Specific Targeted Research Projects (STReP)

SOCIOTAL

Creating a socially aware citizen-centric Internet of Things

FP7 Contract Number: 609112

WP3 Privacy-aware communication

Deliverable report

Contractual date of delivery:	31/05/15
Actual submission date:	

Deliverable ID:	**D3.3**
Deliverable Title:	**Secure Group Communication**
Responsible beneficiary:	UMU
Contributing beneficiaries:	
Estimated Indicative Person Months:	

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

Revision:	
Dissemination Level:	Public

Document Information

Document ID:	D3.3
Version:	0.6
Version Date:	05/06/2015
Authors:	José Luis Hernández Ramos, Jorge Bernal Bernabé, Antonio Skarmeta Gómez (UMU), Ignacio Elicegui Maestro (UC), Carmen López (UC), Michele Nati (UNIS), Christine Hennebert, Benoît Denis, Iulia Tunaru (CEA)
Security:	**Confidential**

Approvals

	Name	Organization	Date	Visa
Project Management Team	Klaus Moessner	UNIS		

Document history

Revision	Date	Modification	Authors
0.1	03/03/2015	First ToC	UMU
0.2	13/03/2015	Updated ToC, first responsibilities assignments	UMU, UC, UNIS, CEA
0.3	08/05/2015	Contributions and improvements added to different sections	UMU, UC, UNIS, CEA
0.4	14/05/2015	Contributions and improvements added to different sections	UMU, UC, UNIS, CEA
0.5	25/05/2015	Contributions and improvements added to different sections. Document ready for internal review	UMU, UC, UNIS, CEA
0.6	05/06/2015	Comments from internal review by UME and DNET were addressed, and the document is ready for delivery	UMU, UC, UNIS, CEA

Content

Executive summary

Description of the deliverable content and purpose

The main goal of SocIoTal is to foster the creation of a socially aware citizen-centric Internet of Things. For this purpose, SocIoTal is putting significant emphasis on the design and development of security and privacy-preserving mechanisms suitable to cope with the inherent requirements of IoT, in order to reach a full acceptance of the different stakeholders involved. In fact, nowadays, security and privacy implications are being considered as one of the main obstacles for a large scale adoption of IoT, since citizens will only accept these deployments if they are based on secure, trustworthy and privacy-preserving infrastructures. Thus, the application of these mechanisms is intended to encourage the adoption of value and innovative services to be used by the society in future smart cities.

Under the main foundations that are being envisioned and developed within SocIoTal, as part of the SocIoTal architecture developed in WP1, WP2 has already provided a security framework, which is compliant with the *Architectural Reference Model* (ARM) given by the IoT-A project [60], as well as different mechanisms to address security and privacy requirements for IoT scenarios. Moreover, the purpose of WP3 is to develop a context-aware framework using different *enablers* (e.g., *Face-to-Face* (F2F) or indoor location) based on the results of WP1 and WP2. Specifically, as part of WP3, the task 3.3 aims to explore low-level cryptographic schemes that allow a secure communication among groups of smart objects (i.e., communities and bubbles). This document provides the design and evaluation of the different mechanisms developed within this task, in order to enable groups communications adaptive to the context in which they take place. Additionally, these mechanisms are framed within the security framework that has been designed as part of WP2, which is intended to provide a holistic approach to address different security and privacy challenges in IoT.

Thus, **Section 1** of this document explores different low-level encryption techniques to enable secure communications between groups of entities. Specifically, different symmetric encryption schemes and mechanisms for the generation and distribution of these keys are presented. Additionally, this section presents *Identity-Based Encryption* (IBE) and *Attribute-Based Encryption* (ABE), as the most prevalent schemes based on certificateless public key cryptography. In particular, the description of the *Ciphertext-Policy Attribute-Based Encryption* (CP-ABE) is presented as a promising approach due to its flexibility and expressiveness. **Section 2** presents the main foundations that are leveraged by SocIoTal in order to manage secure group communications. Specifically, different alternatives for generation, distribution and use of cryptographic keys are presented, by considering the techniques previously presented. Additionally, the application of context awareness to these communications is also discussed.

Section 3 aims to show the main interactions between the security functional components of the security framework, in order to realize the secure group communication mechanisms for SocIoTal. The design of symmetric group keys generation and delivery approaches, as well as their application in the SocIoTal security framework are presented. The main interactions of the framework for the design of a context-aware CP-ABE based group secure communication mechanism is detailed. Furthermore, **Section 4** provides a description of the application of these mechanisms for secure communications among bubbles and communities of smart objects. Starting from the definition of different types of communities and bubbles, this section provides a detailed description of the different processes to communicate and share information, by making use of the alternative encryption techniques. Finally, **Section 5** provides a set of experimental results of such mechanisms. Moreover, the process to generate and deliver group symmetric keys is evaluated. This section also provides an evaluation of the process to communicate information by making use of the CP-ABE scheme. Specifically, these evaluations have been carried out by using the SocIoTal Context Manager, in order to share information with groups of entities in a secure way, through the well-known publish-subscribe pattern. Finally, **Section 6** concludes the document.

Section 1 - State of the art

The realization of the IoT ecosystem imposes significant restrictions on security and privacy, since physical objects are being seamlessly integrated into the Internet infrastructure, through the use of different wireless communication technologies. In particular, IoT scenarios are intended to manage particularly sensitive data, and consequently, any information leakage could seriously damage the user privacy. This problem is exacerbated in the IoT, since any smart object will be able to create new information and communicate it to any other entity. Consequently, the requirements presented by common IoT scenarios require more flexible data sharing models between entities while the privacy of smart objects involved is still preserved. Unlike the current Internet, IoT interaction patterns are often based on short and volatile associations between entities without a previously established trust link. These mechanisms should be applied on the smart objects themselves in order to provide an end-to-end secure data dissemination. Furthermore, given the pervasive, dynamic and distributed nature of IoT, it is necessary to consider more flexible data-sharing models in which some information can be shared with a group of consumers or a set of unknown receivers and, therefore, not addressable a priori. In this section, we review existing cryptographic schemes that can be potentially used to develop secure information sharing mechanisms for the IoT.

1.1 Secure Group Communication for the IoT

Symmetric keys are used to ensure the security of data during a session or a communication. A negotiation protocol is usually started prior to authenticate devices with a certificate and a pair of asymmetric keys. This protocol resulted in the generation and delivery of the symmetric session key. Currently, standardized security protocols for sensor networks with constrained resources, only address the connection peer-to-peer, at the network layer (compressed IPsec) or below the application layer (*Datagram Transport Layer Security* (DTLS)). Group management and data exchange from one to many is not addressed by the different standards.

OSCAR protocol [61] addresses this theme to bring the cluster security with flexibility. But it fits on top of the application layer, therefore asks the developer to know the intricacies of security, which is both risky and not acceptable.

Thereafter, several security management techniques for groups are detailed for centralized and distributed networks. In the paragraph 3.1 we show how a variant of the Iolus algorithm [1] can be adapted to personal bubbles and communities of SocIoTal to ensure the security of data exchange in a group of heterogeneous devices, at the network layer of the OSI model, and therefore transparent to users and application developers.

1.1.1 Key Distribution methods for WSN

Secure group communication is crucial for building distributed applications that work in dynamic network environments. Key management is the base for providing common security services. Several mechanisms have been introduced for the group key establishment (see Figure 1). They can be classified into two main categories. First, the distributary approach is based on an element called *Group Controller and Key Server* (GCKS) that is responsible of the generation and the distribution of the group key. This approach can be sub-divided into two sub-categories: the centralized and the decentralized techniques. In a centralized key establishment protocol, the key server is the only element responsible of the group key generation and distribution. In a decentralized protocol, a hierarchy of key servers shares the task of distributing the group keys to the members. For that, sub-groups including a leader node are formed. The renewal of keys can be launched when a member joins or leaves the group or, at periodical times. Second, the contributory approach is based on an equal involvement of all the participant of the group to generate the shared key.

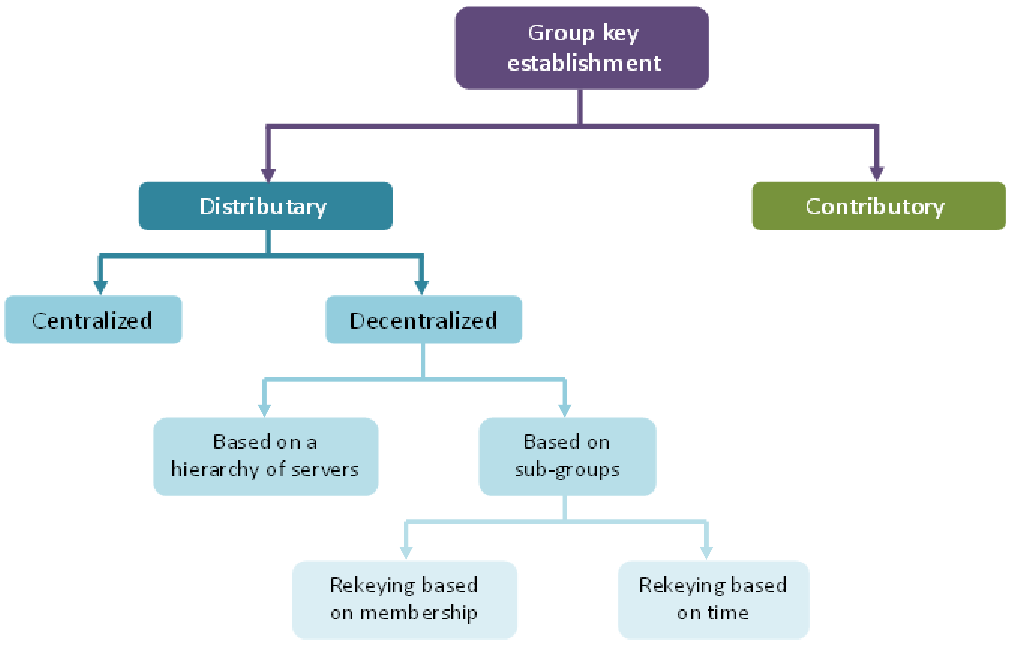

Figure 1. Group key establishment categories

The members in a group may change frequently. When a new member joins the group, a new group key should be generated and distributed to all the nodes. This avoids a new member to get the previous group key and to access to information previously exchanged by the group. This ensures the foreword security. Conversely, when a member leaves a group or is evicted, it has no more right to know the information exchanged by multicast over the group. This ensures the backward security. This is called the one-affects-all phenomena.

1.1.1.1 Decentralized approach based on subgroups

The decentralized group key management approach based on sub-groups consists in the division of the group into many sub-groups. The security of the group is ensured with a group key shared between all the members of the group, whereas all sub-groups shared their own sub-group key material. This approach needs less bandwidth than the centralized approach for the rekeying. It is more suited for the constrained devices. In the following, the protocols are classified into membership-driven protocols and time-driven protocols.

1.1.1.1.1 Membership-driven re-keying

This approach implies that the group key is changed at each time a member joins or leaves the group.

<u>Iolus protocol</u>

Iolus [1] is a framework of hierarchy of multicast sub-groups. The overall sub-groups form a virtual multicast group. Each sub-group is managed by a Group Security Agent (GSA) responsible of the key management inside the sub-group. A main Group Security Controller (GSC) manages the top-level sub-groups.
Each sub-group uses its own group key when a membership change occurs, and only that sub-group is involved in the re-keying process. By this way, large groups are scaled into sub-groups and the one-to-all phenomena is mitigated.

Join procedure:

1. The member who wants to join the network locates the appropriate GSA and sends a join request to it using a **unicast** secure channel.
2. The GSA checks its database to decide if it approve or deny the request. If it approves, the GSA generates a new secret called $K_{GSA\text{-}MBR}$ as session key to send only to the new member via a **unicast** secure channel.
3. The GSA generates a new sub-group key $K1_{SGRP}$ and encrypts it with the previous sub-group key $K0_{SGRP}$ that is the current secret shared between the group members. The new encrypted sub-group key is then **multicast** to the current members.
4. The new sub-group key $K1_{SGRP}$ is encrypted with the session key $K_{GSA\text{-}MBR}$ and sent to the new member via a **unicast** secure channel.

Leave procedure:

1. The GSA generates a new secret called $K2_{SGRP}$ as session key.
2. As there is no common channel between the remainder members of the sub-group, the new sub-group key is encrypted with each session key of the remainder members and sent separately using a **unicast** secure channel.

Message exchange:
When a member wants to multicast a message to all the members of a group, the process is divided into two stages, the multicast of data and the multicast of key material:

<u>Multicast of data:</u>

1. The sender generates random keying material and encrypts the data with it.
2. The sender multicasts the encrypted data to its sub-group members and to its local GSA.
3. The local GSA transmits the encrypted data to others GSA.
4. The GSA multicast the encrypted data over in their sub-groups.

<u>Multicast of key material:</u>

1. The sender encrypts the key material with its session key and sends the encrypted message to its GSA.
2. The local GSA decrypts the message to get the key material, and re-encrypts it with the sub-group key it wants to address. Then, it multicasts the resulting message to the targeted sub-group.
3. The local GSA transmits the encrypted data to others GSA.
4. The targeted GSAs decrypt the incoming message and re-encrypt the key material with their current sub-group key. Then, it multicast the encrypted key material over in their sub-groups. By this way, all the members of the network can read the data.

A drawback of Iolus is that it affects the data path. The data going from one sub-group to another must be translated from one sub-group key to another one. That induces encryption/decryption operations and a delay that should be tolerated by the applications.

Iolus is an interesting scheme that could be considered with the focus of the SocIoTal framework architecture. It is not linked to a dedicated network topology. The Group Security Agent and Group Security Controller could be introduced in a relevant way to handle SocIoTal bubble and communities.

<u>KHIPS protocol</u>

Keyed Hierarchical multicast Protocol (KHIP) [2] is based on a multicast tree built using a routing protocol. It uses a different group keys for each branch of the tree. An authentication service uses the certification to authenticate the members and on-tree routers. The multicast tree is organized into sub-branches managed by a trusted router that deals with the Group Key (GK).

KHIP suffers of delays for the transmission induced by the successive encryption/decryption operations. Moreover, KHIP is based on a tree network topology which does not fit with the SocIoTal bubble or community better organized into stars with different granularity according to the branches.

SAKM protocol

Scalable and Adaptive Key Management (SAKM) [3] addresses the one-affects-all phenomena and the re-keing overheads using the dynamic of the group members. It tackles the scalability issue by organizing the multicast group into clusters with its own Group Key (GK). This organization is periodically updated depending on the dynamism of the members during the secure session. In fact, some parts of the network may be more dynamic than others during the same periods of time. This protocol restricts the re-keying to the areas with frequent membership changes.
With SAKM, a group can be divided into autonomous sub-groups. This notion is interesting for SocIoTal in order to introduce granularity in the management of the groups for a user personal bubble. It may enable modularity in the sharing of the user devices.

Scalable Multicast Key Distribution

This group key distribution method is based on Core Based Trees (CBT) architecture [3] and protocol (RFC1949 [4] and RFC2201 [5]). When a CBT group is created, the group initiator has the responsibility to create a multicast group access list (ACL). This list should be digitally signed. It is sent to the primary core. The ACL consists of group membership information (join or leave). It may be reasonable to restrict key distribution capabilities to trusted nodes (routers) of the network.

However, with SMKD, the forward secrecy is not ensured, as the leaving members can still decrypt the message content even when they have left the group.

Moreover, the key distribution is delegated to the node at the initiative of a group creation. This implies for this node to be a trusted element. It is a very strong constraint for the group management.

IGKM protocol

The protocol Intra-Domain Group Key Management [6] defines a multicast group as a domain divided into areas. Each area includes an Area Key Distributor (AKD) and a Domain Key Distributor (DKD) ensures the coordination between the areas.

This scheme is close to Iolus, but less flexible in its implementation.

1.1.1.1.2 Time-driven re-keying

Kronos

Kronos [7] is a scalable approach based on time-driven re-keying. The group is provided with a date of birth and a date of death. A key management framework is built upon, close to IGKM protocol, but in which, the AKD independently generate the same MKey synchronously, at the same time, and transmit it to the area members.

An issue of Kronos is to make the AKD generate the key synchronously by remote devices. A clock synchronization process should be used. Another issue is that the AKD must share some state information and the same key generating algorithm.

MARKS

Marks protocol [8] is based on a key sequence construction that uses a Binary Hash Tree (BHT). The BHT requires two blinding functions well-known by the nodes. For instance, the first well-known blinding function, called 'left' function b_0, could be a one bit left circular shift followed by a hash function, while the second blinding function, called 'right' function b_1, could be a one bit right circular shift followed by the same hash function. Alternatively, for efficiency, two variants of a hash function could be used with two different initialization vectors.

In SocIoTal architecture, the groups are not built as a tree structure and we cannot identify a parent and children for any devices. The BHT introduced is an interesting notion to consider the group management over a decentralized network. Unfortunately, the tree structure does not fit in SocIoTal topology.

1.1.1.2 Decentralized approach based on hierarchical servers

The decentralized approach based on hierarchical servers is generally used for network whose the topology is hierarchical. A hierarchy of secret keys is built according to the hierarchy of the elements of the network.

GKMP protocol

In Group Key Management Protocol (GKMP) [9] [10], the key server shares a secret key (KEK) with each active member (MB). For that, it generates a Group Key Packet (GKP) that contains two keys: a Group TEK (GTEK) and a Group KEK (GKEK). The GTEK is used to encrypt the traffic and the GKEK is used to secure for the management of the keys, as re-keying.

This protocol may be considered for socIoTal though less modular than Iolus.

LHK protocol

The basis for the LKH approach for scalable group rekeying [11] is a logical key tree structure maintained by the key server. The root of the key tree is the group key used for encrypting data in group communications and it is shared by all users. Each leaf node of the key tree is a secret key shared only between an individual user and the key server. The middle level keys, KEKs, are used to facilitate the distribution of the root key. Of all these keys, each user owns only those keys that lie on the path from its individual leaf node to the root of the key tree. As a result, when a user joins or leaves the group, all of the keys on its path have to be changed and re-distributed to maintain backward and forward data confidentiality. Note that all the keys in the key tree are randomly generated and there are no functional relationships among them.

