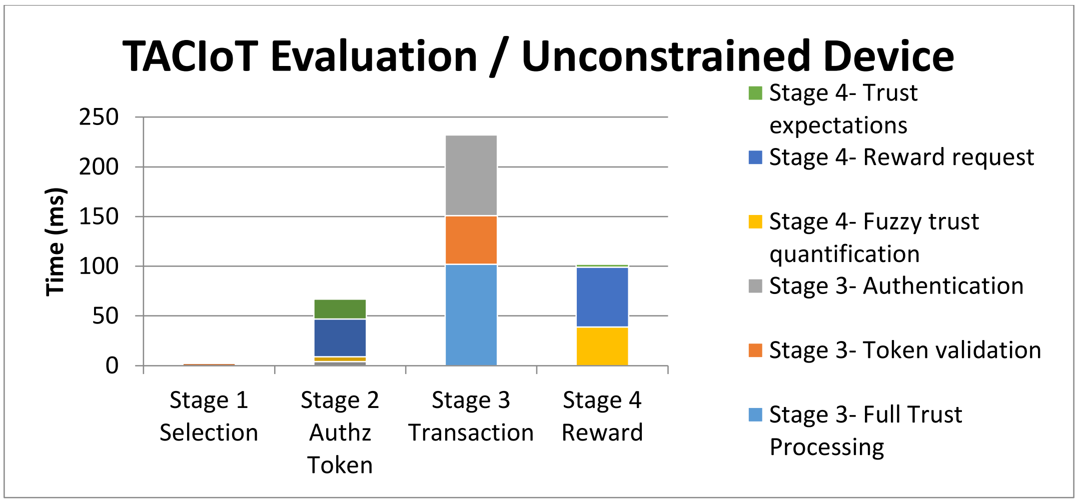


Figure 19: TacIoT Evaluation – Unconstrained device

During the Stage 1 of TACIoT, the following tasks are required:

- Trust request: This includes the time required for CoAP messages processing. This delay is only necessary in constrained devices in which the TacIoT Trust Manager cannot be deployed in the same device.
- Trust query processing: This includes the delay that is required by the TacIoT Trust Manager to get a trust value about a device. The TacIoT Trust Manager computes trust only after adding new evidences which is not the case in this operation.

As can be seen from the chart in Figure 18, Stage 1 is the fastest compared to the other three since it is not necessary to compute trust and the TacIoT Trust Manager just returns the actual trust value of the requested device.

During Stage 2, the requester smart object obtains an authorization credential to access a service on the target smart object. It includes:

- Sign message: The time required for an authenticated CoAP message exchange between the smart object and the Token Manager.

- Validate signature. To validate the signature of the response. This operation depends strongly on the kind of device validating the signature, since in the constrained device the Elliptic Curve Digital Signature (ECDSA) takes around 288ms whereas in Android with Android SDK, it takes around 5 ms.
- Authz request: The time required to process the CoAP messages.
- Token generation: It includes the token generation in the Token Manager. It should be pointed out that the PDP time is omitted since it is strongly dependent on the policy engine, the use case, and the specific representation of XACML policies.

The transaction between both smart objects is carried out in Stage 3, including:

- Authentication: It refers to the time for ECDSA operations as well as the delay required to generate a session key, in order to protect the CoAP request in which the token is attached in the initial stage. Again this time is very small in unconstrained devices.
- Trust processing: The time to obtain a trust value from the TacIoT Trust Manager about a device, attaching in the request new evidences with information of the current transaction. It includes the time to infer a new trust value based on the new attached evidences. In the case of a constrained smart object without TacIoT Trust Manager installed in it, it also includes the time needed to perform the CoAP request to the TacIoT Trust Manager.
- Token validation: It includes the delay to evaluate the authorization credential and the time to validate the Token Manager's signature. Again this time differs significantly for both kind of devices.

As can be seen in the charts of Figure 18, Stage 3 is the heaviest for both constrained and unconstrained devices. This is due to the fact that this stage requires more cryptographic operations, which are more expensive compared to trust quantification.

Finally, during Stage 4, the device provides information about the transaction just carried out, in order for the TacIoT Trust Manager to be kept up-to-date on the device’s behavior. Stage 4 can be split in three main operations:

- Reward Request: This time represents the delay required to process CoAP request including processing the evidences coming in the request.
- Trust Expectations: This time represents the time required to quantify expectation values for each trust property defined in deliverable D3.3 [21] based on historical evidences and the new ones, as well as generate the current four dimension values.
- Fuzzy Trust quantification: This operation is the most intensive during trust computation and refers to the time needed to infer the result by the fuzzy control system.

As can be seen in Figure 19, the time required to accomplish Stage 4 is equivalent to carry out the trust processing task of Stage 3.

It is worth analyzing the behavior of the TacIoT Trust Manager when it has to handle different amount of evidences attached in a reward-punish request. As can be seen in Figure 20, it takes barely the same time to quantify trust after performing the reward operation, regardless of whether the request is attaching one evidence or the full set of evidences (the evidences of the 19 trust properties identified in deliverable D3.3). The small difference lies in the time to decode a request with more evidences attached.

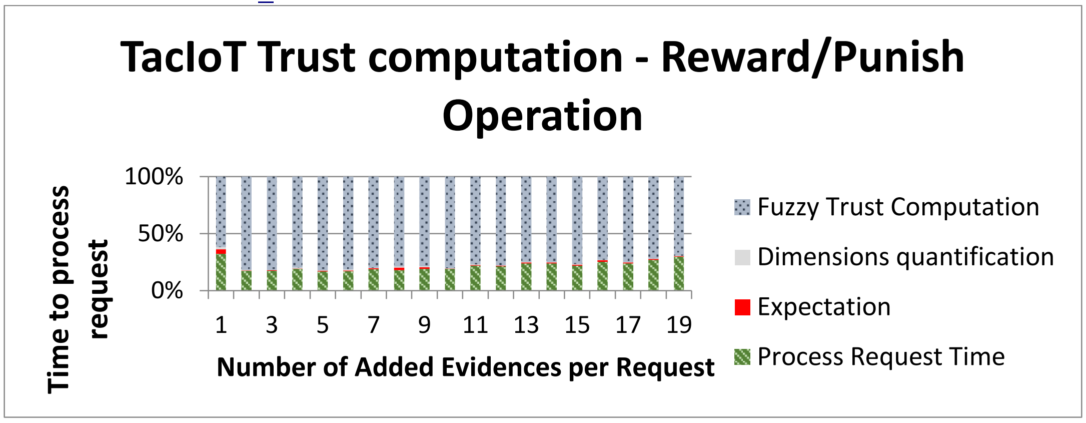


Figure 20: TacIoT Trust Computation - Reward/punish Operatoin

In order to analyze the TacIoT Trust Manager's performance, we measured the time that is required to compute trustworthiness when it has to deal with different amount of devices. To see the behavior trend, we considered up to 100 devices. The test randomly generates evidences for each of the 19 trust property values and queries the TacIoT Trust Manager attaching all of them each time. By default, for each device the TacIoT Trust Manager is configured to hold up to 10 historical evidences per trust property i.e. it holds up to 190 evidences per device, then it dismisses the oldest ones.

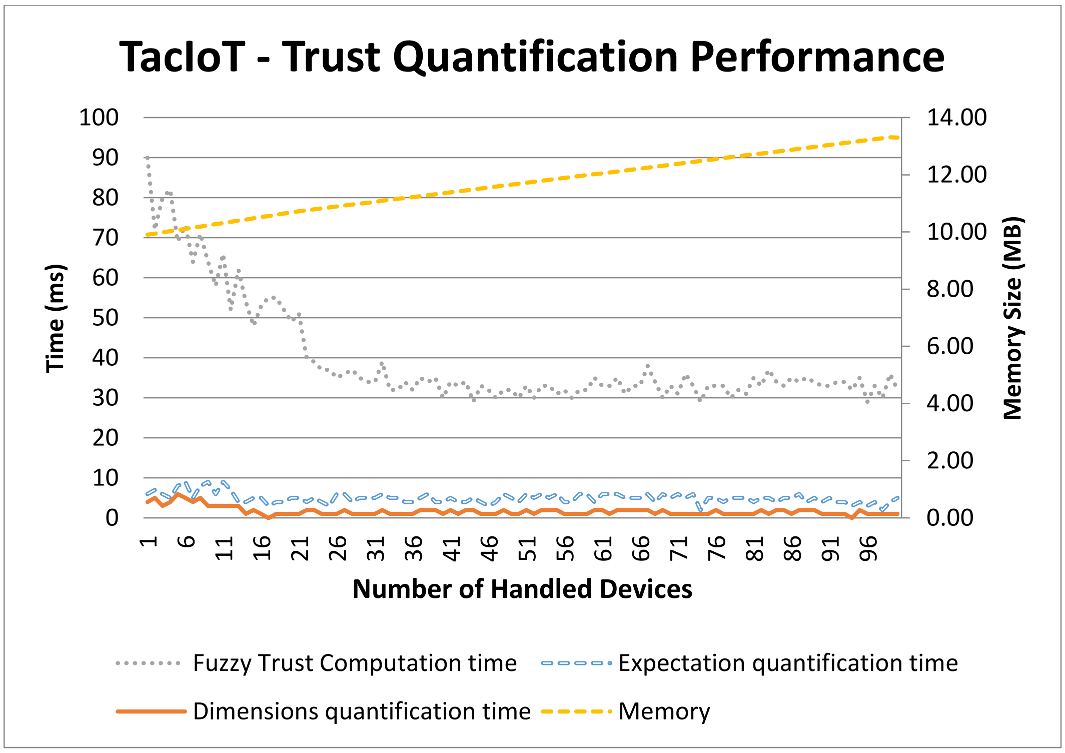


Figure 21: TacIoT Trust Quantification performance

Figure 21 shows the trust quantification performance. As it was predictable, as the amount of handled devices increases, the TacIoT Trust Manager requires more memory to handle all the historical evidences. Nonetheless, the memory consumption follows a moderate plain linear trend. Moreover, after some executions the time to compute trust remains more or less steady regardless of the amount of devices handled, since the information about other devices do not influence trust quantification.

The achieved results show reasonable performance times as well as the suitability of TACIoT for IoT scenarios, whereby heterogeneous devices can interact each other in a trusted and reliable way.

6.4 Implementation and Integration

The SocIoTal Authorization System is composed of a set of implemented libraries and components explained below:

Capability Client:

This is a HTTPS client, which is intended to making requests to the Capability Manager to obtain capability tokens, which are used to perform actions over entities that are registered with the Context Manager. This client has been integrated in the Mobile Environment, the enablers such as the F2F, and the Web User Environment,

Capability Verifier:
This is a HTTPS server, receiving access requests from Capability Clients. Such access requests contain a capability token, which is evaluated by the Capability Verifier in order to deny or grant the requesting action. This entity makes use of the Capability Evaluator functionality, which is intended to validate capability tokens. This library has been integrated mainly in the Context Manager to validate the tokens.

Capability Manager:
This is a HTTPS server accepting requests for capability tokens generation. Additionally, this entity acts as a HTTP client requesting authorization decisions to the Policy Decision Point.

Policy Decision Point (PDP):

This is a HTTP server based on XACML and node.js. It accepts JSON-encoded XACML requests, which are attached to HTTP requests within the body. The PDP is contacted by the Capability Manager before generating a capability token for the Capability Client.

Policy Administration Point (PAP):

This is the entity responsible for managing the access control policies. It provides the functionality so users can define XACML policies in a user-friendly way. The PAP has a GUI to facilitate the generation of XACML policies.

It should be noted that while the entities have instantiated with HTTPS, a CoAP-DTLS version has already been provided, which is intended to deal with scenarios in which other smart objects can interact each other. As already mentioned, the SocIoTal authorization system has been adapted to address scenarios where users are endowed with X.509 certificates, and other cases in which they do not have such credentials. For further implementation details readers are referred to the SocIoTal wiki.

Section 7 - Trust & Reputation Management

Authentication by what one knows (e.g., password), what one has (e.g., security token), what one is (e.g., biometrics), has been the age-old guiding principle of authentication. However, practical challenges complicate the replication of cryptographic keys (associated with passwords, security tokens, etc.) and biometric templates across the vast network infrastructure that is the IoT. Moreover, as testified by numerous widely publicized incidents, passwords and keys are frequently compromised, biometrics are often defeated. This situation calls for a new non-cryptographic and non-biometric paradigm that is scalable and distributed in nature to complement traditional authentication mechanisms. The solution is trust and reputation (T&R) management. An authenticated user must have sufficient trust/reputation scores to access certain data or functionality. The advent of the social IoT has introduced the social dimension to T&R management. SocIoTal introduces two services for evaluating users' trust and reputation scores. This section details these services. The reputation score relates to the service SOCIOTAL_SV_008 as detailed in deliverable D1.1 [36].

7.1 Trust Manager

There are different ways of defining trust management and trust-reputation systems, in spite of the fact that the context in which computation is quantified may be different, the model and the final goals are the same. As previously described, the trust and reputation systems are mainly based on model defining, trust score computing and management of reputation data, respectively providing secure and efficient data recovery.
The Trust Manager is developed as a component that will enable user to add, remove and manage his own set of rules that are assigned with different weights that will be used to quantify a final reputation score. This score is used to determine the reliability and trustworthiness of devices/context in the IoT scenarios.

The reputation scores are calculated based on rules managed by the user. Different reputation rules can be assigned with weights. Generic model for rules enables mapping between provided JSON format and relational database for mining and extraction of rules previously added over a registration API. The crucial component that Trust Manager utilize to continuously maintain the updated version of score in respect to last attribute value changes is a Context Manager. The Trust Manager utilizes and relies on other SocIoTal platform components, i.e. on Context Manager to receive/push the updated version of the entity values which is used for building the reputation score.

In SocIoTal, with several different enablers that computes the reputation in different context, Trust Manager quantifies a final score based on the enabler's sensed/sent values, thus enabling common reputation score computation. Input for the Trust Manager are not necessarily trust oriented, as the Trust Manager can extract any context from the Context Manager and used it for generating the score.

7.1.1 Implementation and integration

The Trust Manager is developed in Java as REST based web service that offers several API for accessing and managing rules for building reputation score. Generic model for rules enables mapping between provided JSON format and relational database for mining and extraction of rules previously added over a registration API. The crucial component that Trust Manager utilize to continuously maintain the updated version of score in respect to last attribute value changes is the SocIoTal Context Manager. More details are given in the D2.3 [25], where Trust Manager is initially presented and in D1.3.2 [37] were API interfaces are elaborated.

7.2 Location-based Reputation & Trust

The Location-based Trust & reputation experiment aims to demonstrate the ability to locate motes in a real environment thanks to their sensed physical attributes and to derive Trust & Reputation scoring from the contextual collected values. It is fully integrated to the Sociotal platform.

7.2.1 Integration of localization devices in the SocIoTal environment

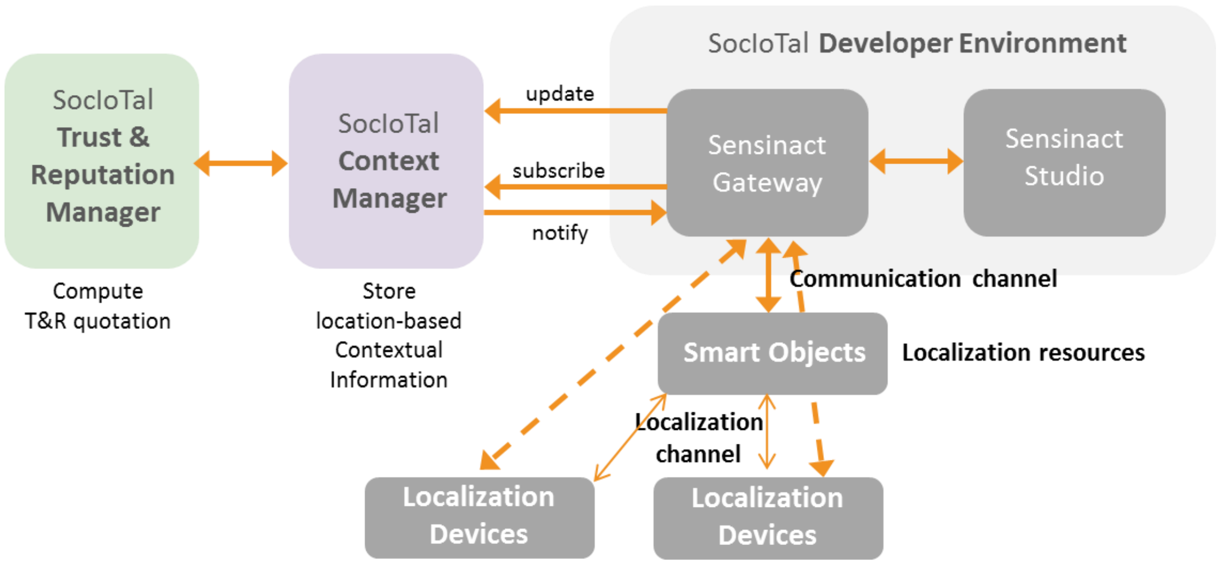


Figure 22: Location-based Trust & Reputation (T&R) integration

In the implemented location-based Trust & Reputation (T&R) framework (see Figure 22), low-level Localization devices and/or Smart Objects are connected to the SocIoTal testbed via the SensiNact Gateway through a wireless communication channel. The so-called localization devices are capable of acquiring raw time-stamped location-dependent information (either absolute information or relative information with respect to their neighbors) that can be interpreted further by any localization engine (i.e., in terms of ranging, positioning or position tracking). The related location resources (e.g., ranges based on peer-to-peer RSSI readings or round trip - time of flight estimates, RSSI fingerprints wrt. multiple Access Points, or even absolute 2D Cartesian coordinates…) depend on the underlying technology (e.g., Zigbee, IR-UWB, Wifi, GPS…). Localization devices can be static or mobile. They could be reference nodes/beacons (possibly with known coordinates) belonging to a pseudo-infrastructure (e.g, anchors of a Wireless Sensor Network, Wifi Access Points) or mobile SmartObjects to be positioned. The latter devices can communicate wirelessly directly with the SensiNact Gateway, or with an intermediate SmartObject that thus acts as a sub-gateway. During this collection phase, any communication standard can be handled such as Bluetooth Low Energy, ZigBee, 6LowPAN, WiFi… Accordingly, the technology used to acquire the raw localization attributes can be independent of the technology used to share and/or centralize these attributes. Overall, the SensiNact Gateway stands as a sink that collects the raw location-dependent measurements along with associated time stamps, which are subsequently transmitted to the SocIoTal Context Manager through the Publish/Subscribe NSGI9/NSGI10 protocol and stored as contextual information in the broker database (see Figure 23). More sophisticated secondary location-dependent information can be computed out of the raw collected/stored data, such as the detected room occupancy (based on raw instantaneous estimated coordinates and prior map knowledge), the probability of room transition (calculated "on the wing" based on the history of detected room occupancy), etc. Finally, an exchange is established between the Context Manager and the Trust & Reputation Component so that location-based R&T scores are computed depending on available raw and secondary location-dependent information (according to a priori composition rules). The latter scores can be pushed back as additional context information (i.e., Figure 23).

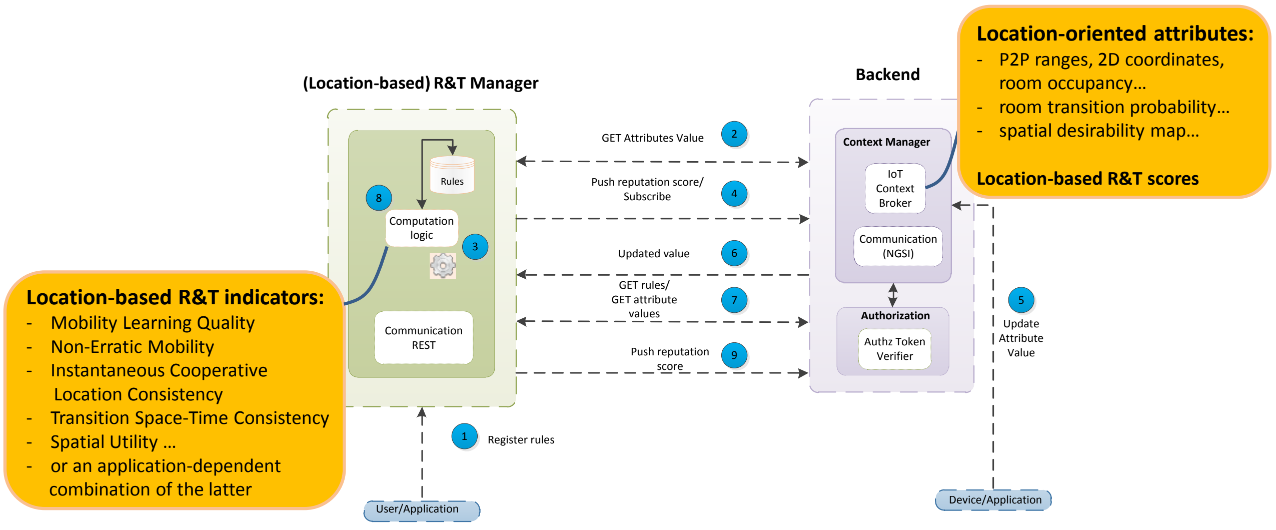

Location-oriented attributes:
- P2P ranges, 2D coordinates, room occupancy…
- room transition probability…
- spatial desirability map…
Location-based R&T scores
(Location-based) R&T Manager
Backend
Rules
Computation logic
Communication REST
Context Manager
IoT Context Broker
Communication (NGSI)
Authorization
Authz Token Verifier
GET Attributes Value
Push reputation score/ Subscribe
Updated value
GET rules/ GET attribute values
Push reputation score
Register rules
Update Attribute Value
User/Application
Device/Application
Location-based R&T indicators:
- Mobility Learning Quality
- Non-Erratic Mobility
- Instantaneous Cooperative Location Consistency
- Transition Space-Time Consistency
- Spatial Utility …
- or an application-dependent combination of the latter

Figure 23: Exchanges between the Trust & Reputation Manager and the Context Manager enabling the computation of Location-based T&R scores

As a summary, the process consists of the several steps illustrated in Figure 23. First, the user connects to its profile handling the requested localization devices ❶. Through its profile, he indicates to the Trust & Reputation (T&R) Component what rules to use for calculating its location-dependent T&R scores ❷ & ❸. The T&R Component will then GET the localization attributes from the Context Manager ❹, perform the score(s) and push the result to the Context Manager database ❺. This score value can be periodically updated taking into consideration new inputs and/or any timely change in the T&R composition rules.