OFT protocol

In the OFT protocol [12], the key server maintains a binary key tree where each key in the tree is derived from its child key. Each node u holds a node secret S_u and a node key K_u. The node key is derived from the node secret thanks to a hash function:

$$K_u = \text{hash}(S_u)$$

The parent node secret is derived from the keys of the left child and the right child.

$$S_{parent} = \text{hash}(K_{left\_child} ; K_{right\_child})$$

Each member knows the node secrets on the path from its node to the root (and therefore the node keys along this path), and the blinded node secrets that are siblings to this path, and no other node secrets nor node keys.

1.1.2 Physical layer symmetric key generation and related single-link applications

As explained in the previous section, group keys, which are used for multicast confidential communications, can be generated and distributed by relying on initial unicast secure channels. In order to achieve these initial secure unicast channels, it is often assumed that initial symmetric keys are already distributed between the concerned parties (i.e., usually a regular node and a server node). In the following sections, an alternative approach to symmetric key generation and agreement based on the physical properties of the wireless channels is discussed. The keys generated in this manner can be used either for directly encrypting the communications in unicast communications or for distributing a group key that will later serve for multicast secure communications. Moreover, in dynamic scenarios, when the radio channel changes, these keys can and should be refreshed regularly using the same procedure as the first time.

Main key generation models

Secret key generation based on the physical layer in wireless communications is a particular case of information theoretic secret generation, which is a more general framework and consists of two main models: the source and the channel models [13]. The source model implies the existence of a source of randomness (modelled by a discrete memoryless source, DMS) observed by all the participants (including the eavesdropper) with their corresponding observation noise. The secret key is generated from this source of entropy by quantization, meant to generate binary flows, and public discussion, meant to i) create an advantage of the legitimate users with respect to the eavesdropper and ii) correct possible mismatches between the legitimate observations. This model is related to information theoretic studies on source coding with side information.

In the case of physical layer key generation, the source is the wireless channel probed by the participants (see further details in next subsection). The alternative channel model for key generation is based upon a discrete memoryless channel model (DMC) used to share a locally generated (by one of the parties) random sequence, which will serve as a key. This model also uses a public discussion channel with the same objectives as before. Key sharing over a wireless unsecure channel is possible due to the different fading and noise realisations between the legitimate and illegitimate links.

Source model

Due to the unpredictable fading realizations and to the reciprocity of the propagation of electromagnetic waves, the wireless channel between two legitimate users represents a common source of randomness that can be exploited to separately generate a secret key. Any eavesdropper, situated in a sufficiently distant position with respect to the legitimate users, observes a decorrelated channel and, therefore, will not be able to generate the same key. A typical point-to-point sequential key generation algorithm consists of:

- *Randomness sharing,* which corresponds to channel probing (Figure 2);
- Advantage distillation, an optional step that aims at selecting the channel probes for which the legitimate users consider they have an advantage with respect to an eavesdropper (not represented in Figure 2); this step is optional because the legitimate

users already present an intrinsic advantage with respect to the eavesdropper thanks to the reciprocity and the spatial decorrelation properties.

- *Information reconciliation* meant to correct the mismatches due to asymmetric equipment, noise, interferences or half-duplex communications by using exchanges over a public channel; this step is usually preceded or jointly implemented with a *quantization* phase, which transforms values issued from channel measurements into binary flows (Figure 2);
- *Privacy amplification* (e.g., hash functions or randomness extractors), a deterministic independent processing of the common bit sequences in order to generate a secure secret-key by "compensating" the information leakage on the public channel (Figure 2).

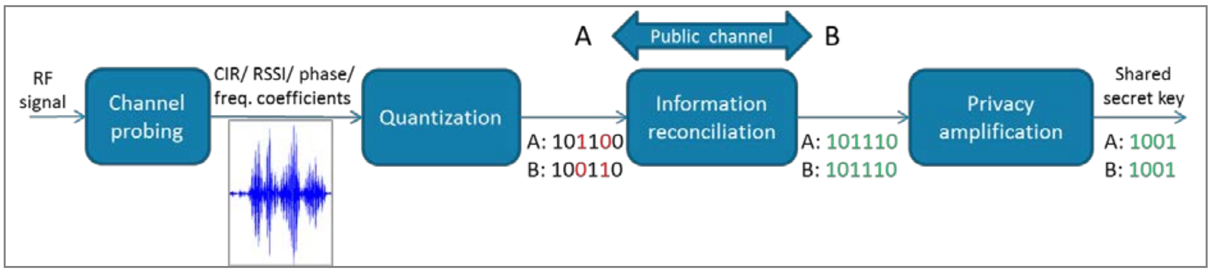

Figure 2. Physical layer key generation phases

Channel probing

Key generation schemes can be classified into scalar (i.e., each channel probe consists of one sample) or vectorized schemes (each channel probe provides several values for quantization). In both cases, channel probing is done several times until the desired number of samples is obtained. The number of samples directly conditions the length of the final key. Vectorized measurements are preferable due to their advantage in terms of entropy. Received signal strength (RSS) is one of the main metrics for scalar key generation schemes because it is widely available in numerous wireless devices [14]. Other commonly employed channel metrics depend on the wireless technology and are obtained after an estimation phase:

- the magnitude or phase information [15] of the channel in narrow band devices (usually scalar measurements because of flat fading); one drawback of key generation using phase lies in the difficulty of achieving phase synchronization between the two parties.
- multiple independent channel coefficients in wideband communications (e.g., frequency diversity in OFDM [16]);
- multiple channel coefficients in MIMO systems [17] (i.e., spatial diversity); in the case of MIMO systems, the required wireless equipment is less compatible with low-cost devices and a certain separation distance between antennas is required to obtain independent samples;
- channel impulse responses (CIR) in ultra-wideband communications (UWB) where the additional dimension of variability can be the excess delay of multipath components [19], [18].

For instance, the IR-UWB technology is particularly suitable for physical layer key generation due to its high multipath resolution capability, which gives access to entropy-rich signals (i.e., several samples at different excess delays per channel probe). Moreover, the samples can be considered as independent due to the large bandwidth. Although relatively long key lengths can be obtained when using IR-UWB metrological test-beds [19], the challenge of implementing UWB key generation in low-complexity devices remains the estimation of the CIR. Sparse channel estimation methods, which are currently studied (e.g., compressed sensing / finite rate of innovation methods), can offer a possible solution to this channel estimation problem. Otherwise, CIR estimations with limited dynamics and temporal resolution

[20] or other CIR-related metrics (e.g., signals issued from the integration of IR-UWB convolved channel responses on the duration of the probing pulse) could be employed.

Besides the available physical layer, the choice of the metric depends also on the possible imperfect reciprocity caused by implementation considerations. The different front-ends (amplifiers, filters, etc.) of the legitimate users' devices could have a detrimental impact on the correlation of the reciprocal channels.

Similarly, accurate time synchronization between the legitimate users is a critical issue that could reduce the correlation between their observations. Therefore, depending on the available technologies in the adopted devices, as well as their accuracy, some channel characteristics should be preferably adopted to generate the key. For example, if an accurate time reference (e.g., from GPS localization services) is available at both legitimate users, it could be used to synchronize their key generation algorithms. In this case, even rapidly varying channel properties (e.g., phase rotation introduced by the channel or the CIR) could be used to generate the key. On the contrary, when no accurate time reference is available, channel dependent properties less sensitive to synchronization errors (e.g., average RSS) should be preferred. Of course, the secure character of the key depends primarily on the “randomness” of the observed phenomenon.

Quantization

Once a vector of channel measurements has been obtained, a quantization algorithm/rule must be applied to convert the values into bits. In the classical sense (e.g., for digital signal compression and/or reconstruction), the term “quantization” refers to the mapping of a continuous set of values into a discrete alphabet (the quantized values) and its binary representation (the codewords). The classical quantization problem is defined by the rate-distortion pair, which describes the trade-off between two conflicting goals: keeping both the rate and the distortion as small as possible. Application of the quantization tools in the key generation context has to be adapted to its specific goals.

First of all, an efficient quantization algorithm should be able to generate as many bits as possible. Secondly, a robust quantization algorithm should generate keys with good reciprocity (i.e., low distortion between the keys generated at the two ends). One option for measuring discrete distortions is the Hamming distance. Finally, key bits should have a random aspect, which can be tested by applying statistical tests for random and pseudo-random number generators (e.g., bit frequency test, pattern evaluation, frequency of oscillations, approximate entropy test [21]). Otherwise, from a design point of view, the frequency of apparition of codewords (i.e., the diversity) can be evaluated and quantization schemes can be adapted depending on the application needs and the desired trade-off between length, reciprocity and random character [18].

Information reconciliation

The information reconciliation phase can be achieved in several ways:

- quantization (in the classical sense) of the measurements and transmission of the difference between the measured and the quantized values over the public channel;
- distributed source coding approach: map the quantized bits to a trellis or a Hamming codebook and exchange the corresponding syndromes over the public channel; the same method can be enforced by prior Low Density Parity Check (LDPC) decoding for better key agreement [22].
- error-correction codes (e.g., Reed-Solomon, LDPC etc.) / Cascade protocol involving iterative exchanges of parity information.

Other more complex options for key reconciliation include compressive sensing algorithms, if we consider the difference between the users' quantized sequences as a sparse vector or neural networks with specific training algorithms corresponding to typical bit errors.

1.1.3 Extension to cooperative scenarios for physical layer multi-link pairwise keys or group keys generation

The aforementioned methods provide a symmetric key shared by only two nodes (i.e., pairwise key) and generated from a single communication channel (i.e., single link). Both source and channel models for key generation using the physical layer in wireless communications have been extended to cooperative (i.e., involving several nodes) or multi-link (i.e., involving multiple links) scenarios with the intent of: i) generating more robust (still) pairwise keys with the help of other nodes (i.e., relays) or ii) generating a group key.

In [23] and [25], the secret key between two user nodes is generated with the help of a relay. First, pairwise keys (from the main channel and the side channels between the nodes and the relay) are generated from the channel properties (the source model) using a typical key distillation procedure based on channel gains in [23] or phase estimations in [25]. After that, the key between the user nodes is "enriched" by the public diffusion of a bitwise combination of the keys obtained from the side channels. The user nodes can therefore recover the key that was generated by the other user node and the relay and append one of the side channel keys to the final key (Figure 3).

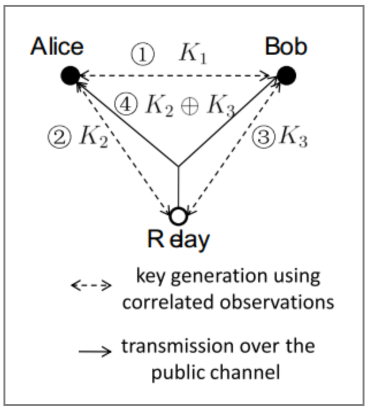
Alice
① K_1
Bob
④ $K_2 \oplus K_3$
② K_2
③ K_3
Relay
key generation using correlated observations
transmission over the public channel

Figure 3. Cooperative physical layer key generation [23]

A similar approach starting from pairwise physical layer keys is taken in [26], where the received signal strength is quantized for pairwise key generation and a group key, generated by a root node, is distributed in the network.

In [27], the authors present another extension of the source model to cooperative pairwise key agreement and group key generation in a pairwise independent network (i.e., a network in which the point-to-point channels are independent, which is a common assumption for most wireless networks). Like before, pairwise keys are generated on each physical network link and the group keys (or extra secret bits for pairwise cooperative keys) are propagated based on one-time padding operations over a graph representation of the network.

In order to deal with the issue of limited entropy of the source model (i.e., if the channels are static there might not be enough information to harvest in order to generate a robust secret key), [24] extends the channel model for key generation to a cooperative scenario (with a relay). The authors derive the upper and lower bound for the secret key rate with a relay and propose a joint optimized design of the different key generation phases (advantage distillation, information reconciliation and privacy amplification) regarding the trade-off between security and protocol efficiency.

1.2 Secure Data Dissemination

The application of *Symmetric Key Cryptography* (SKC) has been traditionally considered as cryptographic solution for secure group communications. Indeed, due to the heterogeneous nature of IoT environments, where constrained devices can communicate each other, SKC is being currently considered as the cryptographic solution for emerging scenarios. However, these approaches suffer from well-known drawbacks related to a cumbersome key management and distribution, and consequently, they are not able to provide desirable scalability properties for managing the security of billions of smart objects interacting each other. In contrast, *Public Key Cryptography* (PKC) presents significant requirements on computing and memory resources. This issue has fostered the development of recent research initiatives in order to make the application of PKC on IoT scenarios feasible through several optimizations on *Elliptic Curve Cryptography* (ECC) [31]. Furthermore, traditional PKC-based solutions only allow a specific piece of information to be shared with a certain data consumer. In addition, the use of certificates in scenarios with a potentially large number of interacting entities can be cumbersome because of the need for validation and revocation mechanisms.

In recent years, the application of *Certificateless Public Key Cryptography* [28] to Future Internet scenarios is receiving significant attention from the research community [29] [30]. Indeed, in order to address already mentioned issues, Identity-Based Encryption (IBE) [32] was proposed as a certificate-less public key scheme alternative, in which the identity of an entity can be associated to a public feature of it (e.g. the email address), represented as a character string. Consequently, smart objects could employ public features to share information in a secure way, without the need to obtain the certificate of other entities that are usually stored and supported by a *Public Key Infrastructure* (PKI). This is a relevant feature to be leveraged on IoT scenarios, in which ephemeral transactions will be common. In this direction, *Attribute-Based Encryption* (ABE) [34] represents the generalization of IBE, in which the identity of the participants is not represented by a single string, but by a set of attributes related to their identity. In both schemes, cryptographic keys are managed by a *Trusted Third Party* (TTP), usually called *Attribute Authority* (AA). However, while a TTP is required, these keys can be used for a secure *Machine-to-Machine* (M2M) communication between smart objects. ABE is gaining attention because of its high level of flexibility and expressiveness, compared to previous schemes. In ABE, a piece of information can be made accessible to a set of entities whose real, probably unknown identity, is based on a certain set of attributes. This represents a step forward in order to realize a privacy-preserving and secure data sharing scheme in pervasive and ubiquitous scenarios, since consumers do not need to reveal their true identity to obtain information, while producers can be sure that their data are accessed only by authorized entities. Based on ABE, two alternative approaches were proposed. In the *Key-Policy Attribute-Based Encryption* (KP-ABE) scheme [35], a ciphertext is encrypted under a set or list of attributes, while private keys of participants are associated with combinations or policies of attributes. In this case, a data producer has limited control over which entities can decrypt the content, being forced to rely on the AA entity issues appropriate keys for getting access to disseminated information. In contrast, in a CP-ABE scheme [36], a ciphertext is encrypted under a policy of attributes, while keys of participants are associated with sets of attributes. Thus, CP-ABE could be seen as a more intuitive way to apply the concepts of ABE; on the one hand, a producer can exert greater control over how the information is disseminated to other entities, On the other hand, a user's identity is intuitively reflected by a certain private key. Figure 4 presents a simple example showing the relationship between both schemes.

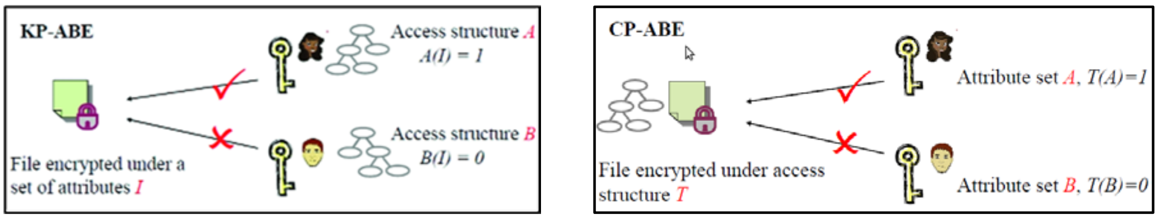

Figure 4. KP-ABE vs CP-ABE

As already mentioned, the application of certificateless public key cryptography on emerging scenarios is gaining attention in recent years. Indeed, some approaches have been recently proposed. In [37] a novel certificateless public key encryption scheme on the elliptic curve over the ring is presented. The security of such scheme is based on the hardness assumption of Bilinear Diffie-Hellman problem and factoring the large number as in a RSA protocol. The work proposed by [38] carries out a study of the feasibility of using concepts of Identity Based cryptosystems to solve privacy and security issues that arise during M2M communications in IoT, in which different drawbacks are highlighted related to performance aspects. [39] presents TIBC, a security solution based on different cryptographic systems addressing the public key replacement issue in identity-based cryptography. Moreover, [40] presents a cooperative approach for CP-ABE, by considering expensive cryptographic operations to be performed on a set of assisting nodes. Such approach is an alternative for the application of Certificateless Public Key Cryptography on IoT scenarios, where constrained devices are employed.

1.2.1 Symmetric-Key Cryptography (group keys)

Symmetric keys are used to ensure the security of data during a session or a communication. A negotiation protocol is usually started prior to authenticate devices with a certificate and a pair of asymmetric keys. This protocol resulted in the generation and delivery of the symmetric session key.
Currently, standardized security protocols for sensor networks with constrained resources, only address the connection peer-to-peer, at the network layer (compressed IPsec) or below the application layer (DTLS). Group management and data exchange from one to many is not addressed by the different standards.
OSCAR [61] protocol addresses this theme to bring the cluster security with flexibility. But it fits on top of the layer applicationet therefore asks the developer to know the intricacies of security, which is both risky and not acceptable.
Thereafter, several security management techniques for groups are detailed for centralized and distributed networks. At the paragraph 3.1 we show how a variant of the Iolus algorithm can be adapted to personal bubbles and communities of SocIoTal to ensure the security of data exchange in a group of heterogeneous devices, at the network layer of the OSI model, and therefore transparent to users and application developers.

1.2.2 IBE

The *Identity-based Cryptography* (IBC) [33] was initially proposed as a certificateless public key cryptography alternative, in which a piece of public information associated with an entity (e.g. its email address) is used as the public key. Under the main foundations of IBC, [32] proposed an *Identity-based Encryption* (IBE) scheme, which allows to encrypt a message under a character string, which is considered as the identity of the message's recipient. Consequently, unlike traditional public key cryptography based on a PKI infrastructure, in IBE, an entity does not need access to the recipient's certificate to send an encrypted message, simplifying key management tasks and reducing the overhead related to certificates transmission. In a typical IBE configuration, a central entity (i.e. a *Trusted Third Party* (TTP)) is responsible for generating private keys and sending them to the corresponding entities. This process of key generation requires an authentication mechanism, by which an entity demonstrates certain identity. Then, these keys can be used by entities for a secure communication without the need to involve the TTP. An IBE scheme usually consists of four main steps:

- **Setup** ($\lambda \rightarrow$ {params, MSK}). This algorithm is executed only once by the TTP. It takes a security parameter k as input, and generates a master secret key MSK, as well as a set of public parameters params of the system.