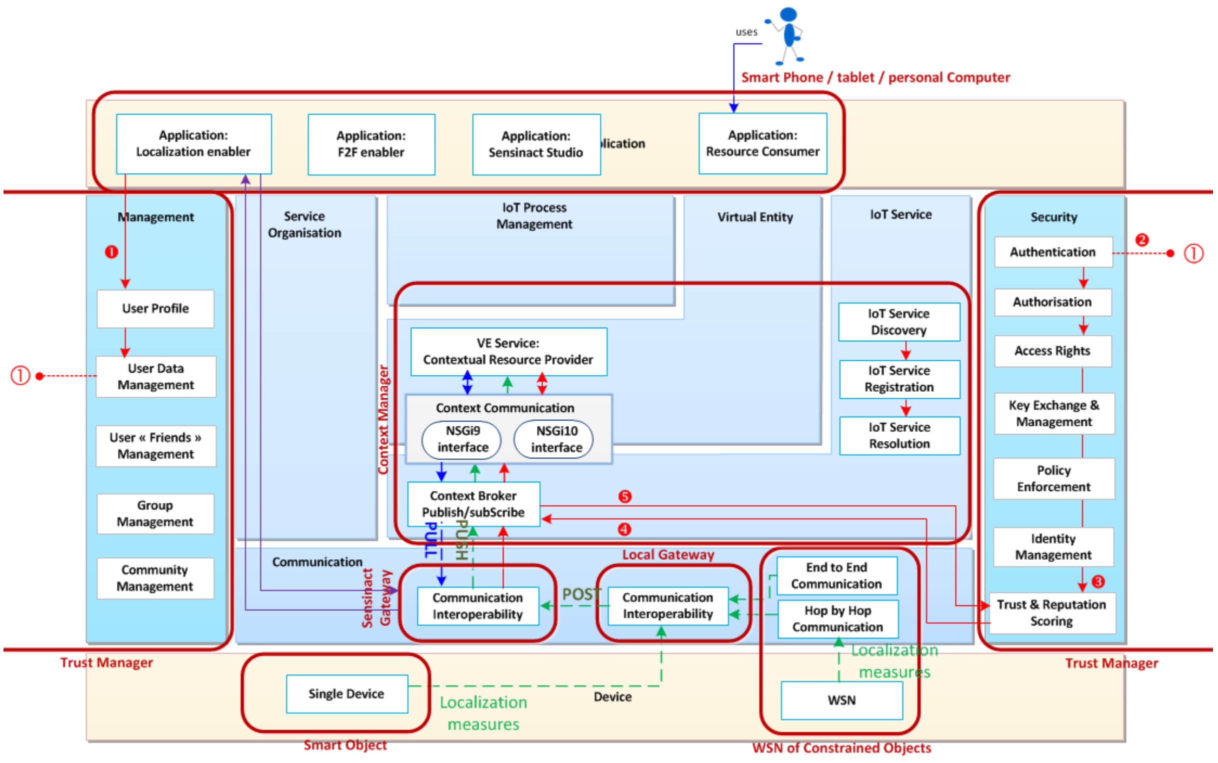


Figure 24: Location-based Trust & Reputation system

7.2.2 Location-oriented attributes as various contextual storage of the information

In the following we focus only on the metrics that will be available in the physically integrated framework (i.e., based on narrowband IEEE 802.15.4 devices operating at 2.4 Ghz in the ISM band). Two levels of location-dependent information are thus required:

- 1st level: Raw location-dependent attributes:
  - $\{RSSI_{ij}(k)\}_{i=1..N_u, j\in Ne_i(k)}$ → Set of single-link RSSI measurements of Device i w.r.t. other neighboring devices $j \in \mathrm{Ne}(k)$ (mobile or anchors), where $\mathrm{Ne_i}(k)$ stands for the neighborhood of Device i at time epoch k, and N_u is the number of mobile users equipped with localized device;
  - $\{\tilde{d}_{ij}(k)\}_{i=1..N_u, j\in Ne_i(k)}$ → Set of single-link relative range measurements of Device i w.r.t. other neighboring devices $j \in \mathrm{Ne_i}(k)$, where $\mathrm{Ne_i}(k)$ stands for the neighborhood of Device i at time epoch k, according to an a priori calibrated path loss model where each single range measurement is determined out of the $RSSI$ reading as follows (omitting both time and device subscripts for simplicity):

    $\tilde{d} = \exp(M)$ with $M = \dfrac{(RSSI_0 - RSSI)}{10\alpha} + \log(d_0)$, where $RSSI_0$ is the reference RSSI reading at the reference distance d_0 (usually 1m), and α is the path loss exponent (calibrated).
  - $\{\tilde{x}_i(k), \tilde{y}_i(k)\}_{i=1..N_u}$→ Set of mobile devices' estimated absolute 2D Cartesian coordinates at time epoch k (among N_u equipped users), based on for example non-

cooperative Linearized Least Squares positioning with respect to static anchors only $j \in Ne_i(k), j \leq N_a$ (with N_a anchors)

- $\{t_i(k)\}_{i=1..N_u}$ → Explicit time-stamp associated with the delivery of Device i's estimated absolute 2D coordinates, set of single-link relative range measurements w.r.t. neighboring nodes $j \in Ne_i(k)$.

The implementation of this location-dependent attribute algorithm in Java is detailed below. It could be implemented in any device that supports Java. For SocIoTal trials, it runs in the local Gateway that pushes the time-stamped 2D (X,Y) coordinates to the SensiNact gateway via an HTTP POST message. Subsequently, the SensiNact gateway pushes the contextual location coordinates to the SocIoTal Context Manager.

```java
public class LocationCalculator {
    private static final int MEAN_SAMPLES = 3;
    private static final double ALPHA = 4.17;
    private static final double REF_RSSI = -54;
    private static final double REF_DISTANCE = 1;
    private static final Map<String, Coord> anchorMap = new HashMap<>();

    static {
         // Fill in the anchor map
        anchorMap.put("00124b00060d85b3", new Coord( 0.00,  0.00));
        anchorMap.put("00124b00060d84c4", new Coord(26.00,  0.00));
        anchorMap.put("00124b00060d8475", new Coord(46.00,  0.00));
        anchorMap.put("00124b00060d8473", new Coord(54.00, 65.00));
        anchorMap.put("00124b00060df480", new Coord(25.00, 65.00));
        anchorMap.put("00124b00060d84ba", new Coord( 0.00, 65.00));
        anchorMap.put("00124b00060d84f5", new Coord( 0.00, 38.00));
        anchorMap.put("00124b00060d84aa", new Coord(23.00, 34.00));
    }

    private final Map<String, List<Integer>> rawRssi = new HashMap<>();

    private Map<String, Double> meanRssi() {
        Map<String, Double> meanRssi = new HashMap<>();
        for (Map.Entry<String, List<Integer>> macRssi : rawRssi.entrySet()) {
            final List<Integer> rssis = macRssi.getValue();
            final String mac = macRssi.getKey();
            if (rssis.size() < MEAN_SAMPLES)
                continue;
            double mean = 0;
            for (int rssi : rssis) {
                mean += rssi;
            }
            mean /= rssis.size();
            meanRssi.put(mac, mean);
        }
        return meanRssi;
    }

    private double distance(double rssi) {
        // d = ( d_ref × 10^(rssi_ref - rssi) / (10 × α) )²
        return Math.pow(REF_DISTANCE *
                        Math.pow(10, (REF_RSSI - rssi)
                                 / (10 * ALPHA)), 2);
    }

    public Coord computeLeastMeanSquare() {
        // Mapping of anchor address ⇔ mean RSSI with this anchor
        Map<String, Double> meanRssi = meanRssi();
        int size = meanRssi.size();
        if (size < 3)
            // We cannot compute localization with less than three anchors
            return null;

        final List<String> macs = new ArrayList<>(meanRssi.keySet());
        // Choose the first anchor as an arbitrary reference for computations
        final String firstMac = macs.get(0);
        final Coord firstCoord = anchorMap.get(firstMac);
        final double firstDistance = distance(meanRssi.get(firstMac));

        // Fill the matrices
        Matrix a = new Basic1DMatrix(size, 2);
        Matrix h = new Basic1DMatrix(size, 1);
        for (int i = 1; i < meanRssi.size(); i++) {
            final String mac = macs.get(i);
            final Coord coord = anchorMap.get(mac);

            //      ┌ x_i - x_0   y_i - y_0 ┐
            // A =  │           …           │
            //      └ x_k - x_0   y_k - y_0 ┘
            a.set(i, 0, coord.x - firstCoord.x);
            a.set(i, 1, coord.y - firstCoord.y);

            //      ┌ x_i² - x_0² + y_i² - y_0² + d_0² - d_i² ┐
            // h =  │                    …                    │
            //      └ x_k² - x_0² + y_k² - y_0² + d_0² - d_k² ┘
            //        └────────────────────────┘
            //                  hprime
            final double hprime = Math.pow(coord.x, 2) - Math.pow(firstCoord.x, 2)
                                + Math.pow(coord.y, 2) - Math.pow(firstCoord.y, 2);
            h.set(i, 0, hprime + firstDistance - distance(meanRssi.get(mac)));
        }

        // Apply the Least Square formula
        // X = ½ (Aᵀ A)⁻¹ Aᵀ h
        final Matrix aTransposed = a.transpose();
        final Matrix result;
        try {
            result = aTransposed
                    .multiply(a)
                    .withInverter(LinearAlgebra.InverterFactory.GAUSS_JORDAN)
                    .inverse()
                    .multiply(aTransposed)
                    .multiply(h)
                    .multiply(.5);
        } catch (IllegalArgumentException err) {
            // Not invertible
            return null;
        }
        return new Coord(result.get(0, 0), result.get(1, 0));
    }
```

- 2nd level: Secondary location-dependent attributes:
    - $\{\tilde{r}_i(k)\}_{i=1..N_u} \rightarrow$ Mobile Device i's detected room based on estimated absolute 2D coordinates $\{\tilde{x}_i(k), \tilde{y}_i(k)\}_{i=1..N_u}$ at time epoch k and the a priori building map.
    - Optionally, estimated Hidden Markov Model (HMM) matrices of state transitions (i.e., room change probability in our case) and emissions (i.e., room detection noise distribution in our case), $\widetilde{T}_\iota(k)$ and $\widetilde{E}_\iota(k)$ (i.e., as function of the time epoch k), given a sequence of emissions $\{\tilde{r}_i(k)\}_{k=1..K_{Learn}}$ (i.e., detected rooms) and a sequence of true occupied states $\{s_i(k)\}_{k=1..K_{Learn}}$ in case of preliminary active learning phase (optional),

using for instance a Maximum Likelihood (ML) approach. Alternatively, instead of being re-computed “on the fly” based on the latest localization results, transitions probability can be assumed known a priori accordingly to any other calibration/acquisition procedures (possibly independent of the online localization technology) or a given usage pattern (e.g., one can know in advance the expected regular behaviour e.g., in a public transportation context). In the latter case, $\widetilde{T}_i(k) = T_i = constant$. In a very minimal configuration, rather than capturing the actual mobility habits of the users, this transitions probability matrix can at least accounts for physically non-feasible or forbidden moves within the refreshment period (e.g., one pedestrian cannot walk from one given office room to another very distant/far room in a given building within 30 sec or some users may not be allowed to access a certain room from the corridor, both moves being represented by a null probability). Then all the remaining feasible/allowed transitions can be simply associated with equivalent probabilities (i.e., ensuring the same probability for the transitions conditioned on the currently detected room). This simplified option will be retained for convenience in the integration/demonstration framework.

7.2.3 Trust & Reputation indicators as new rating mechanisms in the T&R Manager

In the following, we describe only simplified rating strategies in line with the current integration efforts. Note that more complex scoring mechanisms and composition rules, which are not taken into account herein (e.g., regarding user’s spatial predictability in terms of mobility learning quality or erratic mobile behaviour, or user’s spatial utility…), are detailed in D2.3 [6]. The considered location-based T&R rating mechanism mostly aspires to capture the reliability of instantaneous user’s location.

<u>Instantaneous Cooperative Location Consistency</u> (ICLC): This ingredient reflects the compatibility of the current estimated position with the positions claimed/estimated by its neighbors and the perceived relative distances. Practically, at each mobile $\mathrm{i} = 1..\mathrm{N_u}$, one checks the reliability of each of its mobile neighborsj $\in \mathrm{Ne_i(k)}$, by verifying the compatibility between the instantaneous locations $\{\tilde{x}_j(k), \tilde{y}_j(k)\}$, the instantaneous locally estimated location $\{\tilde{x}_i(k), \tilde{y}_i(k)\}$ and peer-to-peer ranging measurements $\{\tilde{d}_{ij}(k)\}_{i=1..N_u, j\in Ne_i(k)}$ with respect to these neighbors. We consider a basic approach producing intermediary binary ratings, by computing reputation scores through statistical updating of a Beta probability density function (pdf) where positive hard-decision outcomes represent situations where no outlier has been detected (positive). Note that detected outliers (negative) may result from largely erroneous measurements (affecting ranging and/or positioning at one or two of the involved legitimate devices) or from erroneous information from a malicious mobile (while claiming/storing its own estimated location). The Beta pdf somehow expresses the uncertain probability that future interactions will be also positive. We take the expectation $\mathrm{R_{ij}(k)}$ of this Beta pdf at each new epoch (function of the relative number of positives and negatives).

```
For $k = 1..K$ % Loop on time epochs
   For $i = 1..N_u$ % Loop on users
      For j $\in Ne_i(k)$ % Loop on users’ neighbors
      $\Delta_{ij}(k) = \left| \tilde{d}_{ij}(k) - \sqrt{\left(\tilde{x}_i(k) - \tilde{x}_j(k)\right)^2 + \left(\tilde{y}_i(k) - \tilde{y}_j(k)\right)^2} \right|$

         % Difference between the RSSI-based perceived relative distance and the
         corresponding distance based on latest position estimates (on both sides of the link),
         to be compared with the a priori threshold $Th_d$

         For $k > 1$
```

```
            $\begin{cases} r_{ij}(k)=r_{ij}(k-1)+1 \quad \text{and} \quad s_{ij}(k)=s_{ij}(k-1) \quad \text{if} \quad \Delta_{ij}(k)\le Th_d \\ r_{ij}(k)=r_{ij}(k-1) \quad \text{and} \quad s_{ij}(k)=s_{ij}(k-1)+1 \quad \text{otherwise} \end{cases}$
            % In the steady-state regime, if the difference between the perceived RSSI-based distance and the distance based on the latest estimated positions exceeds the threshold, then the intermediate scores $r$ and $s$ are updated to account for a timely anomaly  detection (integrating events over time).
         End
         $\begin{cases} r_{ij}(1)=0 \\ s_{ij}(1)=0 \end{cases}$
         $\mathrm{R_{ij}(k)} = \frac{1+\mathrm{r_{ij}(k)}}{2+\mathrm{r_{ij}(k)}+\mathrm{s_{ij}(k)}}$ $R_{ij}(k)=\frac{1+r_{ij}(k)}{2+r_{ij}(k)+s_{ij}(k)}$
      End
   End
End
```

The scores reflect the confidence in the current claimed positions wrt. The measured distance $\tilde{\mathrm{d}}_{\mathrm{ij}}(\mathrm{k})$. If the latter (based on the RSSI measurement) is close enough from the expected distance $\sqrt{\left(\tilde{\mathrm{x}}_{\mathrm{i}}(\mathrm{k})-\tilde{\mathrm{x}}_{\mathrm{j}}(\mathrm{k})\right)^2+\left(\tilde{\mathrm{y}}_{\mathrm{i}}(\mathrm{k})-\tilde{\mathrm{y}}_{\mathrm{j}}(\mathrm{k})\right)^2}$ (and thus compliant with the latest claimed positions estimates), then the difference $\Delta_{\mathrm{ij}}(\mathrm{k})$ is smaller than the a priori threshold $\mathrm{Th_d}$ and the “confidence” coeff is incremented by +1 , whereas the “doubt” coeff remains the same… the final score R below combines both r and s (R being the expectation of a pdf expressing the uncertain probability that future interactions will be also positive, See the introducing text above before the equations).

Then one can centralizes all the intermediate pair-wise scores to compute an average score $R_i(k)$ $\mathrm{R_i(k)}$ per user (based uniquely on the perception of his mobile follows $j\in Ne_i(k)$ $\mathrm{j} \in \mathrm{Ne_i(k)}$), as follows:

```
For $k=1...K$
   For $\mathrm{i}=1..\mathrm{N_u}$
      For j ∈ $\mathrm{Ne_i(k)}$
                     $\mathrm{R_i(k)} = \frac{1}{|\{\mathrm{j} \in \mathrm{Ne_i(k)}\}|}\sum_{\mathrm{j}\in\mathrm{Ne_i(k)}}\mathrm{R_{ji}(k)}$
         % Average of pair-wise scores

               $\mathrm{ICLC_i(k)} = \frac{\mathrm{R_i(k)}}{1-\mathrm{P_{FA}}}$
         % Final ICLC score
      End
   End
End
```

Note that we introduce here a normalization factor $(1-\mathrm{P_{FA}})$ so that the ICLC indicator falls within the interval [0,1]. This factor can be theoretically computed a priori depending on the detection threshold setting Th_d, based on expected positioning and ranging error regimes (i.e., their standard deviations) while assuming Gaussian centered random errors and legitimate devices on both side of the link for each peer-to-peer intermediary rating.

Transition Space-Time Consistency (TSTC): This ingredient reflects if (and to which extent) the successive positions estimated/claimed by the users are plausible or even just physically feasible. One verifies the compatibility between the claimed instantaneous sequences and the learnt mobility patterns, detecting forbidden state transitions violating the learnt HMM transition probability matrix (e.g., detecting a non-physical transition from one room to another too distant room within a very short time duration) or very unlikely transitions (anyway with limited impact on scores in case of false alarms) against an arbitrarily low detection threshold Th_t (e.g., 5% in the following):

```
For $k = 1...K$
    For $\mathrm{i} = 1..\mathrm{N_u}$
        $$\Theta_i(k) = \tilde{T}(\tilde{r}_i(k-1), \tilde{r}_i(k)) \quad i = 1..N_u$$
        % Estimated probability of the observed room transition according to the a priori room transition probability matrix $T$ whose entries are respectively the latest known occupied room and the current detected room.
        For $k > 1$
            $\begin{cases} r_i'(k) = r_i'(k-1) + 1 \quad \text{and} \quad s_i'(k) = s_i'(k-1) \quad \text{if} \quad \Theta_i(k) \le Th_t \\ r_i'(k) = r_i'(k-1) \quad \text{and} \quad s_i'(k) = s_i'(k-1) + 1 \quad \text{otherwise} \end{cases}$
        End
        $\begin{cases} r_i'(1) = 0 \\ s_i'(1) = 0 \end{cases}$
        $$\mathrm{TSTC_i(k)} = \frac{1 + r'_i(k)}{2 + r'_i(k) + s'_i(k)}$$
        % Expectation of the Beta pdf
    End
End
```

Global score: A global score can be computed as a weighted combination of the previous ICLC and TSTC ingredients. As an example, equal weights (i.e., γ = 0.5) are arbitrarily chosen hereafter.

```
For $k = 1...K$
    For $i = 1...N_u$
        $GS_i(k) = \gamma TSTC_i(k) + (1-\gamma) ICLC_i(k)$
    End
End
```

7.2.4 Delivery of contextual location from the SensiNact Gateway to the SocIoT Context Manager

Before we explain the way the communication is handled between SensiNact and the SocIoTal context manager, let us recap the overall integration of SensiNact (see Figure 22).

SensiNact is built out of two components: the SensiNact Gateway and the SensiNact Studio. The goal of the SensiNact Gateway is to provide a unified access to local and remote devices, offering a unified API to third parties. This API is then used by SensiNact Studio, which provides facilities for gateway monitoring and application creation. The Studio is optional, and in our case, it has been only used for troubleshooting purpose.

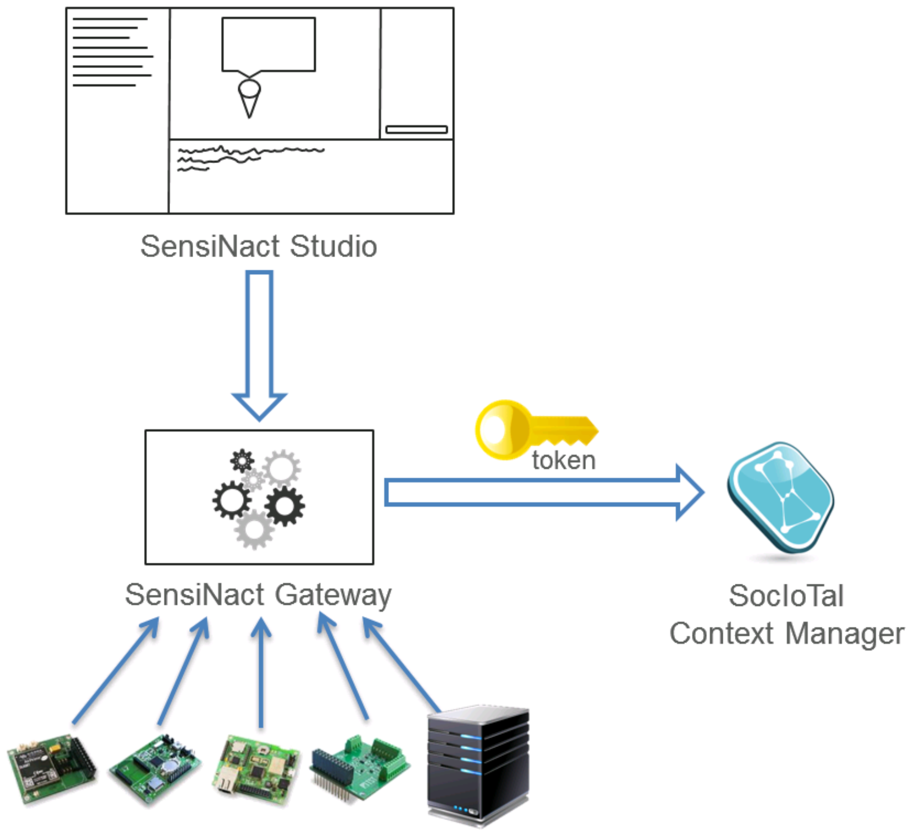


Figure 25: SensiNact integration in SocIoTal

Once location data has been retrieved by SensiNact Gateway, this later has to send it to the SocIoTal Context Manager. To do this, a token is mandatory. This one has to be generated using credentials (login and password). For security purpose, it's a good practice to generate this token manually, and to provide it to the gateway. This way, if the server on which the gateway is running is compromised, only the token can be accessed by attackers. As soon as a new token is generated, the old one cannot be used anymore. This is the reason why token generation is not shown in Figure 25.