- **Extract** ({MSK, params, ID} $\rightarrow$ SKID). This phase is performed by the TTP when an entity requests a private key. It takes the master key MSK, the public parameter params, and a characters string ID as an input, generating a private key SK associated to the identity ID.

- **Encrypt** ({params, M, ID} $\rightarrow$ CT). This step is executed by an entity that wants to send an encrypted message M to an entity whose public identity is reflected by ID. The result of the algorithm is a ciphertext CT.

- **Decrypt** ({params, CT, SK} $\rightarrow$ M). Once the recipient gets CT, it tries to get M by using the public parameters params and its private key SKID. While IBE provides relevant features to be leveraged in different scenarios, in this work, it is used as one of the proposed alternatives, in order to enable an anonymous access control mechanism for IoT scenarios

As already mentioned, IBE provides a flexible cryptographic scheme, in which any public aspect of a smart object can be used to encrypt information. This feature can be leveraged in order to enable a secure group sharing mechanism by associating an IBE key to a specific role. In this way, the process to get this key could involve an additional procedure by which an entity proves a specific set of attributes, and the TTP associates such attributes to a particular role. For example, the set of Alice's smart objects in her office can be given with an IBE key associated to the role="AliceOffice". Consequently, these smart objects can share information in a secure way, by encrypting information with the ID = "AliceOffice". Only those devices with a proper SK will be able to decrypt the data being communicated.

1.2.3 CP-ABE

As already mentioned, in the *Ciphertext-Policy Attribute-Based Encryption* (CP-ABE) scheme [36], a ciphertext is encrypted under a policy of attributes, while keys of participants are associated with sets of attributes. In addition, CP-ABE is secure to collusion attacks, that is, different keys from their corresponding entities cannot be combined to create a more powerful decryption key. This feature is due to the use of individual random factors for each key generation. Moreover, in order to enable the application of CP-ABE on constrained environments, the scheme could be used in combination with SKC. Thus, a message would be protected with a symmetric key, which would be encrypted with CP-ABE under a specific policy.The main CP-ABE algorithms are described as:

- **Setup** ($\lambda \rightarrow$ {PP, MSK}). It takes an implicit security parameter λ as input. The algorithm generates the public parameters PP which are common to all users of the system (for example, the universe of attributes U) as well as a master secret key MSK which are used by the TTP to generate secret keys for participants.
- **Key Generation** (MSK, A} $\rightarrow$ SK_A). After an entity proves it has a certain set of attributes A, the algorithm takes the master key MSK and the set A as an input. The result is a private key SK_A.

- **Encrypt** ({PP, M, PT} →CT). It takes the message M, public parameters PP and a decryption policy PT representing subsets of attributes, which are allowed to decrypt M. The result of this algorithm is a ciphertext CT containing PT.
- **Decrypt** ({PP, CT, SK_A}→ M). The decryption algorithm takes as input the public parameters PP, the ciphertext CT with a PT associated, and a private key SK_A. If the set A satisfies the policy PT, the entity will be able to decrypt CT with SK_A.

Compared to IBE, CP-ABE provides a more expressive cryptographic scheme in order to enable a secure data communication among smart objects. In this case, devices can be enabled with CP-ABE keys associated to their identity attributes (e.g. contained in their X.509 certificates), which can be used to create a secure communication mechanism with groups of entities (i.e. communities and bubbles). However, unlike with IBE, the creation of a group does not require the generation of a new CP-ABE key, since the same key can be employed to participate in different bubbles or communities.

Section 2 - SocIoTal Secure Group Communication

This section presents the main foundations that are leveraged by SocIoTal in order to manage secure group communications. Namely, different alternatives for generation, distribution and use of cryptographic keys are presented, by considering the techniques introduced the previous section, mainly based on SKC and CP-ABE. Additionally, the application of context awareness to these communications is also discussed.

2.1 Symmetric Secret Key Generation Using the Physical Layer

Due to a patent application in progress, in this section, we will only present an initial solution for group key generation using the physical layer, which is inspired from state of the art articles like [26], [27] and then adapted to the context of Impulse Radio Ultra-Wideband (IR-UWB) capable nodes. As stated before, the IR-UWB technology is preferable to narrow band alternatives due to its capability of providing a richer signal representing the communication channel (i.e., channel impulse response/CIR vs. received signal strength/RSS). It should be noted that the developments associated with the patent application will be accounted in another upcoming SocIoTal deliverable.

In a mesh network, group key distribution using physical layer key generation methods can be achieved in two phases (similarly to [26], [27]):

- Each node generates a pairwise symmetric key with each of its neighbours based on the properties of the radio channel (i.e., the source model);
- A group key, generated by a lead node, is propagated in the mesh by one-time padding operations using single-link keys. The security of the scheme relies therefore on the single-link keys.

Radio technologies such as narrow band IEEE 802.15.4/Zigbee or IEEE 802.15.4a/IR-UWB are two of the candidates for the physical layer of IoT devices and can be exploited for symmetric key generation on each link of the mesh. The quantized channel metric can be either RSS measurements in narrow band technologies or estimated CIRs in IR-UWB devices. The quantization algorithm uses uniform quantization with guard-bands (i.e., the samples that fall in the guard-band near the edges of the quantization cells are not quantized and the dropped samples' indexes are shared over the public channel) and Gray encoding for the mapping of binary codes to the quantization cells.

In the case of IR-UWB capable nodes, the channel probing phase consists in the transmission of sequences of short pulses (with durations of the order of 1 ns, corresponding approximately to an equivalent bandwidth occupancy of 1 GHz, for instance centred at 4.5 GHz respecting the regulation emission mask). The received signal is a superposition of delayed pulses (i.e., multipath) according to the interactions of the transmitted signal with the physical environment (Figure 5.).

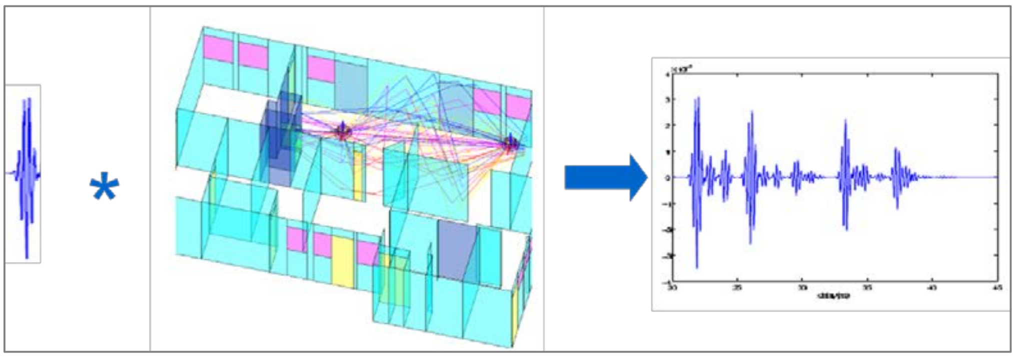

Figure 5. IR-UWB transmission principle: left (transmitted unitary pulse), middle (multipath between Tx and Rx in a typical indoor environment), right (received multipath profile at Rx)

Because of the large bandwidth and the high carrier frequency, the direct sampling of this kind of signals with non-metrological equipment is limited in dynamics and time resolution. Channel response estimation is very challenging given the state of the art algorithms, which need either high sampling frequencies (maximum likelihood approaches) or high computational resources (sparse estimation methods). Perfect synchronization between the reciprocal channel probes is also problematic so the chosen metric for quantization (i.e., the quantization step required for key generation, not the quantization of the directly/early sampled received signal) should be both easy to extract from the received signal and robust to synchronization errors. In consequence, we propose to use the signal obtained by integration over the duration of a pulse (Figure 6) as input for quantization.

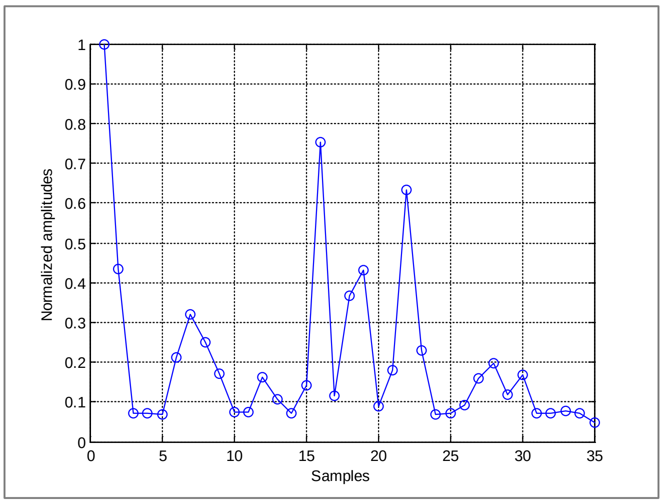

Figure 6. Example of obtained signal after integration

Information reconciliation or error correction can be achieved by Reed-Solomon codes. First, one of the two nodes encodes its binary sequence generating a syndrome and then it sends the syndrome to the other node over the public channel. The other node will decode its own sequence using the received syndrome and correcting (if possible) the mismatches. If the binary sequences of the two nodes are equal, privacy amplification can be implemented with any function from a universal family of hash functions, which is a deterministic function that maps a n-bit binary sequence in a k-bit secret key (n>k).

Once a symmetric key is available on each link of the network, a lead node can generate a group key that will then be transmitted to all the nodes of the network using the pairwise keys as one-time pads. It can be noted that there is no need to have pairwise keys on all the links but at least on a sufficient number of links that can generate a spanning tree over the given network. The length of the group key is limited by the length of the pairwise keys in the following manner:

- If we consider that there is a pairwise key between the lead node and each other node, the length of the group key is the minimum of the lengths of these keys.
- If not, the group key length is limited by the minimum key length in the most “advantageous” spanning tree (i.e., the spanning tree containing the links with the largest key lengths). In this case, additional traffic between nodes is needed in order for the lead node to be able to determine the group key length before initiating the distribution procedure.

2.2 CP-ABE Key Generation

In order to enable a CP-ABE based secure group sharing mechanism, the corresponding CP-ABE keys must be delivered to the smart objects. Indeed, the security of the model depends on the fact proper keys are generated for legitimate entities. Consequently, an explicit authentication mechanism is required in order to generate CP-ABE keys associated to the set of identity attributes related to a smart object. In addition, this process could require privacy concerns to be taken into account so a smart object can get CP-ABE keys associated to a specific subset of attributes from its whole identity. Such subset of attributes can be identified as its *partial identity*, which was considered in D2.1 [67].

Figure 7 shows the basic process by which a smart object gets a CP-ABE key from the Attribute Authority (AA). In this case, the device makes use of typical authentication mechanism in order to demonstrate it is who claims to be. For example, the smart objects could use a X.509 certificate in order to get a CP-ABE key associated to the set of attributes A contained within such credential (e.g. manufacturer or device class). Specifically, these attributes can be attached as part of the *subjectDirectoryAttributes* extension within an X.509 certificate [41]. Once it is authenticated by the AA, this entity is intended to generate a new CP-ABE key for the set of attributes in the certificate. For this purpose, it makes use of the *KeyGen* algorithm previously described, and a new SK_A key is delivered to the smart object. It should be noticed that a secure communication protocol is assumed for the key delivery. In this case, the use of *Transport Layer Security* (TLS) [42], or *Datagram Transport Layer Security* (DTLS) [43] if the *Constrained Application Protocol* (CoAP) [44] is used, can be applied for this purpose.

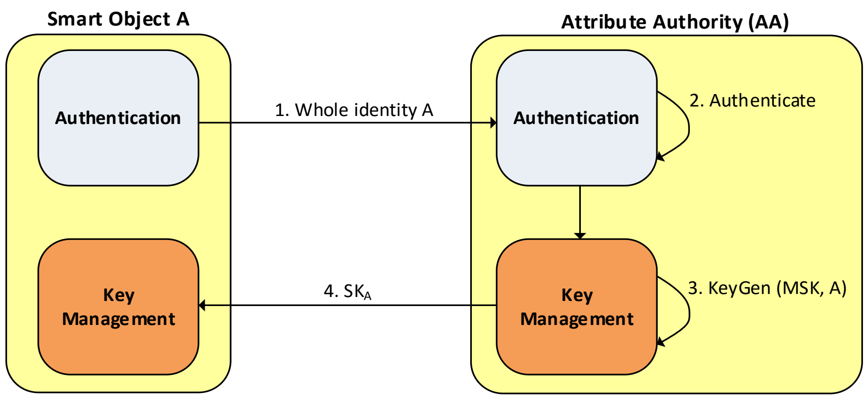

Figure 7. Certificate-based CP-ABE key generation

An additional aspect to be noticed is, in this case, the AA identifies unequivocally the smart object, and consequently, its privacy can be compromised. In order to address this issue, optionally, *Anonymous Credential Systems* [46] (e.g. Idemix [45]) could be used in order to preserve the privacy of the smart object. In this case, once the smart object A is successfully authenticated, a privacy-preserving AA generates and delivers a CP-ABE key which is associated to the set of attributes which were proved during the anonymous authentication process (e.g. attributes in an Idemix proof). Figure 8 shows a high-level description of this process.

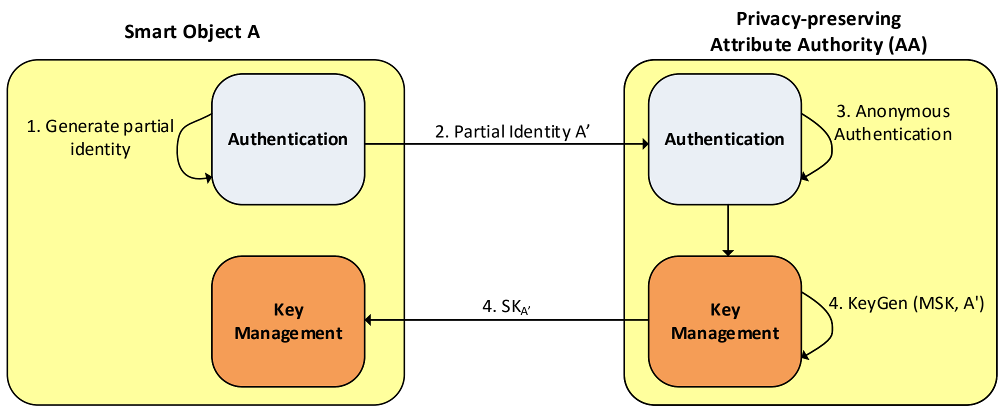

Figure 8. Privacy-preserving CP-ABE key generation

While this approach provides privacy-preserving features for the key generation process, it should be pointed out that both approaches can be used for a flexible secure group communication. Unlike more traditional symmetric key approaches, the same CP-ABE key can be employed to share information with different groups composed by entities with different attributes. This is due to the fact that data are encrypted by using attribute policies instead the keys themselves.

2.3 Secure data Sharing based on CP-ABE

Under the main foundations of the CP-ABE cryptographic scheme, in this section we provide a high-level description related to the application of such mechanism for a secure group-aware communication. Specifically, Figure 9 shows a CP-ABE based publish/subscribe scenario According to it, in addition to smart objects acting as information *producers* and *consumers*, as well as the TTP component (typically known as *Attribute Authority* (AA)), this model requires a special node known as Broker. Such entity is in charge of receiving data from producers to be published to consumers who have expressed interest over certain information. Depending on the scenario to be considered, the broker functionality may be provided by another smart object, or a central platform provided by the city council in the context of smart cities. Before data sharing among smart objects can take place, it is necessary that the system participants are in possession of the cryptographic material which is required for a secure information exchange based on CP-ABE. For this purpose, the attribute authority (AA) distributes the public parameters (common for all participants) as well as private keys (each one of them associated with a set of attributes). For this stage, the alternative processes described in Section 2.2 could be employed. Furthermore, consumers must subscribe their interest to the broker in order to get data from producers. When a producer decides to publish certain information, it sends a message consisting of a ciphertext *CT* and a label *Tag*. In this case, CT is the result of the *Encrypt* algorithm execution, which was described in Section 1.2.2. In this example, the policy (a1 ^ a2) means a subscriber that receives the message, can decrypt CT only in the case its private key is associated at least with the set of attributes a1 and a2. Additionally, Tag is used by the broker to disseminate CT to subscribers whose interest matches the Tag value.

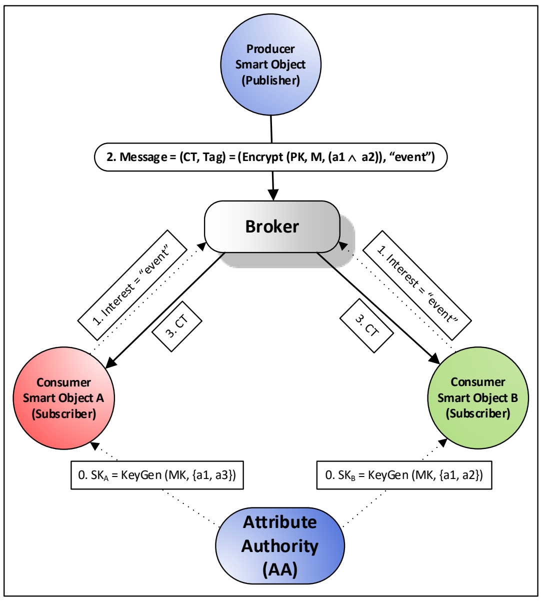
Producer Smart Object (Publisher)
2. Message = (CT, Tag) = (Encrypt (PK, M, (a1 ∧ a2)), "event")
Broker
1. Interest = "event"
3. CT
1. Interest = "event"
3. CT
Consumer Smart Object A (Subscriber)
Consumer Smart Object B (Subscriber)
0. SK_A = KeyGen (MK, {a1, a3})
0. SK_B = KeyGen (MK, {a1, a2})
Attribute Authority (AA)

Figure 9. Publish/subscribe sharing based on CP-ABE

An example of this kind of situation can be envisioned in the context of smart cities with a data sharing platform at the city council to be employed by citizens. In this scenario, consider a family in which children go to school every day. Their smartphones, equipped with different sensors such as GPS, provide their position in real-time acting as information producers. To ensure safety of children, this data should be only known by their parents or relatives but not by any other entity or user. Thus, the location information can be encrypted with the ("parent" ∨ "relative") sharing policy and uploaded to the platform to be accessed by authorized users. The main advantages of this approach arise from the application of the CP-ABE scheme to the publish/subscribe model. On the one hand, smart objects acting as data producers can share information with consumers which do not need to disclose their identity. Additionally, data confidentiality is preserved even in the presence of a compromised broker. On the other hand, compared to classical pull approaches, the asynchronous communication of this model can result in energy saving, a relevant feature in scenarios where resource-constrained IoT devices are used. Moreover, this schema is gaining attention and emerging protocols, such as *Message Queuing Telemetry Transport* (MQTT-s) [47] are being proposed and developed to be used on IoT scenarios.

The main assumption of the previous model is the existence of a broker as central component responsible for data dissemination. However, given the pervasive and dynamic nature of the scenarios envisioned by IoT community, this may represent a significant limitation in situations where data sharing is required and the setup of a broker is not possible. In this direction, in recent years, an alternative model is gaining collection, as a result of the application of the *opportunistic computing* [48] to the IoT paradigm [49]. Unlike the previous approach, this solution leverages opportunistic contact and ad hoc connection among devices, simulating the way people communicate in the physical world. Opportunistic communities can be composed spontaneously, particularly based on physical proximity, and using short-range communications technologies without infrastructure, such as Bluetooth. Due to the inherently mobile nature of smart objects (such as mobile phones or vehicles), this model has an important interest to be exploited in IoT. For example, Alice can notify that she wants to share a taxi to go home when she is in a pub. In this way, she can encrypt this message so only friends who live in her neighborhood can decrypt the message.

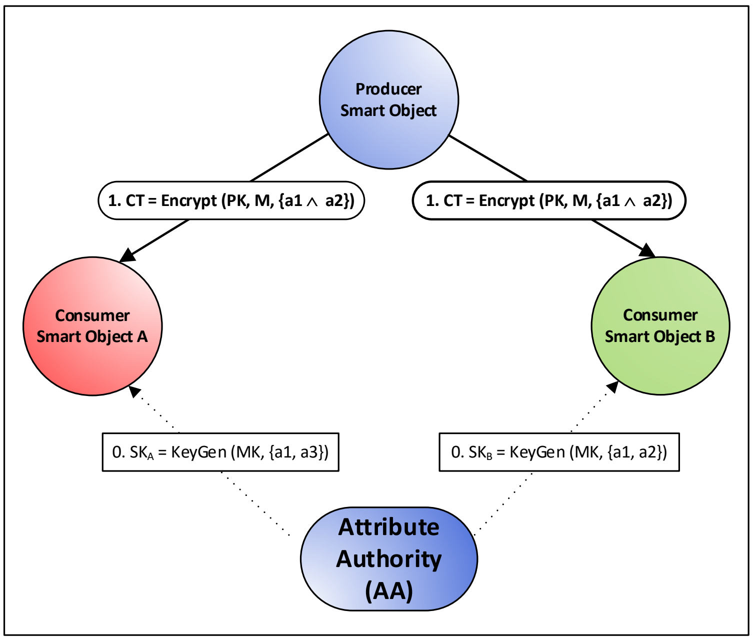
Producer Smart Object
1. CT = Encrypt (PK, M, {a1 ∧ a2})
1. CT = Encrypt (PK, M, {a1 ∧ a2})
Consumer Smart Object A
Consumer Smart Object B
0. SK_A = KeyGen (MK, {a1, a3})
0. SK_B = KeyGen (MK, {a1, a2})
Attribute Authority (AA)

Figure 10. Opportunistic secure communication based on CP-ABE

Figure 10 shows the application of CP-ABE for sharing information in an opportunistic IoT scenario. An additional advantage over the previous approach is that smart objects do not need to send additional messages to establish a publish/subscribe channel with the broker. Therefore, the delay necessary for the information dissemination is smaller, which may result in an added value in situations like those described above. An additional requirement to be considered for both approaches is the usage of techniques to ensure the validity of the source and quality of the information disseminated. While CP-ABE allows preserving the privacy of consumers, it is necessary to ensure that the information being produced do not proceed from malicious or compromised nodes. For this purpose, traditional digital signature schemes can be used. However, the use of highly sensitive personal information, such as user location, could still compromise the privacy of producers. Therefore, the application of privacy-enhancing techniques may be required so that data sharing among smart objects is carried out in a secure and privacy-preserving way. For example, a producer could sign a message by using an *Attribute-Based Signature* (ABS) [50] [51] scheme, in order to hide his real identity when sending data. In this way, data consumers could be sure that the information comes from a trusted source, while producers' privacy would be preserved.

2.4 DCapBAC for secure group communication

As was reported in D2.2 [68], the *Distributed Capability-Based Access Control* (DCapBAC) is part of the SocIoTal access control system. DCapBAC has been postulated as a feasible approach to be deployed on IoT scenarios even in the presence on devices with tight resource constraints [52]. Inspired by SPKI Certificate Theory [53] and *authorization-Based Access Control* (ZBAC) [54] foundations, it is based on a lightweight and flexible design and that allows authorization functionality is embedded on IoT devices, providing the advantages of a distributed security approach for IoT in terms of scalability, interoperability and end-to-end security. Specifically, it makes use of CoAP as a communication protocol, an optimized ECC library, as well as *JavaScript Object Notation* (JSON) [55] as a format to represent capability tokens.

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

Figure 11. Capability token example for group communications

As already mentioned, one the advantages of DCapBAC is the flexible design, allowing different specifications according to the scenario being considered. While it is mainly intended to enable a secure *Machine-to-Machine* (M2M) communication between smart objects, it can be used as a mechanism to get access to a group of entities, that is, communities and bubbles. Figure 11 shows an example of a capability token, which can be employed to access to different smart objects. In this case, the “device” field (that is typically a URI of a resource in DCapBAC) makes reference to the set of smart objects composing the bubble B. Consequently, this token could be employed by the holder smart object to get access to the set of resources being hosted by the set of entities that are part of bubble. Then, when a smart object of bubbleB receives the access request with the token, it evaluates this token according to the DCapBAC evaluation. That is, it checks whether the token is valid or not checking the signature of the entity that issued the token and if it has expired. Furthermore, it checks if any of the access rights contained in the token matches the requested action, including the conditions described in the token (if any). If these conditions are satisfied, the request is accepted and the service is provided to the requester smart object. It should be noticed that, in this case, it is assumed that smart objects are aware of their membership to a specific community or bubble.

2.5 Device context-aware data sharing

While the notion of context awareness has been well researched in recent years [56], currently there is a lack of security and privacy-preserving mechanisms that take into account dynamic context conditions [57] for the IoT. Given the pervasive, distributed and dynamic nature of IoT, context information should be considered as a first-class security component in order to drive the behavior of devices. This would allow smart objects to be enabled with context-aware security solutions, in order to make security decisions adaptive to the context in which transactions are performed. At the same time, context information should be managed by taking into account security and privacy considerations. In particular, current IoT devices (e.g., smartphones) can obtain context information from other entities of their surrounding environment, as well as to provide contextual data to other smart objects. These communications can be performed through lossy networks and constrained devices, which must be secured by suitable security mechanisms and protocols. Additionally, trust and reputation mechanisms should be employed to assess the trustworthiness of data being provided by other entities in the environment. In this way, smart objects can discard information that comes from less reliable devices. Moreover, high-level context information can be reasoned and inferred by considering privacy concerns. Thus, a smartphone could be configured to provide information about a person's location with less granularity (e.g., giving the name of the city where he is, but not the GPS coordinates), or every long periods of time in order to avoid daily habits of that person could be inferred by other entities.

As already described in D2.2 [68] the SocIoTal security framework considers context awareness as an aspect key to drive the security behavior of IoT smart objects. Specifically, for CP-ABE based secure group communication, the high-level context information coming from other smart objects can be used by the smart object to select a specific CP-ABE policy to encrypt a certain piece of data. In particular, devices are enabled with a set of *Sharing Policies* specifying how the information should be disseminated according to contextual data. These policies can be evaluated by the device itself before information is disseminated to other smart objects. The result of the evaluation of these policies is, in turn, a CP-ABE policy indicating the set of entities which will be enabled to decrypt the information to be shared. An example of sharing policy could be "*IF contextA="atPub" AND "data=myLocation", THEN CPABE policy="myfriends OR myfamily"*", specifying the location of a user is shared with friends or family members when he is at a pub. Users can be enabled with high level graphical applications in order to facilitate the specification of sharing policies for their devices. As already mentioned, when a sharing policy is successfully evaluated, the resulting CP-ABE policy is used to encrypt the information to be shared. In the case of two or more sharing policies are successfully evaluated, the most restrictive CP-ABE policy could be selected. After the information is encrypted and disseminated, the smart objects receiving such data will try to decrypt it by making use of their CP-ABE keys, which were previously obtained

2.6 Context-aware Group Sharing

One of the main aspects of SocIoTal is the implementation of context-driven privacy and security solutions. For this reason, a set of enabling technologies has been researched and developed in T3.1, and will be provided as tools to be integrated by T3.2 in the SocIoTal devices, in order to support their functionalities, including secure communication of data within groups, communities and bubbles. As already presented in D3.1.1 [69], such enablers allow the identification of different context situations, ranging from the possibility to recognize users and devices presence, and in general SocIoTal entities, in specific indoor environment (e.g., through the Indoor Localization enabler), to the possibility of characterizing users relations, in light of their devices degree of interaction (e.g., through the Face-to-Face enabler).

SocIoTal will investigate the possibility to use the information generated by the above-mentioned enablers in order to support the following secure communication functionalities:

- Providing additional attribute for the definition and identification of groups of devices to which the Group Manager will assign dedicated keys for secure communication. It should be noted that in this case, the Group Manager is considered as a central entity, which is responsible for managing the group of devices
- Opportunistically identify a distributed Trusted Third Party (TTP) acting as Group Manager for distributing keys for secure group communication.

In the first case, communities and bubbles will be built according to the indoor position the devices belong and or the kind of social relations they share (e.g., personal). The location and social relation attributes will be part of the special entity that contains the bubble or community definition. Through the Context Manager (defined in T3.2) a particular indoor position or social relation among SocIoTal entities will be identified and when the existence of a group will be recognized, keys will be either request to the Group Manager and/or pushed to the Group Manager down to the involved devices if this group is identified from the context information received through the Context Manager. To support such functionalities, SocIoTal devices will feature a Group Manager client. The definition of the Group could be initially pushed to the Group Manager using visual tools, such as the SocIoTal User Environment and the integrated Group Manager client functionalities.

Users may need to discover the bubbles along with the identity attributes that are needed to be part of the bubble. To this aim, the bubble can be registered in the centralized Context Manager as a special kind of entity, attaching the identity attributes that will be needed to demonstrate to obtain the keys, and therefore, to be able to see the cyphered exchanged data. The user environment can also provide means to register the bubbles in the Context Manager.

On the other hand, final devices that want to share data securely in bubbles should be endowed with a Group Manager implementation capable of requesting sharing keys as well as encrypt and decrypt exchanged data. Optionally, devices could also feature the modules of the Group manager in charge of dealing with on-device group sharing policies. These sharing policies could specify particular contextual conditions in order to drive device's data exchange in the bubble according to the context. The context information used in the policies may be obtained from the on-device Context Manager or from the centralized Context Manager.

In addition, in order to rely on recurrent and not only transient relations, group will be formed according to given trust and reputation scores, a computation of which will be performed by the Trust and Reputation Manager based on metrics considering the number of occurring social relations of a given type between two devices and/or the number and type of indoor location visited by a given device. This information will be added to the SocIoTal entity identity profiles and accessed by the Group Manager through the Context Manager, to verify the attributes for group participation of a given entity.

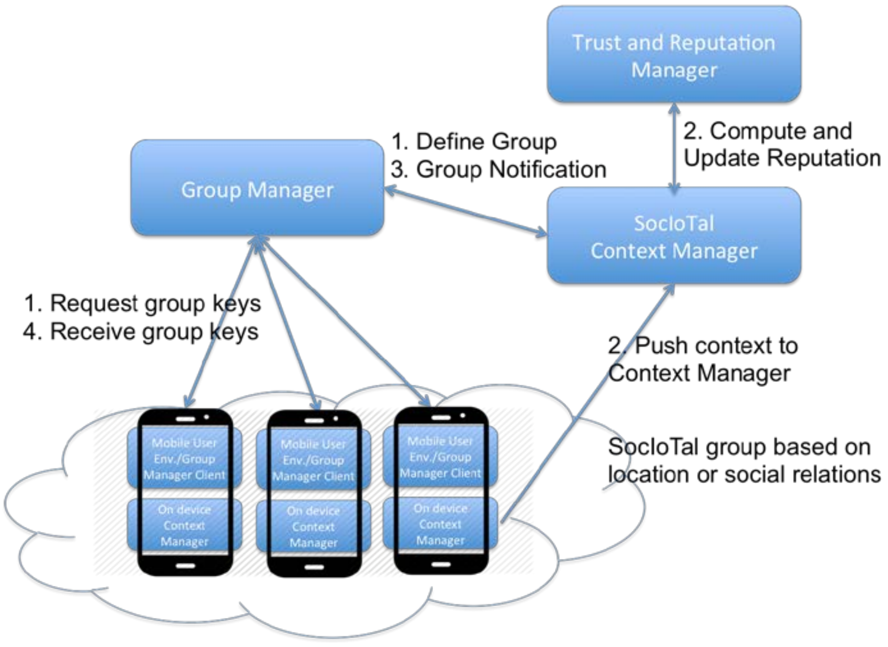
Trust and Reputation Manager
2. Compute and Update Reputation
1. Define Group
3. Group Notification
Group Manager
SocIoTal Context Manager
1. Request group keys
4. Receive group keys
2. Push context to Context Manager
Mobile User Env./Group Manager Client
On device Context Manager
SocIoTal group based on location or social relations

Figure 12. Context-aware group sharing interactions

Figure 12 shows an overview of all the interactions of the central Group Manager is involved in order to support the secure device communication required by the SocIoTal entities. As first step the Group is defined by the Group Manager as an entity pushed to the Context Manager. Group definition can also be initiated by the SocIoTal devices by means of the Group Manager client. Subsequently Context information are pushed to the Context Manager either using devices information extracted by the On Device Context Manager or a combination of them provided by the Trust and Reputation Manager. Finally, when the existence of a given Group is notified to the Group Manager, cryptographic keys are distributed to devices participating to a given group.

In the second case, using context information to identify TTP acting as Group Manager, the reputation attribute associated to each entity identity could be used to locally and opportunistically identify a device able to act as Group Manager and to distribute keys to devices participating to an opportunistic and infrastructure-less community which will not require connection to the central SocIoTal Group Manager. This represent an initial idea that will be further investigated during the project. For instance, the device which shows less mobility and is for most of the time bounded to the same indoor location might gain a higher reputation scores thus making it suitable to be selected as Group Manager, as well as the one which share a more frequent number of personal relations with the surrounding devices can be identified as the most central one selected for this role. While this represents an interesting solution, able to increase the level of privacy of the system, in which no third party are involved into the process and trust for the selected Group Manager is automatically and transparently recognized by the other participating devices, however the realization of this will be investigate further during the project as it will require to solve different issues, such as how to opportunistically advertise the creation of a new Group Manager which is not centrally identified at bootstrapping time.

Section 3 - Secure Group Sharing within SocIoTal security framework

The challenging complexity for a secure management of IoT smart objects imposes the need to consider architectural approaches, taking into account the inherent requirements of the application of security and privacy-preserving mechanisms on IoT scenarios. Indeed, the huge potential of IoT may be threatened if security and privacy concerns are not taken into account from the beginning, supporting aspects such as privacy by design [58], and data minimization [59] principles, in order to give people maximum control over their personal data. IoT-A [60] was a large-scale European project focused on the design of an Architectural Reference Model (ARM), in order to optimize the interoperability among isolated IoT domains to create a global ecosystem of services under a common view. This promoted additional initiatives adopting ARM as the starting point of design activities, favoring the alignment of architectures and enabling to reuse functionalities and components among different application domains. However, security and privacy concerns are not the main focus of such architectures. The SocIoTal ARM-compliant security framework, which is shown in Figure 13, addresses these requirements by instantiating and extending the security functional group of ARM, which promotes its applicability and interoperability in a wide range of IoT scenarios, in which security and privacy are required.

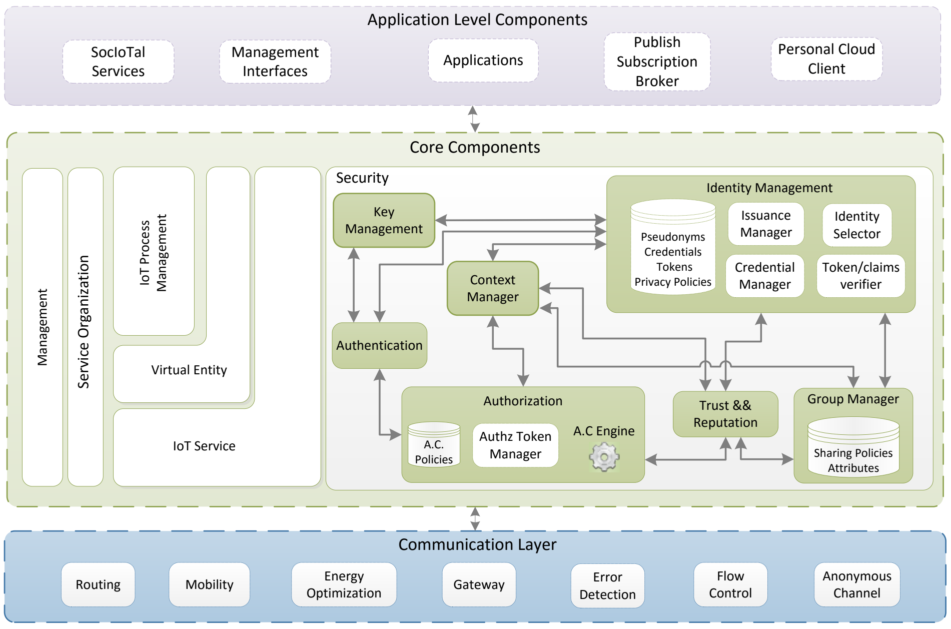

Figure 13. SocIoTal security framework

This framework is used as a starting point for the definition of the mechanisms for secure group communications in SocIoTal. These mechanisms are based on the technologies that were presented in Section 1, as well as the main foundations described in Section 2. Specifically, Section 3.1 provides a detailed explanation of the main mechanisms to generate and deliver group keys based on Symmetric Key Cryptography, while Section 3.2 gives the description of the main processes and interaction for CP-ABE based secure group communications.