To send information to the SocIoTal Context Manager, SensiNact Gateway uses NGSI 10 API. This one provides three functionalities for updating context information: APPEND, UPDATE and DELETE. APPEND is used for creating context entities and attributes. UPDATE can replace existing values (and/or metadata) with new ones. DELETE is used for removing context entities at the NGSI 10 level. Bear in mind this does not affect the model defined using NGSI 9.

Since the full description of the API can be found in the SocIoTal tutorial [22] (paragraph 4.2.3), we are only focusing here on the UPDATE function which is the one we have used extensively.

SocIoTal SERVER	SocIoTal_CM_V2_IP:PORT	
POST	/SocIoTal_CM_REST_V2/NGSI10_API/updateContext	
HEADERS		
	Content-type	application/json
	Accept	application/json
	Capability-token	[JSON file/text including SocIoTal Capability token]
Payload:		
{		

```
    "contextElements": [{
        "type": " urn:x-org:sociotal:resource:device ",
        "isPattern": "false",
        "id": "SocIoTal:IoTWeek:WeatherStation:Dev_001",
        "attributes": [
            {
                "name": "Location",
                "value": "43.472057, -3.800156",
                "type": "http://sensorml.com/ont/swe/property/Location",
                "metadatas": [{
                    "name": "WorldGeographicReferenceSystem",
                    "value": "WSG84",
                    "type": "http://sensorml.com/ont/swe/property/WorldGeographicReferenceSystem"
                }]
            }],
    }],
    "updateAction": "UPDATE"
}
```

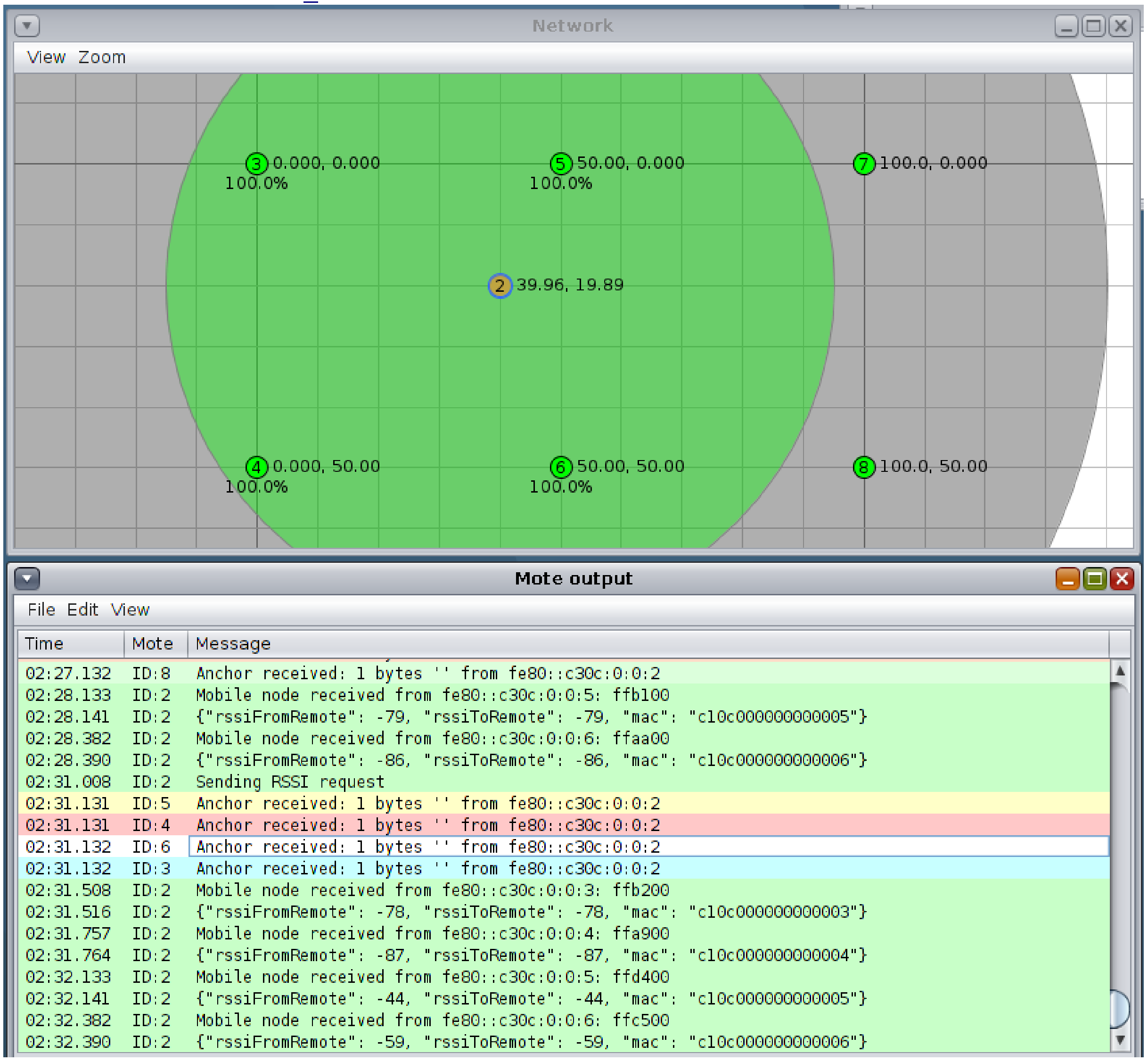
Error Messages (Status codes):

errorCode element (Response body)	Message/Additional info
200 “OK”	The request has been properly executed and the result has been included in the response payload. (Initial versions won’t show error code in this case)
400 “BAD REQUEST”	Any of the fields have been not properly performed. The response payload will include further info if available.
401 “UNAUTHORIZED”	The Capability-token is either corrupted, not valid or does not contain the appropriated credentials
404 “NOT FOUND”	No context elements found (Check the id element)
500 “INTERNAL SERVER ERROR”	Error accessing Context Broker element of the Context Manager
Status Code (returned by the server)	Message/Additional info
405 “METHOD NOT ALLOWED”	Check you’re using POST
415 “UNSUPPORTED MEDIA TYPE”	Check “Content-type application/json” header and the payload
500 “INTERNAL SERVER ERROR”	Error accessing Context Manager server/service

In this example, we can see that the Capability token has to be provided in the HTTP POST request header. This is a JSON description, describing the user rights and signed by the issuer. The payload contains the information to be updated (the location), as well as the action to be executed on the server: UPDATE.

7.2.5 Conclusion

The Location-based Trust & Reputation experiment enables a complete integration of the computation of (X,Y) coordinates of a constrained mote location based on its RSSI attributes in the SocIoTal platform. Moreover, it allows deducing Trust & Reputation scoring thanks to several estimation mechanisms that may be chosen by the end-user or dedicated to the need of the application. Simulation results showing the location of the anchors and the way to collect the RSSI values are presented in Figure 26.



```
02:27.132  ID:8   Anchor received: 1 bytes '' from fe80::c30c:0:0:2
02:28.133  ID:2   Mobile node received from fe80::c30c:0:0:5: ffb100
02:28.141  ID:2   {"rssiFromRemote": -79, "rssiToRemote": -79, "mac": "c10c000000000005"}
02:28.382  ID:2   Mobile node received from fe80::c30c:0:0:6: ffaa00
02:28.390  ID:2   {"rssiFromRemote": -86, "rssiToRemote": -86, "mac": "c10c000000000006"}
02:31.008  ID:2   Sending RSSI request
02:31.131  ID:5   Anchor received: 1 bytes '' from fe80::c30c:0:0:2
02:31.131  ID:4   Anchor received: 1 bytes '' from fe80::c30c:0:0:2
02:31.132  ID:6   Anchor received: 1 bytes '' from fe80::c30c:0:0:2
02:31.132  ID:3   Anchor received: 1 bytes '' from fe80::c30c:0:0:2
02:31.508  ID:2   Mobile node received from fe80::c30c:0:0:3: ffb200
02:31.516  ID:2   {"rssiFromRemote": -78, "rssiToRemote": -78, "mac": "c10c000000000003"}
02:31.757  ID:2   Mobile node received from fe80::c30c:0:0:4: ffa900
02:31.764  ID:2   {"rssiFromRemote": -87, "rssiToRemote": -87, "mac": "c10c000000000004"}
02:32.133  ID:2   Mobile node received from fe80::c30c:0:0:5: ffd400
02:32.141  ID:2   {"rssiFromRemote": -44, "rssiToRemote": -44, "mac": "c10c000000000005"}
02:32.382  ID:2   Mobile node received from fe80::c30c:0:0:6: ffc500
02:32.390  ID:2   {"rssiFromRemote": -59, "rssiToRemote": -59, "mac": "c10c000000000006"}
```

Figure 26: Collect of the RSSI values from the anchors

7.2.6 Implementation and integration

Further implementation is envisaged, notably the integration of the cooperative scoring algorithm details in paragraph 7.2.3. This techniques needs that several mobile nodes cooperate to estimate a trust & reputation score. The development will be performed in Java and implemented in the local gateway for a fist proof of concept before to migrate in the Sociotal Trust & Reputation component located in the cloud.

7.3 User Trust using the Gait Recognition App

In IoT frameworks, it is important to determine the source and destination of information and verify that the sender or receiver are actually who they claim to be. There are two aspects to this.

- The first is a commonly addressed problem: is the actual device who it claims to be? An example of when this is not the case is IP address spoofing, when the source IP address in an IP packet is replaced with a fake one.
- The second aspect is whether the device is being operated by the rightful user. To solve this problem, the manner in which the device is being used must be examined in order to determine whether this is true or not.

The aim of user trust is to determine whether smart phone used by SocIoTal users are being operated by the rightful user, or whether an imposter has obtained the phone through illegitimate means. The aim is to determine a metric which quantifies how much the SocIoTal framework trusts the user. In this case, trust means how much the SocIoTal framework trusts that the rightful user is currently in possession of the smartphone. This is an aspect of the SocIoTal Reputation Service [SOCIOTAL_SV_008] as detailed in deliverable D1.1.

Identification of whether the correct user is currently in possession of the mobile phone is done using the gait recognition app. This section will include the final definition of the gait recognition algorithm and a summary of all previous results. In addition, the integration with the SocIoTal platform will be detailed. This includes the interaction with the Context Manager and the role of the Trust Manager.

The SocIoTal platform provides access to resources based on both context and user identity. In the case of user identity, the possession and access to the resources on a mobile actually provide a source of identification. For example, if a user can access a mobile phone and submit a resource request from the phone, it is often assumed that the mobile phone is in the possession of the rightful user, and not an imposter. A further security issue arises when a mobile is producing and consuming content when the phone is not actively being used, for example the phone is broadcasting its location or receiving some data while the phone is in the pocket. Modern security methods provide a measure of solution to the first problem, but not the second. In addition, it provides a more automated form of user authentication where it occurs unobtrusively without the need to enter a password of swipe pattern, or scan a figure print. Another advantage is that it uses sensors on the phone that are usually embedded and used for a variety of apps, rather than a piece of hardware that has a dedicated use, as in the case of a fingerprint scanner.

The gait recognition app aims to provide a solution to both problems. The app analyses the users walking pattern and aims to determine if the rightful user is in possession of the phone, or if an imposter has obtained the phone. The gait recognition app was detailed in D2.3 “Reputation and Trust Management” [25]. Further results were presented in D5.2 “SocIoTal Evaluation” [20]. In this deliverable, the final specification of the app is detailed, including the integration of the app into the SocIoTal platform

7.3.1 Gait Recognition Algorithm

The gait recognition app uses a machine learning method (KPCA) to construct a model of the walking pattern of the rightful user. This is then used to determine when the rightful user is in possession of the phone, and when an imposter is in possession.

7.3.2 Implementation and integration

The gait recognition algorithm is implemented as an app for the Android Mobile Operating System, but it is extendable to other mobile platforms as well. The base sensing system makes use of the app that was developed for the face-to-face enabler, which was initially presented in Deliverable D3.1.1. This app is implemented using a recycling multithreaded approach, where the collection and classification is executed in separate threads retrieved from a thread pool in order reduce the computational burden.

7.3.3 Performance and acceptability

Deliverable D5.2 presented performance results for the gait recognition app. An important aspect of a security device is to minimize the number of unauthorized access attempts, while maximizing the number of authorized access attempts. An evaluation involving 10 users shows that the app is able to minimize the number of unauthorized access attempts. The highest percentage of unauthorized access attempts was 5%, with the lowest being 0%. The percentage of authorized attempts varies has a minimum value of 33% and has a highest value of 100%. This shows that for some users the app is easily able to identify the rightful user of the phone, and for others the app has more difficulty.

In addition to a performance evaluation, Deliverable D5.2 provides metrics on acceptance. Users who took part in the evaluation were presented with the results and an explanation of their meaning, and asked to provide a value from 1 to 10 that indicated their trust in the application to perform the task of user identification correctly. The average score for the user trust metric was 8.4, quite a high value considering the novelty of the application. Individual scores range from 3 to 10.

7.3.4 Integration and Communication

The gait recognition app is integrated into the SocIoTal platform. It communicates with the Context Manager using a RESTful communication protocol with the information represented in a JSON structure. The process of communication is detailed in Figure 27. Data generated by the smartphone is applied to the model, which will then determine whether the rightful user or the imposter is currently walking with the phone. This result is then communicated to the context manager using a RESTful HTTP Post operation. The Context Manager then has the current status for the ID of the smartphone. The current status is continually updated as walking occurs. More details can be found in D4.3 [28].

The information generated by the Gait Recognition App is used by the Trust Manager to determine whether the phone should be trusted, i.e., is it currently being held by the rightful user. The Trust Manager queries the Context Manager to determine the current status of the phone with a specific ID, and the result of “Rightful User” or “Imposter” will be returned.

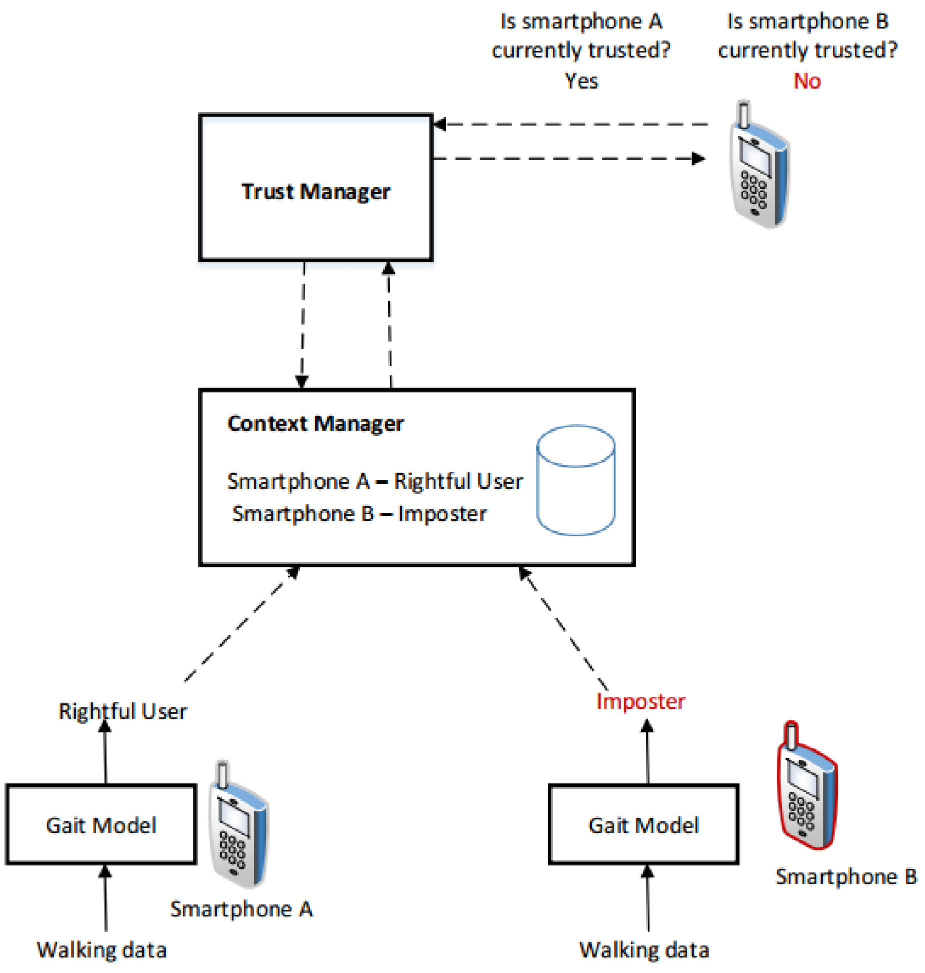


Figure 27: Interaction of the gait recognition app with the SocIoTal framework.

7.4 Real-world social graph

Social relationships constitute a significant part of human behaviour. While people socially interact in their daily lives they establish different types of social relationships. Understanding these relationships are an important part of psychology and other areas, which allows the quantification of other part of human behaviour such as trust relationships.

Scientists initially focused on quantifying social relationships among people through less automated methods such as questionnaires and surveys. These techniques suffer from a large amount of error induced by the involved human factor [8]. Various techniques have been developed to understand social relationships among users in on-line social networks. These techniques are focused on information retrieved from users' social accounts. However, as research has shown on-line and real-world social networks tend to differ [9]. Currently there is no work that is able to measure social relationships among people through mobile phones based on contextual information from their real-world interactions.

Very few works focused on extracting social graphs from real-world situations. To produce real-world social graph there is a need to understand the social links among people such as friendships and social relations. As this type of graphs focuses on real-world situations, pervasive self-acting tools are required that will be able to measure the social links among people. Initially, literature focused on creating obtrusive monitoring mobile devices to log users' social interactions [10] and derived a real-world social graph. In order to limit the obtrusiveness of the system, researchers extracted a real-world social graph from Bluetooth proximity using both mobile and stationary devices [11]. This approach did not consider the existence of a social interaction and required the deployment of additional hardware in the environment. In [12], authors developed an approach based on active RFID-tags that detected the social interactions among people and also derived a real-world social graph. The approach required the deployment of RFID-readers in the monitored environment, which increases the intrusiveness of the system and also does not allow large-scale deployment.

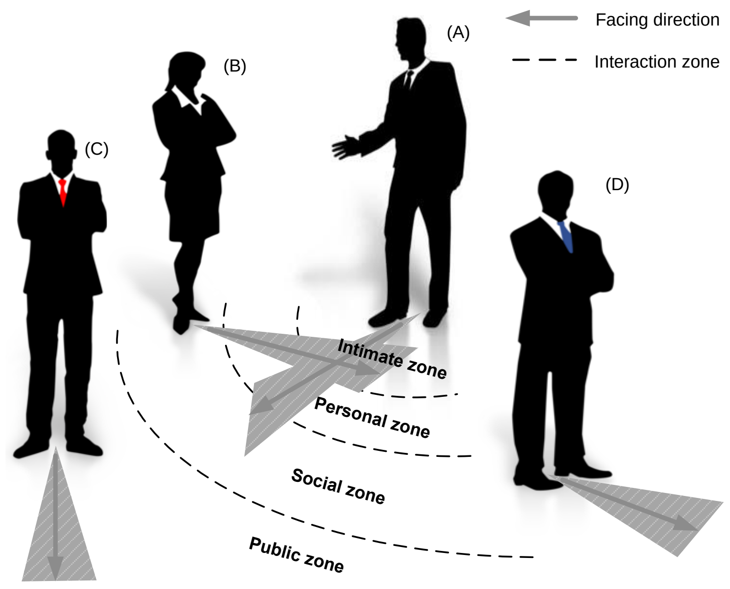


Figure 28. This figure depicts the importance of spatial arrangement in order to understand the existence of social interactions and the social relation among people.

The quantification mechanism for social relationships is based upon the face-to-face enabler work [13] for detecting real-world social interactions and social relations (See Figure 28). An opportunistic sensing and inference system was presented, which considered only smartphones and was independent on any additional hardware or firmware modifications. Hall [14] introduced a taxonomy of the interpersonal distance among people called interaction zones and mapped them to their social relation. Thus, an interpersonal distance estimation technique was built upon the notion of interaction zones to understand peoples' social relations. Also, a relative orientation computation approach was proposed to consider the directionality of users in proximity. A collaborative sensing component allowed the smartphones that had deployed the application, to perform ad-hoc communication and exchange information. In this section, we build upon our prior work and leverage the above contextual information to extract a real-world social graph.

7.4.1 System Design

Real-world social interaction detection may provide significant information regarding a person's social behaviour. Literature has mainly focused on leveraging graphs extracted from on-line social networks. As shown in [9], on-line social graphs include a considerable amount of false positive links among users. One important piece of information that could be derived from a person's daily interactions is the real-world social graph. This constitutes a more realistic representation of the social graph with whom users interact in a daily basis.