3.1 Group Keys generation and delivery

3.1.1 Requirements for Group Key Management

One driving requirement of a cluster key management framework is the need of scalability to enable new members to join the group, to enable a member to leave the group and to enable the eviction of a member, while ensuring the foreword and the backward security and mitigating the one-affects-all phenomena.

So, we would like that the framework provides:

- **Scalability**: A group can contain highly dynamic membership that leads to frequent re-keying. This phenomena should be mitigated the frequency of the re-keying that is resource consumer.
- **Robustness**: The effect of an attack should be minimized thanks to the ability of self-adaptation of the design.
- **Security Layer Independence**: The framework should enable secure multicast through scalable key management over the network layer. The security embedded at the network layer ensures a peer-to-peer unicast secure channel.
- **Security Technology Independence**: The security can be achieved using multiple cryptographic algorithms and protocols.
- **Communication Protocol Independence**: It should be possible to implement the framework over any communication protocol that supports unicast and multicast services.

3.1.2 The cluster key management sub-framework

Large and dynamic groups do not scale up for multicast security. This may create a scalability problem. The framework that we introduce in the following is adapted from Iolus framework [1]. Several smaller multicast sub-groups forming bubbles of users are introduced. The whole bubbles form a virtual secure multicast group. Scalability is achieved as each bubble is independent from the others. Each sub-group or bubble has its own multicast address and keying material. Thus, when a member joins or leaves the group, it affects only its local sub-group.

From D2.1 “IoT Communities and Identity Management**”** (Section 2.2.2) [67], we have adopted these three definitions:

Bubble: A Personal Bubble consists of smart objects sharing a common trust relationship (related to its owner’s identity), where security and privacy are their main features.

Community: A hybrid communities is mixing infrastructure support for several functionalities such member registering, management or event pre-processing and ad-hoc mode to provide direct access to user’s shared services or devices.

Opportunistic Bubble: It is a new kind of bubble or community based on a specific set of security policies and created spontaneously when specific circumstances or relationships conditions arise.

At this stage, three new entities are introduced: A personal bubble is composed of several sub-clusters holding smart objects or devices. Each of these sub-clusters holds a Group Security Agent (Figure 14). The Group Security Agents (GSA) may form a hierarchy of sub-clusters. A Group Security Intermediary (GSI) is a trusted server authorized to act as a proxy of the Group Security Controller (GSC) that manages its associated sub-clusters in a bubble. Via their personal bubble, several users can be grouped into a community to share services or devices. So, a same user may belong to several communities at the same time, sharing the same or different sub-clusters with each community. The GSC ensures the link and the consistency between all the bubbles for a given community and enables the multicast of data in the virtual group. The Group Security Controller (GSC) ensures the control of the top-level sub-group at the root of the distribution tree (Figure 15).

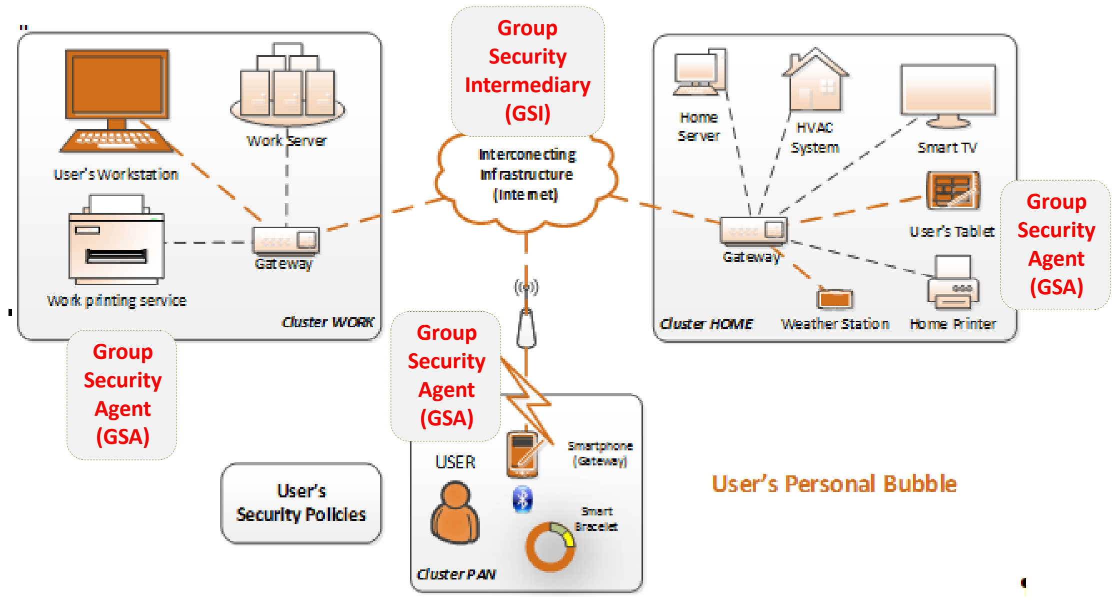

Figure 14. Group Security elements into a personal bubble

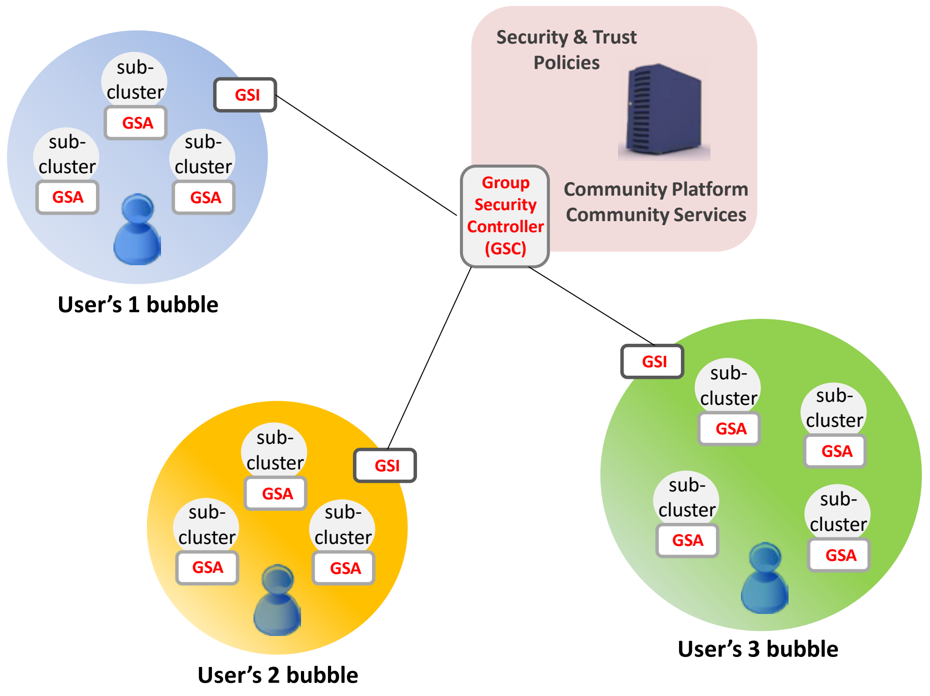

Figure 15. Group Security elements into a community of bubbles

In the following, the Cluster Key is used at the network layer. It is established thanks to the Personal Key that can be symmetric or preferably a couple of Public/Private Key based on asymmetric cryptography via a unicast channel (Figure 16).

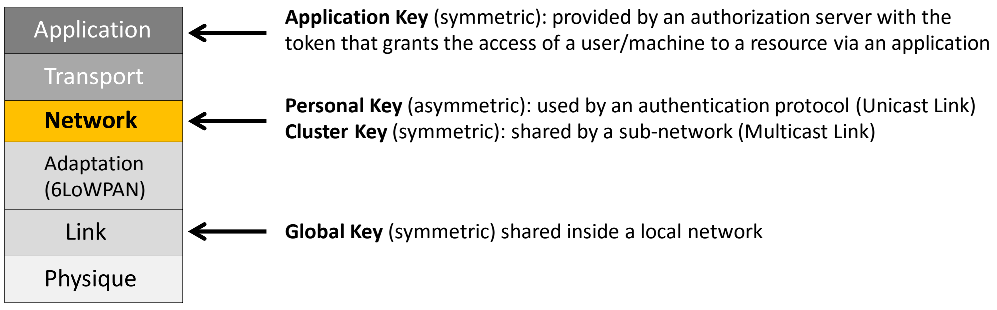
Application
Transport
Network
Adaptation (6LoWPAN)
Link
Physique
Application Key (symmetric): provided by an authorization server with the token that grants the access of a user/machine to a resource via an application
Personal Key (asymmetric): used by an authentication protocol (Unicast Link)
Cluster Key (symmetric): shared by a sub-network (Multicast Link)
Global Key (symmetric) shared inside a local network

Figure 16. Mapping of the Keys into the layers of the OSI model

3.1.3 The cluster key management protocols

3.1.3.1 Join protocol

1. The member MBR who wants to join the network locates its appropriate GSA and sends a join request to it using a **unicast** secure channel.
2. The GSA checks its database to decide if it approve or deny the request. If it approves, the GSA generates a new secret called $K_{GSA\text{-}MBR}$ as session key.
3. The GSA sends $K_{GSA\text{-}MBR}$ only to the new member via a **unicast** secure channel.
4. The GSA generates a new sub-cluster key $K1_{SGRP}$ and encrypts it with the previous sub-cluster key $K0_{SGRP}$ that is the current secret shared between the group members.
5. The new sub-cluster key $K1_{SGRP}$ is encrypted with the session key $K_{GSA\text{-}MBR}$ and sent to the new member via a **unicast** secure channel.
6. The new encrypted sub-cluster key is then **multicast** to the current members.

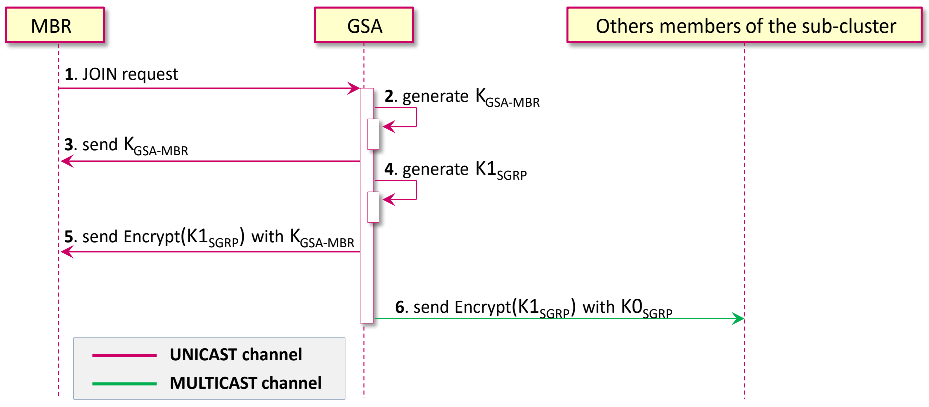
MBR
GSA
Others members of the sub-cluster
1. JOIN request
2. generate $K_{GSA\text{-}MBR}$
3. send $K_{GSA\text{-}MBR}$
4. generate $K1_{SGRP}$
5. send Encrypt($K1_{SGRP}$) with $K_{GSA\text{-}MBR}$
6. send Encrypt($K1_{SGRP}$) with $K0_{SGRP}$
UNICAST channel
MULTICAST channel

Figure 17: Join protocol

3.1.3.2 Leave protocol

1. The GSA generates a new secret called $K2_{SGRP}$ as session key.

2. As there is no common channel between the remainder members of the sub-cluster, the new sub-cluster key is encrypted with each session key of the remainder members and sent separately using a **unicast** secure channel.

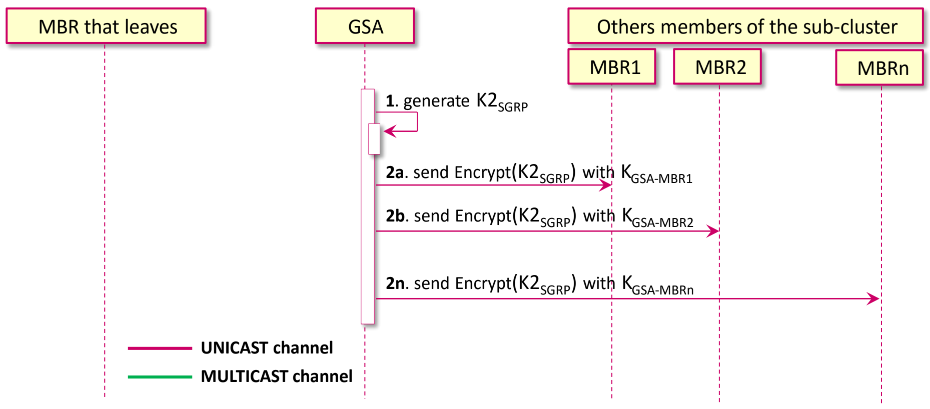
MBR that leaves
GSA
Others members of the sub-cluster
MBR1
MBR2
MBRn
1. generate $K2_{SGRP}$
2a. send Encrypt($K2_{SGRP}$) with $K_{GSA-MBR1}$
2b. send Encrypt($K2_{SGRP}$) with $K_{GSA-MBR2}$
2n. send Encrypt($K2_{SGRP}$) with $K_{GSA-MBRn}$
UNICAST channel
MULTICAST channel

Figure 19: Leave protocol

3.1.3.3 *Exchange of message from one to many into a personal bubble*

1. The sender generates random keying material K_{random} and encrypts the message with it.
2. The sender **multicasts** the encrypted message to its sub-cluster members and to its local GSA.
3. The local GSA transmits the encrypted data to others GSA via a **unicast** channel
4. Each GSA **multicasts** the encrypted data over in its sub-cluster.

10. The sender encrypts the key material with its session key and sends the encrypted message to its GSA.
11. The local GSA decrypts the message to get the key material K_{random}.
12. The GSA re-encrypts K_{random} with its sub-cluster key and **multicasts** the result into its sub-cluster.
13. The local GSA transmits the encrypted data to others GSA via a **unicast** secure channel.
14. The targeted GSAs decrypt the incoming message.
15. Each GSAre-encrypts the key material K_{random} with their current sub-cluster keys. Then, it **multicasts** the encrypted key material messages over their sub-clusters.

By this way, all the members of the network can read the data.

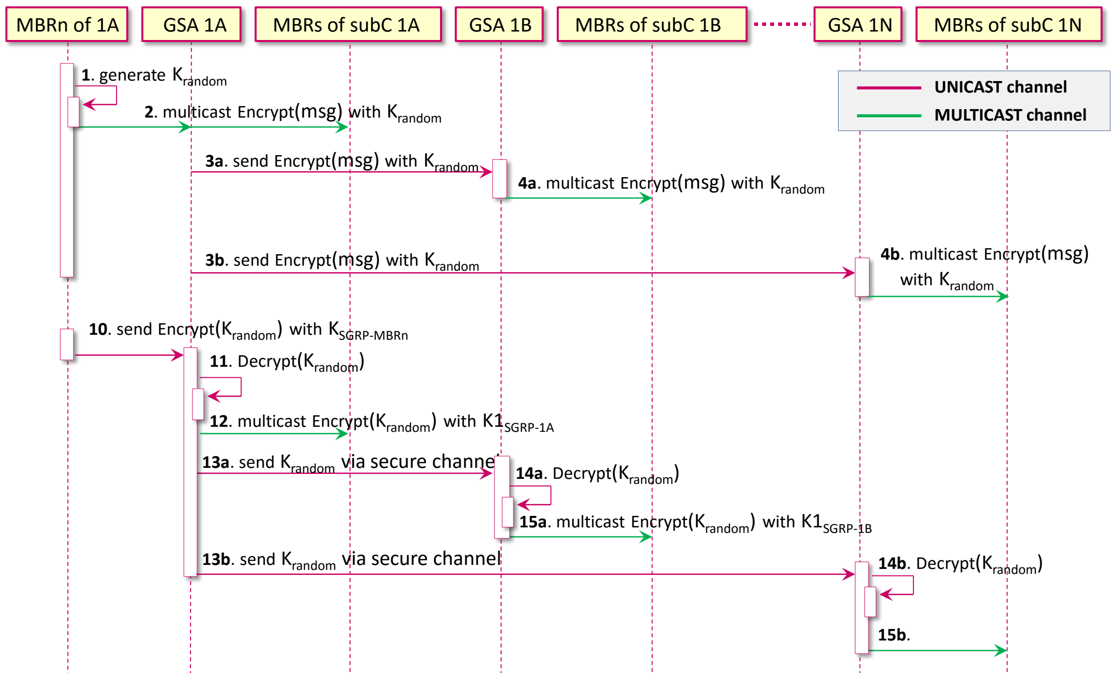
MBRn of 1A
GSA 1A
MBRs of subC 1A
GSA 1B
MBRs of subC 1B
GSA 1N
MBRs of subC 1N
UNICAST channel
MULTICAST channel
1. generate K_random
2. multicast Encrypt(msg) with K_random
3a. send Encrypt(msg) with K_random
4a. multicast Encrypt(msg) with K_random
3b. send Encrypt(msg) with K_random
4b. multicast Encrypt(msg) with K_random
10. send Encrypt(K_random) with K_SGRP-MBRn
11. Decrypt(K_random)
12. multicast Encrypt(K_random) with K1_SGRP-1A
13a. send K_random via secure channel
14a. Decrypt(K_random)
15a. multicast Encrypt(K_random) with K1_SGRP-1B
13b. send K_random via secure channel
14b. Decrypt(K_random)
15b.