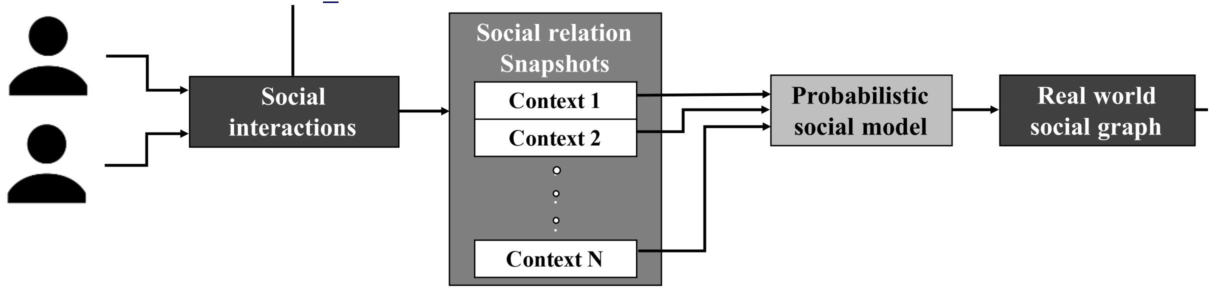


Figure 29. This figure depicts the architecture of the system that infers the social relation among people and extracts a real-world social graph.

7.4.1.1 Approach

A dynamic social graph is proposed that adjusts based on real-world situations. While users daily interact, they form certain social relations. A real-world social graph (see Figure 29) is derived from snapshots of users' social relations provided by the interpersonal distance estimation [13]. The edges of the graph are weighted with the social relation and the confidence of the social relation estimation. The model initially computes for each context the confidence of the social relation estimates and selects the most confident estimate in each context. Given a series of probability weights provided by psychology, the confidence of each social relation is computed across all contexts and the most confident is selected. The next subsection describes in more detail the social relation model.

7.4.1.2 Social relation model

Initially, a set S is defined that includes all the possible social relations that two people may have with respect to the system.

$$\mathbb{S} := \{r : r \in \{Public, Social, Personal\}\} \qquad (1)$$

The confidence of each social relation between a pair of users in a particular context is provided by the Equation:

$$P(r) = \frac{Q(r)}{N}, where\ r \in \mathbb{S}\ and\ Q(r), N \in \mathbb{N}^{+} \qquad (2)$$

where Q(r) is the number of inferences that are related to social relation r and N is the total number of social relation inferences. The most confident social relation between a pair of people in a particular context is calculated through Equation

$$f(r) = argmax_{r\in\mathbb{S}}\left\{\frac{Q(r)}{N}\right\} \qquad (3)$$

In essence, Equation (3) computes the confidence of the social relation between two people in a particular context. In order to consider multiple contexts in our model, we provide Equation (4) including the weighting probability of how significant is the estimated social relation r in a certain particular context cj.

$$\begin{aligned} R\,(r) &= argmax_{r\in\mathbb{S}}\left\{\frac{1}{C}\sum_{j=1}^{C} P(r \cap c_j)\cdot w_j\right\} \\ &= argmax_{r\in\mathbb{S}}\left\{\frac{1}{C}\sum_{j=1}^{C} P(c_j)\cdot P(r \mid c_j)\cdot w_j\right\} \qquad (4)\end{aligned}$$

where P (cj) is the probability of the users being in context cj , P (r | cj) is the probability of the two people having a social relation r given that they are in context cj and wj is a probabilistic weight of a particular context cj with respect to a social relation.

7.4.2 Experimental Setup

This section provides information related to the experimental setup. The purpose of this experiment is to evaluate the proposed social relationship measurement mechanism. Participants were placed in an indoor room and were socially interacting in order to observe the establishment of social relationships.

In the experiment, five participants where involved that did not have any prior knowledge about each other. An important moment when a social relationship is established, is the initial contact [15]. The employment in the experiment of people that do not know each other, allows the observation of the establishment of the social relationships in short-term. This will facilitate the evaluation of the system in a small- scale experiment, as a proof of concept.

In order to establish ground truth, participants were asked questions about their social relationships before the beginning of the experiment and at the end of the experiment. This would indicate the change of the interpersonal relationship that was established during the social interaction in the context of the experiment. Providing the participants with the questionnaire before the beginning of the experiment verified that they did not have any prior knowledge about each other. The questionnaire provided to the participants was identical before the beginning and after the end of the experiment.

Participants were placed in a common indoor environment i.e. a conference room. Each participant initially answered the questionnaire and then was provided with an HTC One S mobile phone having deployed the system application. The participants entered the room and started interacting in groups for 1 hour. At the end of the experiment, participants answered again the questionnaire to express their perception about what kind of social relationships they established during the experiment.

7.4.3 Results

Figure 30 depicts the real-world social graphs extracted from the system and from the surveys provided to the participants. The inferred social graph includes also weights of the social relations in order to show how confident the estimation is. The ground truth social graph provided by the participants shows only the relation among them and does not include any weighting.

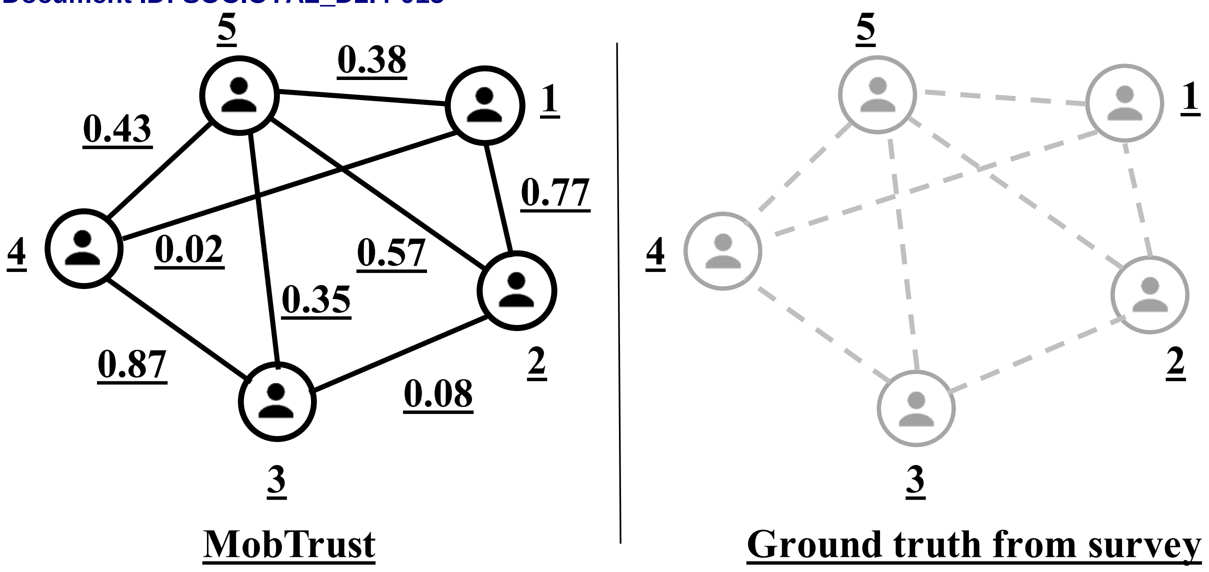


Figure 30. This figure depicts the real-world social graph inferred by the system (left side) and social graph extracted from the survey provided to the participants (left side).

As shown in Figure 30, the system was able to identify all social relations among the participants. Two pairs of participants did not establish a relation as they did not have any prior knowledge about each other and did not interact during the experiment. Thus, in both social graphs the relations (1,3) and (2,4) do not exist. This lack of social relation was correctly identified by the system. The system classified all detected social relations as Social, which was verified by the questionnaires the participants filled in.

For the identified social relations, the system provides also the confidence of the estimation. It should be noted that the social relation does not include temporal factors in the inference. However, a longer duration of interaction indicates an increased number of sample data leveraged for the estimation. The relations among (1,2) and (3,4) are the most confident as during the experiment they interacted for the longest time, allowing the system to include a larger amount of samples for the estimation. For the relations (1,5), (2,5), (3,5) and (4,5) the estimations showed confidence between 0.35 and 0.57 where the participants interacted for 10-15min. The least confident estimations of the social relations are identified on (1,4) and (2,3) where the participants interacted for less than 5min. In overall, the system was able to identify correctly the social relations of the participants and provide the relevant confidence factors.

Section 8 - Context Manager

SocIoTal (centralized) Context Manager acts as one of the core blocks of integration within SocIoTal Integrated Platform. It was initially introduced in WP2 [26] and fully described in WP3 [27] As shown in Figure 31, it provides the SocIoTal entities directory and storage for its current context information plus a complete NGSI9 and NGSI10 API rest interfaces, detailed (and updated) in SocIoTal Wiki [24] and extended with user functionalities in order to provide access and manage all this set of information.

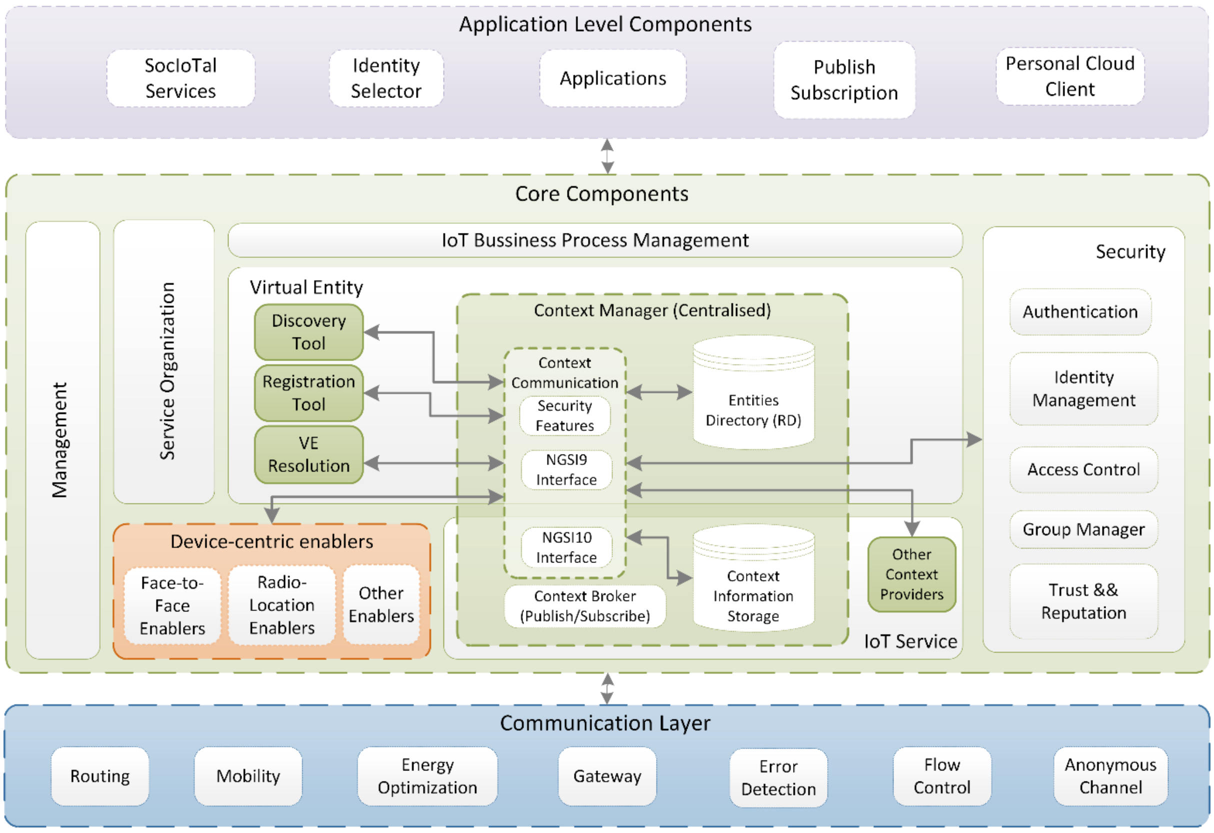


Figure 31: SocIoTal Framework. Centralized Context Manager architecture

Besides managing all context information related to entities, it also provides support to Trust & Reputation (section 7.1) and bubbles creation [27], keeping and supplying information related to scores and attributes. SocIoTal Context Manager is also integrated with the SocIoTal Security Framework by evaluating the Capability-Token and supports communities' management through the corresponding community token.

8.1 Implementation and integration

8.1.1 Capability-Token support

Access to SocIoTal Context Manager (Version 3 deployed) is secured by the SocIoTal Security Framework. Integration between these two SocIoTal components is performed through the Capability-Token and the Capability Evaluator, as introduced in D2.2 and described in Section

5. The Capability-Token, obtained from the Capability Manager, contains all the required credentials of the requestor identity plus the action requested. As shown in SocIoTal WiKi [24], to call SocIoTal Context Manager V3 methods (either, NGSI9, NGSI10 or EXTENDED interfaces) a "Capability-Token" header is required. This header will contain the corresponding capability token to be checked by the capability evaluator, through the libraries included in the Context Manager V3 deployment.

In order to authenticate the identity of the requestor included in the capability token, to alternative ways have been integrated in SocIoTal Context Manager V3 (further detailed in section 4):

- Through User Certificate: using the PKI-based authentication (section 4.2), the user will attach the "Signature" header, including the corresponding signature linked to its certificate. This signature will be captured and passed (by the SocIoTal CM) to the capability evaluator library that will execute the authentication process.
- Through IdM Authentication-Token: the SocIoTal IdM, using its Keyrock core, will provide an authentication token, linked to the user identity. This token will be provided to the SocIoTal CM in the Auth-Token header, and the user requestor id in the Client-Id header. These two parameters will be later used by the capability evaluator library to authenticate, against the IdM Keyrock, the user identity.

Once the capability evaluator library checks both, the supplied capability token plus the user authentication, will authorize or deny (including the corresponding rejection message) the action requested. Then, the SocIoTal Context Manager will execute the action (if allowed) or return the rejection message to the requestor application (when action is denied).

8.1.2 Community-Token support

A SocIoTal Community builds a closed environment where only registered users and entities can share information, representing a group of users and resources with a common objective or inquisitiveness. The SocIoTal component that manages communities and users/entities memberships is the Communities Manager. Details of SocIoTal Communities and the Communities Manager are shown in D3.2.2 [23].
The integration with communities' management is done through the corresponding Community-Token. The whole process and different options are detailed in D3.2.2, section 3.2.2 [23]. Supposing an already registered user, also member of a community, this user can request a community-token that identifies (and authenticates) this user as member of this community to the SocIoTal Communities Manager. The methods to achieve this are detailed in the SocIoTal WiKi [24]. The retrieved UUID (Universally Unique Identifier) of the generated Community-Token can be added now, as the "Community-Token" header, to every request (update, discover, create or subscribe) addressed to the Context Manager. SocIoTal Context Manager supports two different operation modes related to Communities:

- With no "Community-Token" header: the request will be addressed to a default community within a default domain. This is, all resources are assumed to belong to the same community. There will be no users/roles (from the point of view of communities) filtering and only SocIoTal security framework policies and restrictions (through the required Capability-Token) will apply. This could be useful when an instance of SocIoTal is going to be used for a single and well defined purpose, such as the platform for a particular application.
- When "Community-Token" is attached to the request. This token will be validated by the Communities Manager and info related to it will be retrieved to the Context Broker (as mentioned in the section above for API endpoints). The Context Broker here will apply the corresponding communities' restrictions/actions required, depending on the requested method and the info obtained from the validated Community-Token.

Section 9 - Conclusion

Based on the Architectural Reference Model of the IoT-A project, the SocIoTal security framework consists of components that support group management, secure group communications, key management, authentication, identity management, authorization, as well as trust and reputation management.

- The Group Manager (Section 2) facilitates creation of secure groups or bubbles using CP-ABE.
- The Key Management component (Section 3) provides the mechanisms and protocols for pre-distributing cryptographic keys and establishing secure channels between IoT devices.
- The Authentication component (Section 4) supports the conventional password-based and PKI-based authentication mechanisms through the Keyrock IdM. Additionally, it supports Idemix-style anonymous credentials (see deliverable D2.1), facilitating anonymous user participation in the social IoT.
- The Identity Management component (Section 5) is based on the Idemix design, and has been shown to provide scalable performance in terms of the number of attributes in an identity proving operation. Both the Authentication and Identity Management components are still under development.
- The Authorization component (Section 6), also called TAcIoT, checks and enforces users' authorization by implementing the SocIoTal trust model defined in deliverable D2.3. Both constrained and unconstrained devices are catered for. Development work is done, and performance results show that the various stages of computation have only sub-second overhead.
- The Trust and Reputation Management component (Section 7) evaluates trust/reputation based on users' location, and gait characteristics. The location-based approach utilizes hidden Markov model to infer users' location from RSSI measurements, whereas the gait-based approach utilizes kernel principal component analysis to infer whether the phone carrier is the phone owner.
- The Context Manager (Section 8) implements the OMA specifications, providing an interface compliant with NGSI9 and NGSI10. The component has been implemented to completion, and can now serve the Authorization and Trust & Reputation Management components.

The use case in Section 1 provides an example of how these components work together to support secure and privacy-preserving interactions in the IoT.

As of writing, integration exists between some of the components (Authorization, Context Manager, Trust & Reputation Manager). Complete integration of the components is on-going.

Section 10 - References

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

[2] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. In Security and Privacy, 2007. SP'07. IEEE Symposium on, pages 321–334. IEEE, 2007.

[3] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil pairing. In Advances in Cryptology—CRYPTO 2001, pages 213–229. Springer, 2001.

[4] J. Camenisch and E. Van Herreweghen, "Design and implementation of the idemix anonymous credential system", Proceedings of the 9th ACM conference on Computer and communications security (CCS '02), Vijay Atluri (Ed.). ACM, pp. 21-30, 2002,

[5] A. Pérez, G. López, O. Cánovas, and A.F. Gómez-Skarmeta. "Formal description of the SWIFT identity management framework". Future Generation Computer Systems, 27(8):1113-1123, 2011.

[6] C. O'Reilly, et al., "Reputation and Trust Management", Deliverable D2.3 of SocIoTal, Sept. 2015.

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

[8] J. Reason, *Human Error*. Cambridge University Press, 1990.

[9] [H. Wang and B. Wellman, "Social connectivity in america: Changes in adult friendship network size from 2002 to 2007," *American Behavioral Scientist*, vol. 53, no. 8, pp. 1148–1169, 2010.

[10] T. Choudhury and, A. Pentland, "Sensing and modeling human networks using the sociometer," *Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings.*, pp. 216–222, 2003.

[11] A. Antoniou, E. Theodoridis, I. Chatzigiannakis, and G. Mylonas, "Monitoring physical space using mobile phones for inferring social and contextual interactions," *2011 IEEE SENSORS Proceedings*, pp. 1616– 1619, Oct. 2011.

[12] J. Stehle ́, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J.-F. Pinton, M. Quaggiotto, W. Van den Broeck, C. Re ́gis, B. Lina, and P. Vanhems, "High-resolution measurements of face-to-face contact patterns in a primary school." *PloS one*, vol. 6, no. 8, p. e23176, Jan. 2011.

[13] N. Palaghias, S. A. Hoseinitabatabaei, M. Nati, A. Gluhak, and K. Moessner, "Accurate detection of real-world social interactions with smartphones," in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 579–585.

[14] E. Hall, *The hidden dimension*. New York, NY: Anchor Books, 1969.

[15] D. H. McKnight, L. L. Cummings, and N. L. Chervany, "Initial trust formation in new organizational relationships," *The Academy of Management Review*, vol. 23, no. 3, pp. pp. 473–490, 1998.

[16] N. Palaghias, N. Loumis, S. Georgoulas, K. Moessner, “Quantifying trust relationships based on real-world social interactions”, in 2016 *IEEE International Conference on Communications (ICC),* May 2016 (to appear)

[17] IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std 802.15.4™-2003, October 2003.

[18] IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE Std 802.15.4™-2006, September 2006.

[19] “Secure Group Communication”, Deliverable D3.3 of the SocIoTal project, May 2015

[20] “SOCIOTAL evaluation”, Deliverable D5.2 of the SocIoTal project, November 2015

[21] “Secure Group Communication”, Deliverable D3.3 of the SocIoTal project, May 2015

[22] https://github.com/sociotal/SOCIOTAL/wiki/SocIoTal-Communities-Manager

[23] "Privacy-aware context-sensing information-exchange", Deliverable 3.2.2 of the SocIoTal project, February 2015

[24] https://github.com/sociotal/SOCIOTAL/wiki

[25] “Reputation and Trust Management”, Deliverable 2.3 of the SocIoTal project, August 2015

[26] “IoT Communities and Identity Management”, Deliverable 2.1 of the SocIoTal project, August 2015

[27] “Privacy-aware context-sensing device discovery”, Deliverable 3.2.1 of the SocIoTal project, December 2015

[28] “Beta release of integrated SocIoTal platform “, Deliverable 4.3 of the SocIoTal project, February 2016

[29] De Caro, A., & Iovino, V. (2011, June). jPBC: Java pairing based cryptography. In *Computers and Communications (ISCC), 2011 IEEE Symposium on* (pp. 850-855). IEEE.

[30] https://sociotal.inf.um.es:8443/AttributeAuthorityServlet/AttributeAuthority

[31] “Framework Specification for Privacy and Access Control”, Deliverable 2.2 of the SocIoTal project, December 2014

[32] Rada-Vilela J (2014) Fuzzylite: a fuzzy logic control library. http://www.fuzzylite.com

[33] IEEE Standard for Local and Metropolitan Area Networks—802.15.4: Low Rate Wireless Personal Area Networks (LR-WPANs), IEEE standard 802.15.4-2011, Sep. 2011.

[34] IEEE Standard for Local and Metropolitan Area Networks—802.15.4: Low Rate Wireless Personal Area Networks, MAC Sub-Layer, IEEE Standard 802.15.4e-2012, Amendment to IEEE Standard 802.15.4-2011, Apr. 2012.

[35] N. Sastry and D. Wagner, “Security considerations for IEEE 802.15.4 networks,” in Proc. 3rd ACMWorkshopWireless Secur., 2004, pp. 32–34.

[36] “SOCIOTAL scenarios and requirements definition report”, Deliverable 1.1 of the SocIoTal Project, February 2014

[37] “Updated version of API specification”, Deliverable 1.3.2 of the SocIoTal Project, August 2015