Figure 20: Message exchange into the personal bubble of User 1

3.1.3.4 Exchange of message from one to many into a community

1. The sender generates random keying material K_{random} and encrypts the message with it.
2. The sender **multicasts** the encrypted message to its sub-cluster members and to its local GSA. The local GSA transmits the encrypted data to others GSA into the user's personal bubble via a **unicast** channel. Each GSA **multicasts** the encrypted data over in its sub-cluster.
3. The GSA sends the encrypted keying material to the community GSC via a **unicast** channel. The GSC forwards the incoming message to the GSI of the users belonging to the community.
4. The GSI transmits the incoming message to the GSAs of the sub-clusters involved in the community.

10. The sender encrypts the key material with its session key and sends the encrypted message to its GSA.
11. The local GSA decrypts the message to get the key material K_{random}.
12. The local GSA re-encrypts it with the sub-cluster keys it wants to address inside its user's personal bubble. Then, it **multicasts** the resulting message to the targeted sub-group.
13. The local GSA transmits the encrypted data to others GSAs of the community via the other users GSI.
14. The targeted GSAs decrypt the incoming message to get K_{random}.
15. Each GSA re-encrypts the key material with its current sub-cluster key. Then, it **multicasts** the encrypted key material over in its sub-clusters.

By this way, all the members of the network can read the data.

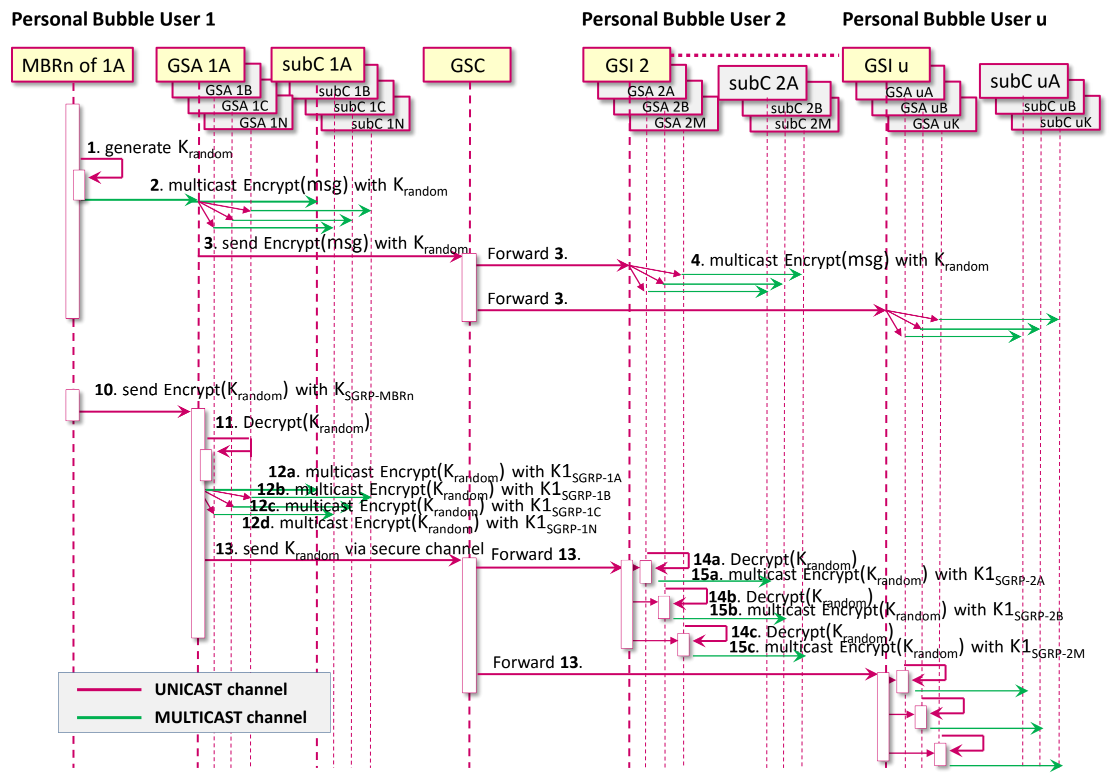
Personal Bubble User 1
Personal Bubble User 2
Personal Bubble User u
MBRn of 1A
GSA 1A
GSA 1B
GSA 1C
GSA 1N
subC 1A
subC 1B
subC 1C
subC 1N
GSC
GSI 2
GSA 2A
GSA 2B
GSA 2M
subC 2A
subC 2B
subC 2M
GSI u
GSA uA
GSA uB
GSA uK
subC uA
subC uB
subC uK
1. generate K_random
2. multicast Encrypt(msg) with K_random
3. send Encrypt(msg) with K_random
Forward 3.
4. multicast Encrypt(msg) with K_random
Forward 3.
10. send Encrypt(K_random) with K_SGRP-MBRn
11. Decrypt(K_random)
12a. multicast Encrypt(K_random) with K1_SGRP-1A
12b. multicast Encrypt(K_random) with K1_SGRP-1B
12c. multicast Encrypt(K_random) with K1_SGRP-1C
12d. multicast Encrypt(K_random) with K1_SGRP-1N
13. send K_random via secure channel
Forward 13.
14a. Decrypt(K_random)
15a. multicast Encrypt(K_random) with K1_SGRP-2A
14b. Decrypt(K_random)
15b. multicast Encrypt(K_random) with K1_SGRP-2B
14c. Decrypt(K_random)
15c. multicast Encrypt(K_random) with K1_SGRP-2M
Forward 13.
UNICAST channel
MULTICAST channel

Figure 21: Message exchange in a community

3.1.4 New components in the group manager

The components “Group Security Agent”, “Group Security Intermediary” and “Group Security Controller” can be introduced into the Group manager of the reference architecture. The User Profile holds the composition of the various sub-clusters of each user’s personal bubble. The community profile will reference the composition of the communities composed of many user’s sub-clusters. A given user may choose to share some of its sub-cluster with a community and the same or others with another community. The way that a user organizes its own sub-clusters gives a granularity in the sharing of its objects or devices with the others users inside a community. The profile of the communities is accessible by the GSC and can address remote and heterogeneous sub-clusters of different users. The Key Management component will generate the keying material for the GSA.

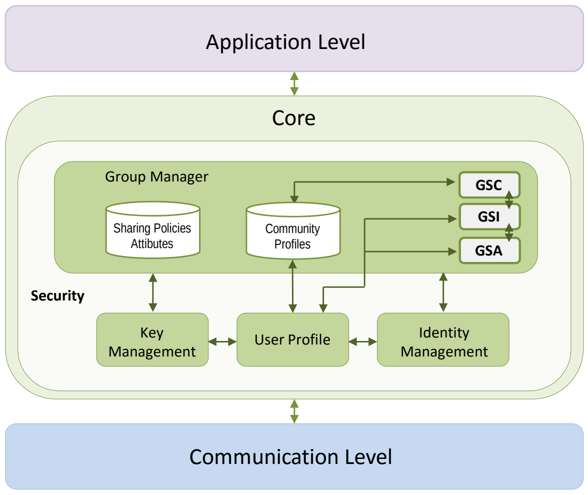

Figure 22. Details of the Group Manager including the new components

3.2 Secure Data Sharing with CP-ABE

One of the alternatives considered in SocIoTal to share data securely across bubbles and communities is the usage of *Attribute-Based Encryption* (ABE) instead of traditional *Symmetric Key Cryptography* (SKC). In this sense, as already described in D2.2 [68], the Group Manager component of the SocIoTal security framework is based on the use of the CP-ABE cryptographic, which was introduced in Section 1. Such scheme enables a flexible secure group communication for the envisioned SocIoTal communities and bubbles of smart objects. Figure 22 shows how contextual information from the Context Manager of a subject entity (acting as an information producer) is used by the Group Manager to select a specific CP-ABE policy to encrypt a specific information. Specifically, the Group Manager contains a set of *Sharing Policies* that enable to specify how the information is disseminated according to contextual data. These policies are evaluated by the *Group Sharing engine* before information is disseminated by the subject. The result of the evaluation of these policies is, in turn, a CP-ABE policy, which is employed by the *CP-ABE engine* to encrypt the information with that policy. After the information is encrypted and disseminated, the Group Manager of a target entity (acting as a consumer) will try to decrypt it with CP-ABE keys related to its identity attributes through its *CP-ABE engine*.

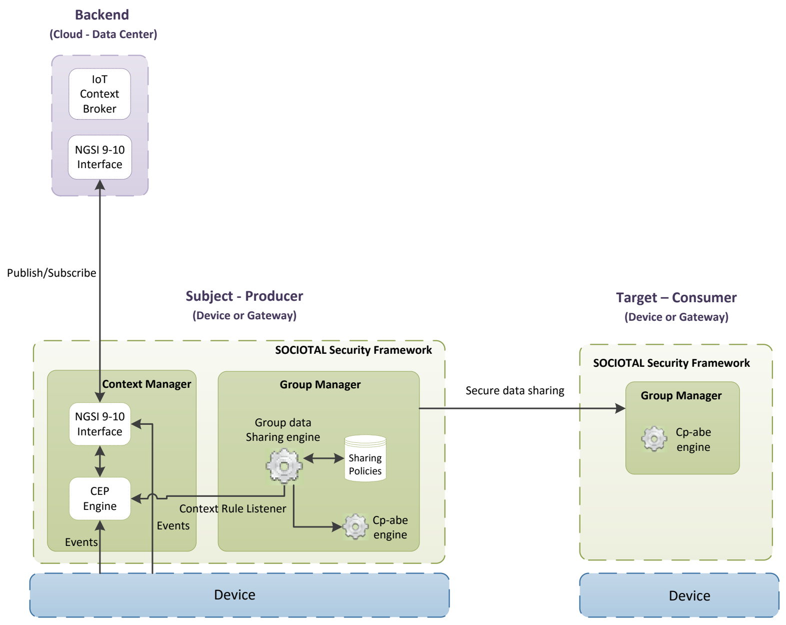

Figure 23. SocIoTal device-to-device context-aware group sharing

As can be seen, in the producer side, the Group manager evaluates the sharing policies taking into account the Context handled by the local Context Manager, which in turn, can interact with the Context broker. As a result of the sharing policies evaluation, a CP-ABE policy is selected to encrypt the data, which is then delivered securely to the consumer(s). Therefore, the Group Manager module allows the data producer to share information with groups of entities which meet certain combinations of identity attributes. These combinations of attributes are represented by sharing policies which, in turn, are influenced by context information in which sharing transaction is going to be performed.

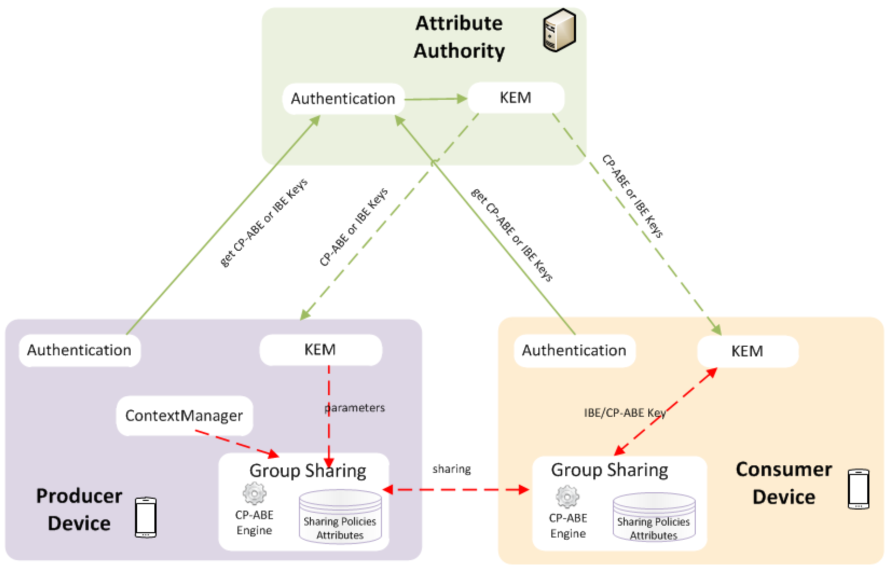

Figure 24. Main framework interaction to obtain CP-ABE keys and realize secure group communications

In addition, Figure 23 shows an overview of the main components interactions of our security framework, which are required when the secure group sharing takes place between a producer and consumer. During a first stage, both, the producer and consumer have to obtain from the Issuer the CP-ABE keys and cryptographic parameters that are required to encrypt and decrypt information with a specific policy of attributes. Notice that, although it is not shown for the sake of clarity, this process also requires an explicit authentication process by which an entity proves it has a specific set of attributes. Then, the Group Manager of the producer entity evaluates the sharing policies taking into account the Context handled by the local Context Manager. As a result of the sharing policies evaluation, a CP-ABE policy is selected to encrypt the data, which is then delivered securely to the consumer(s). Finally, the consumers make use of their keys to decrypt such information.

3.2.1 Sequence Diagram

While D2.2 [68] provided a sequence diagram of the CP-ABE based secure group communication by using a push pattern, according to the SocIoTal security framework interaction, and the use of this cryptographic scheme, Figure 24 shows the sequence diagram for the secure group sharing mechanism, by considering the SocIoTal Context Manager (acting as a broker) in a publish-subscribe patter. At high-level, four entities are considered for this mechanism:

– The Subject role, acting as a *consumer*, which is intended to get access to a specific data. This entity can subscribe to the SocIoTal Context Manager, in order to receive notifications from potential producers.
– An IoT Service role, acting as a *producer*, which will be in charge of publishing data to the SocIoTal Context Manager to make information available to a specific group of entities, by using the CP-ABE scheme. As already shown, such entity consists of other components of the SocIotal security framework which will perform the task of determining a specific CP-ABE policy to encrypt data.
– The *Attribute Authority*, which is responsible for generating CP-ABE keys for potential consumers. This entity will be usually deployed in a separate component. In addition, this entity must deliver the proper public parameters (see Section 1.2.2) for producers.
– The *SocIoTal Context Manager*, which will act as an information broker. On the one hand, it will receive subscriptions requests from consumers for specific producers. On the other hand, it will receive notifications from such entities and, consequently, it will disseminate such data to subscribed consumers.

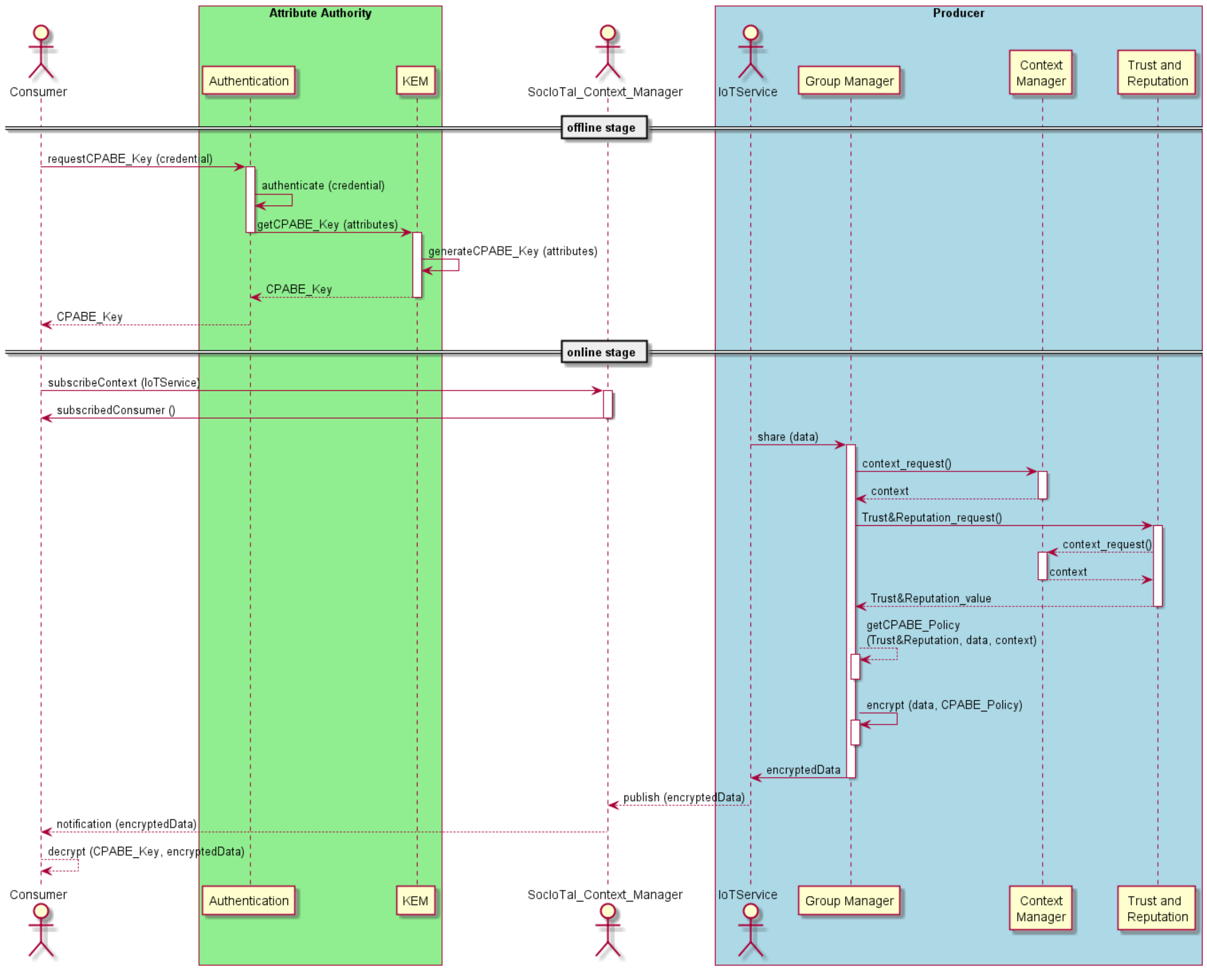

Figure 25. Secure Group Sharing sequence diagram

The description of the sequence diagram is as follows:

- Firstly, during an *offline phase*, the consumer entity wishes to get a CP-ABE key associated with a set of attributes of their identity. These attributes can be specified in the form of an authentication token or credential issued by a trusted entity. For this purpose, the subject presents it credentials to the Authentication/Identity Manager depending on the authentication mechanism employed (e.g. based on X.509 certificates or Idemix). One the subject is successfully authenticated, the KEM component generates a CP-ABE key according to the identity attributes which were proved by the subject during the authentication stage. Consequently, for this stage, any mechanism described in Section 2.2 could be employed. Although it is not shown for the sake of clarity, during this phase, data producers obtain public parameters from the Attribute Authority in order to encrypt information with CP-ABE.
- Then, during an *online* stage, data consumers subscribe to the SocIoTal Context Manager in order to receive notifications from producers. For this purpose, consumers can make use of the NGSI-9 and NGSI-10 methods.
- From the producers' side, in this case, the *IoTService* decides to publish a certain data to the SocIoTal Context Manager. Therefore, it notifies the Group Manager component to perform such publication in a secure way for a set of potential consumers.
- The Group Manager is the component responsible for encrypting data. The decision on which CP -ABE policy to use for this data is determined by three factors:
  - The type of the data to be pushed (e.g., location, identity, etc.)
  - The current context of the IoTService which is obtained according to the previous section.

    - o Optionally, the values of trust and reputation associated with these policies (actually representing groups of smart objects), which are obtained by requesting the *Trust && Reputation* component.
- When these values have been determined, a set of sharing policies which are in the *sharingPolicies* database of the GroupManager are evaluated. The result of this evaluation will be a CP-ABE policy which will be used to encrypt the data.
- Finally, a subcomponent *CPABEEngine* inside the Group Manager is responsible for encrypting the data with the policy obtained in the previous step.
- Once the data has been encrypted, it is returned to the IoTService actor which finally publishes the encrypted data to the SocIoTal Context Manager. For this purpose, the IoTService makes use of the NGSI-10 *updateContext* method of the NGSI interface. This message contains the encrypted data to make it available for a certain group of subscribed consumers, by using CP-ABE.
- When such information is received by the SocioTal Context Manager, it sends the corresponding notifications to the subscribed consumers. It should be pointed out that data from producers cannot be decrypted by the SocIoTal Context Manager. In addition, although notifications are sent to all subscribed consumers, only those entities with CP-ABE keys (obtained in the previous stage) satisfying the CP-ABE policy, will be able to decrypt the information

3.2.2 Interfaces Definition (API)

According to the CP-ABE base secure group mechanism for the publish-subscribe scenario, it should be noted that consumers and producers could make use of the methods provided by NGSI-9 and NGSI-10 API. In this way, they can subscribe and publish information to the SocIoTal Context Manager in a controlled way. In addition to NGSI, the envisioned API functions for the use of a CP-ABE schema to realize group communication security are:

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

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

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

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

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

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

- **share** (data)

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

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

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

Section 4 - Secure Group Sharing for Communities and Bubbles

In order to cope with the secure management requirements for group communications, this section offers a detailed overview related to the mechanisms provided for this purpose. Specifically, according to the main technologies and techniques provided in Section 1 and Section 2, this section provides a description about how framework components can interact, in order to realize such mechanisms for secure communication among communities or bubbles of smart objects. In particular, different cryptographic alternatives are described for the secure management of the several types of groups, which were reported in D2.1 “IoT Communities and Identity Management” [67] and D2.2 “Framework Specification for Privacy and Access Control” [68].

4.1 Group Keys Provisioning

This section describes the group keys provisioning process for secure group communications, by considering the use of Symmetric Key Cryptography. The Group Key provisioning is a part of the Key Management Framework that includes four levels of keys:

- The global key
- The personal key
- The cluster key
- The application key

The *global key* (Figure 26) is shared by all the devices of a local network and protects the whole local network from outside. It is deployed at the Link Layer (MAC layer for IEEE 802.15.4 wireless low-power network). This key is based on symmetric cryptography.

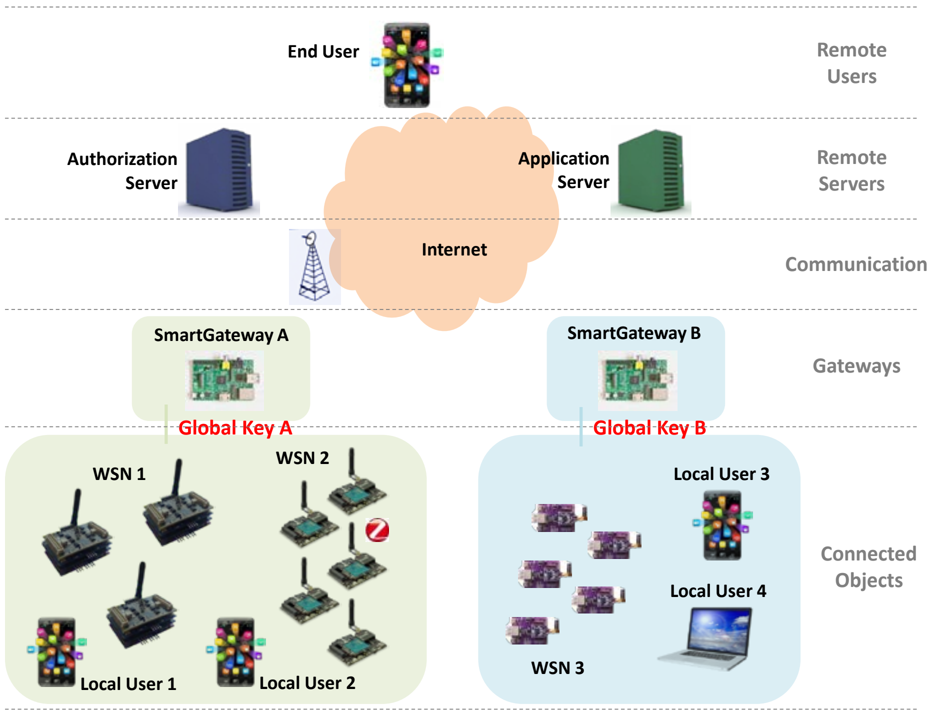

Figure 26. Global Key

The *personal key* (Figure 27) can be symmetric or asymmetric. It is a secret attribute of a member. It is used to ensure the security of a unicast channel between two peers. It could be hold at the network layer.

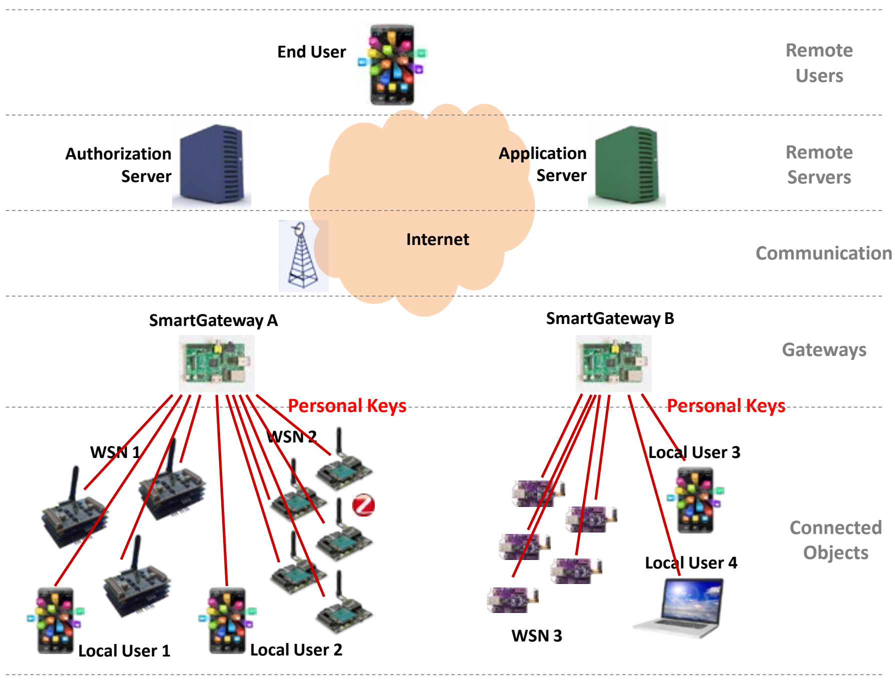

Figure 27. Personal Key

The *cluster key* (Figure 28) is a symmetric key shared between several members of a community who want to share a common service. It is used by multicast communication from one member to many at the network or session layer. Its generation and delivery has been described in paragraph 3.1. The cluster key enables to handle personal bubble and communities and to share securely resources. It is based on Iolus protocol [1] customized in order to fit in the Sociotal framework and architecture.

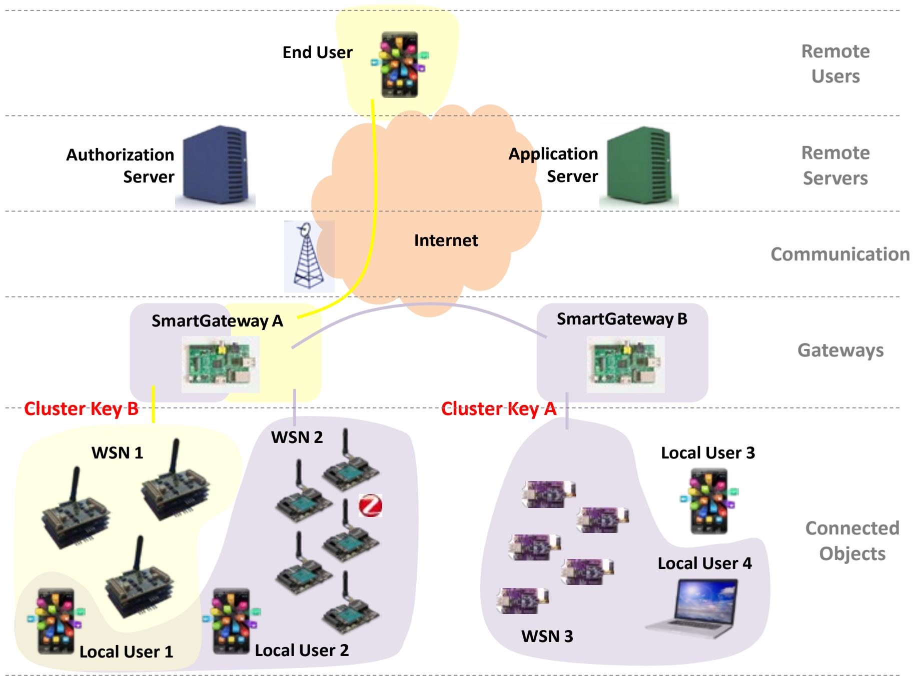

Figure 28. Cluster Key

The *application key* (Figure 29) is symmetric and used when an access is granted for a given user that wants to use a given resource via a given application at the application layer of the OSI model.

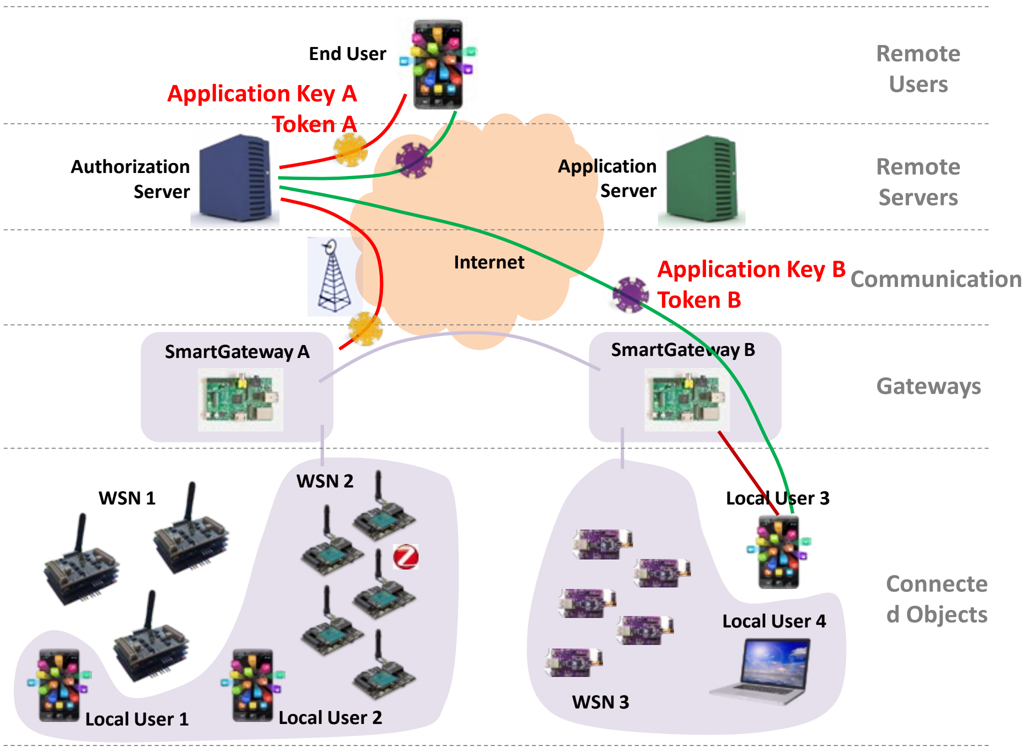

Figure 29. Application Key

4.2 Centralized Community Creation and Data Sharing

In order to provide users with the capability of sharing information safely among them, the SocIoTal Community Management tool is developed. Through this tool, users will be able to create communities to share selected information with other community entities (users or resources), without leaking of information. However, before creating a community there must be a set of premises that have to be solved, such as the user registration, the resource registration, the specification of the relation between the users and their resources, etc. Below, there will be described the different steps and premises that has to be completed in order to finally access a resource shared by users within a community.

4.2.1 User registration

The first step that a user will have to follow to be enrolled within the SocIoTal environment, will be their registration. The user, through the User Environment provided by SocIoTal, will complete a small template where they will complete some needed values or attributes for their identification within the platform. Those values will be useful to provide them with different services and to allow them take part of different communities of users within SocIoTal. Once the user has completed the template, this will arrive to the SocIoTal Registration Tool. It will create a new entry within the SocIoTal User Directory and an identifier (user_ID) will be returned to the user.
Once the users have registered themselves within the platform, the SocIoTal Community Management tool will create, by default, an Own Community of the user and its community_ID will be returned. This Own Community will be the virtual place where all the resources

belonging to the user will be added henceforth. Within this Own Community, the user will play a role of "administrator", so they will have the rights to perform all possible operations among their devices (such as update the information, query the information, add or delete resources, etc.).
However, although the user has the administration role to manage their devices, they will need a token to authenticate them as a user with an administration role within their Own Community. In order to do that, the SocIoTal Community Management Tool will request a token to the authorization block. After a positive checking of the user's attributes against the correspondent policies, a token will be created from the attributes of the user (user_ID, community, role, etc.) and the rights that its attributes assign to them. Once the token is delivered to the user, they will be able to use this key as the link between them, their role and their community.
All the personal information required during the user's registration process will be stored within the SocIoTal User Directory, and will be used only by the platform in order to create and manage the communities and its security. This User Directory and the data bases involved will be subject to the laws and regulations related to personal data protection in force in the country it is deployed and, overall, will be subject to the Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data [70].

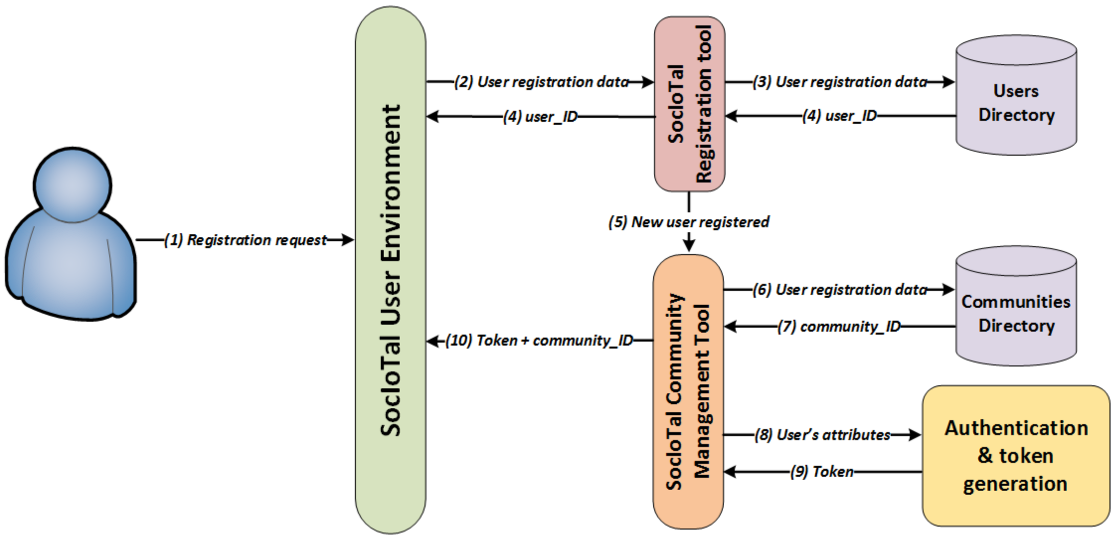
(1) Registration request
SocIoTal User Environment
(2) User registration data
(4) user_ID
SocIoTal Registration tool
(3) User registration data
(4) user_ID
Users Directory
(5) New user registered
(10) Token + community_ID
SocIoTal Community Management Tool
(6) User registration data
(7) community_ID
Communities Directory
(8) User's attributes
(9) Token
Authentication & token generation

Figure 30. User registration process

Figure 30 presents the process of a user registration, which can be summarized as follows:

1. User registration request with data filled in a template through the SocIoTal User Environment.
2. The request arrives to the SocIoTal Registration Tool
3. The user's data is registered within the Users Directory (UD)
4. The UD returns the user_ID to the Registration tool, and this to the user
5. The SocIoTal Community Management tool is informed about the registration of a new user to create this/her Own Community
6. The Own Community related to the user is created within the Communities Directory (CD)
7. The CD returns the community_ID to the CMT
8. The CMT requests a token to the authentication block with the user's attributes (user, community, admin role, etc)
9. Within the authentication block the attributes are checked against the correspondent policies and a token is created and returned it to the CMT

10. The community_ID and token is returned to the user

4.2.2 Device registration

Once the user has been properly registered, they will be allowed to register their own devices, adding therefore new resources to SocIoTal. The process of registering a new device will be performed through the User Environment, which will provide templates to facilitate the registration process and will be in charge of interacting with the SocIoTal Registration Tool.
As it has been said before, the new devices will be added by default to the Own Community of the user. Indeed, the relation between the user's resources and himself will be defined by the belonging of the resources to the Own Community. This will be performed through the addition of an attribute called "Communities" in the resource registration within the SocIoTal Context Manager (SCM), where its value will be a list of communities identifiers where the first value will be the community_ID of the user's Own Community.
The mentioned attribute "Communities" through its value will not only specify the Own Community, but also the set of communities where the correspondent resource will be shared with other users. This way, when a user wants to share a resource within other community, the resource registration will have to be modified in order to add a new member (the new community_ID) to the list within the value of the attribute "Communities".
Although the process of adding a new resource to a community will be managed easily by the user through the User Environment, an example of the coding (transparent to the user) is showed in Figure 31. This presents an example of the payload within the updateContext operation to the SCM when adding a new community where the resource will be shared (laboratory_ID). It can be seen that there is already other community_ID corresponding to the Own Community of the user, added when the device was initially registered.

Payload:
```
{
    "contextElements": [{
        "type": " urn:x-org:sociotal:resource:device ",
        "isPattern": "false",
        "id": "SocIoTal:SAN:WeatherStation:Dev_00X",
        "attributes": [{
            "name": "AmbientTemperature",
                "value": "25.44",
                "type": "http://sensorml.com/ont/swe/property/AmbientTemperature",
                "metadatas": [{
                    "name": "DateTimeStamp",
                    "value": "20141030T113343Z",
                    "type": "http://sensorml.com/ont/swe/property/DateTimeStamp"
                },
                {
                    "name": "Unit",
                    "value": "celsius",
                    "type": "http://purl.oclc.org/NET/ssnx/qu/qu#Unit"
                },
                {
                    "name": "DataDescription",
                    "value": "float",
                    "type": "http://sensorml.com/ont/swe/property/DataDescription"
                }]
            }
            {
                "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"
                }]
            },
            {
                "name": "Communities",
                “value”:[“ownCommunity_ID”, “laboratory_ID”],
                “type”:”communityType”
            }],

    }],
    "updateAction": "UPDATE"
}
```

Figure 31: Payload for the update operation when adding a resource to a new community

4.2.3 Community creation

Once the user has been registered they will be able to create a community. Within the community, the owner and other members will share information safely from the resources that will be added by them to the community, without disclosure of information out of the community. The SocIoTal User Environment will provide the user with the interface to access the functionalities that the SocIoTal Community Management Tool will offer to create a new community in an easy way. The different steps to be performed by a user to create a community can be described as follows. First of all, the user will fill a template to establish values such as the community name, a community description, the owner (the creator) and their role, other users and their roles, etc. Then, the description data of the community will be stored within the SocIoTal Communities Directory and a community_ID will be returned to the community members.
The next step will be to establish the resources that will be shared within the community. Once a user has been added within the community, they will be able to add the community_ID to the list value of the Communities attribute of the resources they want to share within it. As it has been said before, the coding process will be transparent to the user, and they will be easily guided through a friendly interface.
In order that not all the users are able to perform all the possible operations within the community, different roles are defined. The roles within a community describe the rights to perform different operations within the community, i.e., what users will and will not be able to do within the community. For example, a user with an administration role will be able to add or delete users to/from the community, change the roles, delete the community, etc. However, a user with a “user” role will only be able to add or delete their resources, or get information from them. Initially, a small set of different roles with its assigned rights will be available to be selected during the creation of the community and when adding new users.
When the new community has been created, a token to each member of the community will be provided depending the role each one has within it. This token will allow a member to perform different operations (depending on the role played) over the resources within the community or for the management of the community. This token must be included in each request to perform an action, in order to authenticate the requestor and check the proper rights.

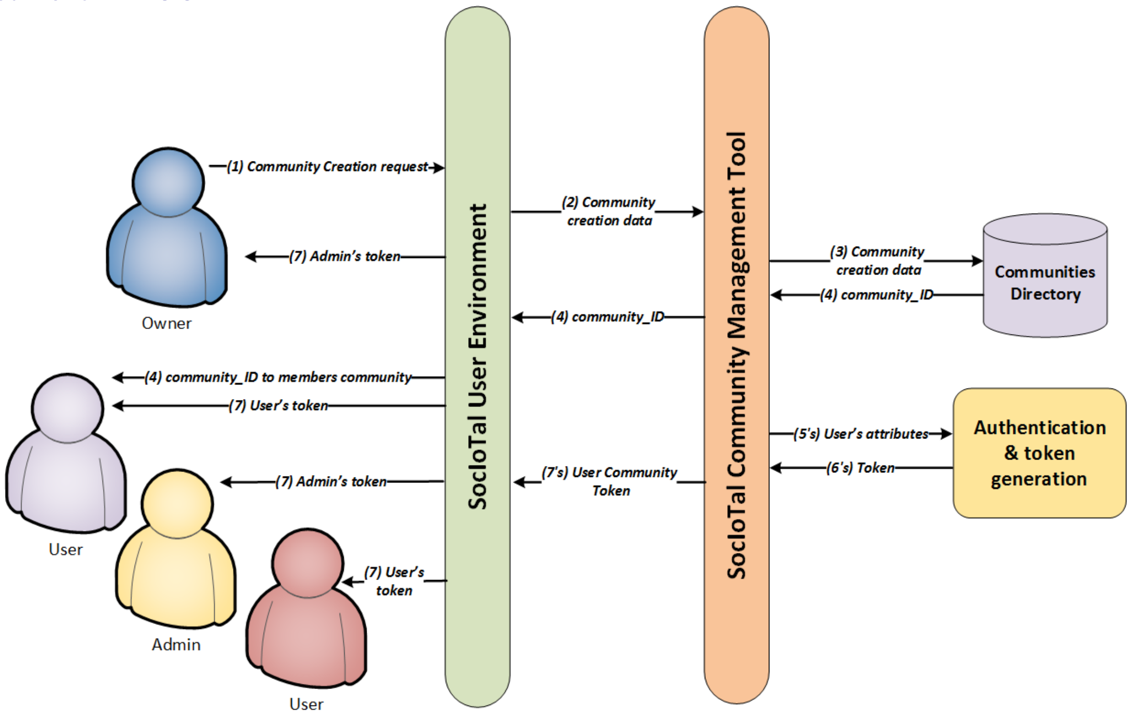
(1) Community Creation request
(7) Admin's token
Owner
(4) community_ID to members community
(7) User's token
User
(7) Admin's token
Admin
(7) User's token
User
SocIoTal User Environment
(2) Community creation data
(4) community_ID
(7's) User Community Token
SocIoTal Community Management Tool
(3) Community creation data
(4) community_ID
Communities Directory
(5's) User's attributes
(6's) Token
Authentication & token generation

Figure 32: Community Creation Process

Figure 32 presents the process of a community creation, which can be summarized as follows:

1. A user fills the template to request the creation of a community through the SocIoTal User Environment. The template will ask for the name of the community, a brief description, owner definition, users to be added and their assigned roles, etc.
2. The data provided by the user arrive to the SocIoTal Community Management tool (CMT)
3. The CMT creates a new entry in the Communities Directory (CD) with the information provided by the user
4. The CD returns the community_ID to the CMT and it to the members
5. With the information of the users provided during the first step the CMT creates some requests to the Authentication block in order to obtain the tokens to the different members of the community (each of them has different characteristics -roles-)
6. The tokens are returned to the CMT
7. The users receive within their profiles through the SocIoTal User Environment the tokens to perform the actions allowed to them within the community (access to resources, management of the community, etc.)

4.2.4 Access to a resource within a community

When a user wants to perform an action over a resource (get information, change an attribute, etc.), the request will have to be accompanied with the token that indicates the rights that the user have within the community where the resource is located. The token will be verified by the corresponding Authorization component before the SocIoTal Context Manager checks if the resources belongs the community. After positive verifications, the Context Manager will carry out the requested action, and will send the correspondent response to the user. In order to understand better how the resource sharing works, below is presented and de-kernelled and example were some users create communities and share resources safely within them.

Figure 33 presents an example of a situation where Alice, Bob and Charlie have created two communities (alfa and beta). Figure 34 and the text bellow shows the process to create alfa and beta and how they share information within the communities.

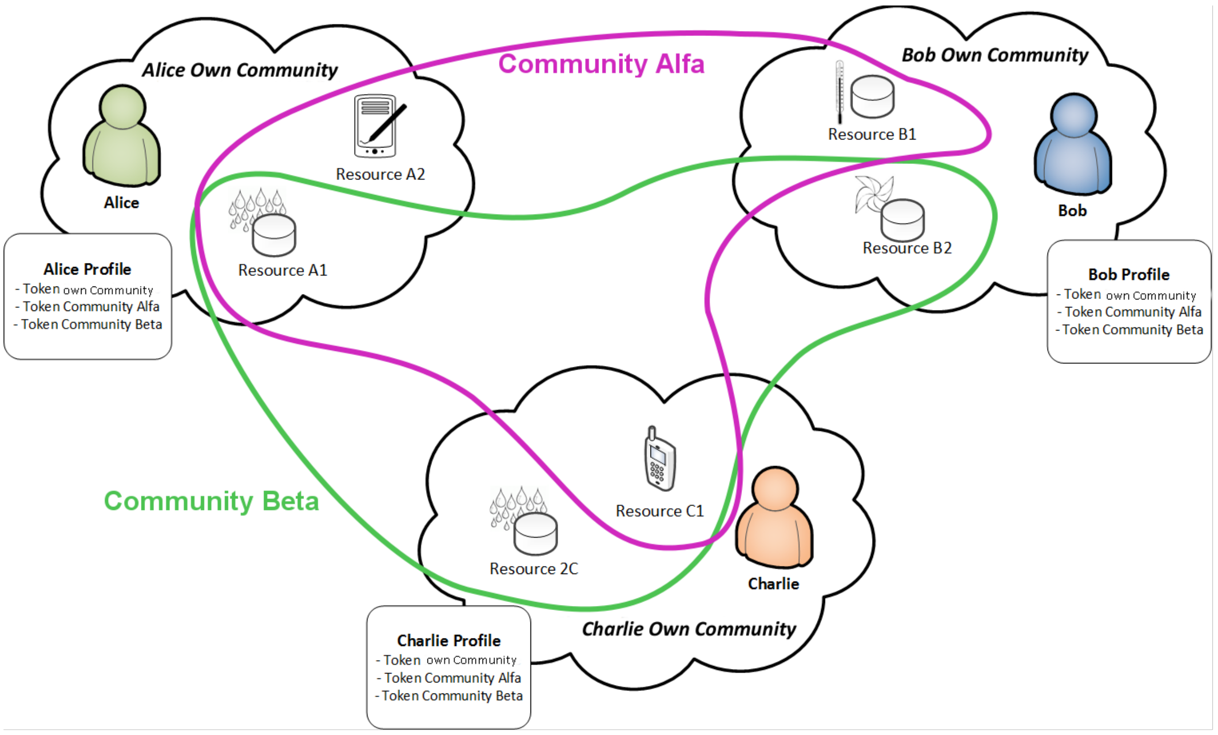

Figure 33: Example of three users who have created two communities alfa and beta

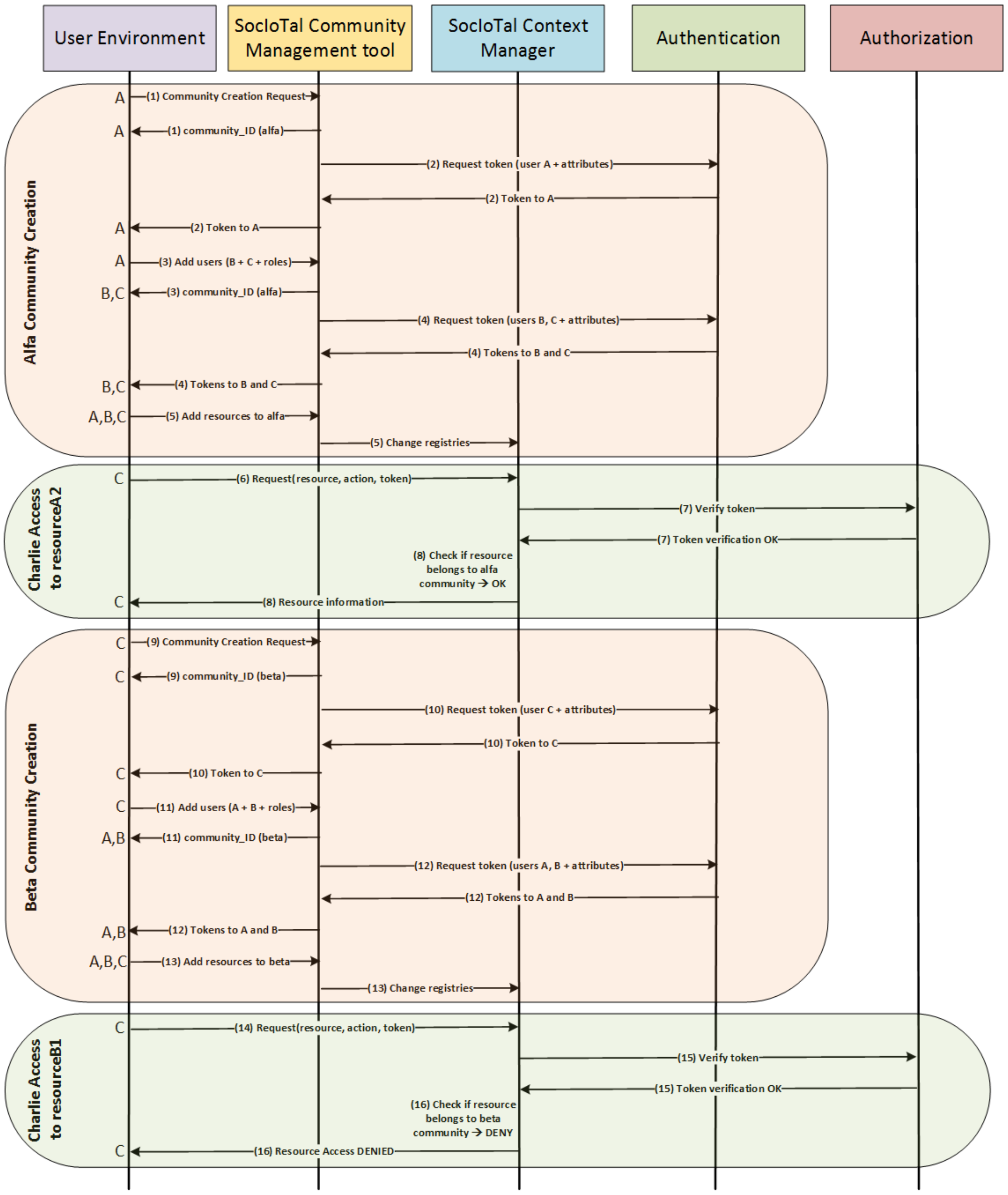
User Environment
SocIoTal Community Management tool
SocIoTal Context Manager
Authentication
Authorization
Alfa Community Creation
A (1) Community Creation Request
A (1) community_ID (alfa)
(2) Request token (user A + attributes)
(2) Token to A
A (2) Token to A
A (3) Add users (B + C + roles)
B,C (3) community_ID (alfa)
(4) Request token (users B, C + attributes)
(4) Tokens to B and C
B,C (4) Tokens to B and C
A,B,C (5) Add resources to alfa
(5) Change registries
Charlie Access to resourceA2
C (6) Request(resource, action, token)
(7) Verify token
(7) Token verification OK
(8) Check if resource belongs to alfa community → OK
C (8) Resource information
Beta Community Creation
C (9) Community Creation Request
C (9) community_ID (beta)
(10) Request token (user C + attributes)
(10) Token to C
C (10) Token to C
C (11) Add users (A + B + roles)
A,B (11) community_ID (beta)
(12) Request token (users A, B + attributes)
(12) Tokens to A and B
A,B (12) Tokens to A and B
A,B,C (13) Add resources to beta
(13) Change registries
Charlie Access to resourceB1
C (14) Request(resource, action, token)
(15) Verify token
(15) Token verification OK
(16) Check if resource belongs to beta community → DENY
C (16) Resource Access DENIED

Figure 34: Description of an example about the creation of communities alfa and beta and resource sharing within them.

Before creating new communities, some premises has been occurred:

- o Users A, B and C are registered within SocIoTal platform
- o Users belongs to their Own Communities (created by default during their registration)
- o Users register their resources within the SocIoTal Context Manager and they are added by default to their Own Communities

Once the premises mentioned before has been accomplished, following our example, the users will create two communities and will try to share information within each of them:

1. A user, Alice, creates a community alfa as explained in Section 4.2.3. Once the community has been created she receives the alfa community_ID.

2. The SocIoTal Community Management tool will request a token for Alice, with the role of administrator (because she is the owner) within the community alfa.

3. Alice wants to add Bob and Charlie to the community alfa assigning them a role "user" within the community. With this role, the users will only be able to access the information from the resources belonging to the community. Once they have been added they receive the alfa community_ID.

4. Alice, Bob and Charlie will receive a token to be able to perform different actions within the community alfa. The token will be created from information such as the community name or id, the user name or id, the role, etc. and each token will indicate the rights each one has depending their attributes.

5. Alice, Bob and Charlie update the registration of those of their devices that they want to share within the community alfa. This modification is performed through the User Environment within the SocIoTal Context Manager.

At this point the community has been created and it has a set of users able to share resources.

6. Now, Charlie wants to access the resourceA2 from Alice, in order to do that he will perform a request to the SocIoTal Context Manager. This request will go with the correspondent token to access the community alfa.

7. The token is verified by the Authorization block to find if the token allow the access to the community alfa.

8. When the SocIoTal Context Manager receive the positive response, it will check if the resource that the user want to access belongs to the community alfa. As Charlie belongs to the community alfa, the SocIoTal Context Manager returns the information requested by Charlie.

9. Now, Charlie creates a new community beta to share other kind of resources and receives the community_ID for beta.

10. The SocIoTal Community Management tool will request a token for Charlie, with the role of administrator within the community beta.

11. Charlie adds Alice and Bob to the community beta assigning Alice an "admin" role, that allows her to perform all possible actions within the community, and assigns Bob a "user" role that allow him to access information provided by the resources belonging to the community. Alice and Bob receives the beta community_ID.

12. Alice and Bob will receive a token corresponding to their attributes within the community (of course, this token will be different from the token assigned to them within the community alfa).

13. Alice and Bob update their resources to define what of their resources will be part or the community beta.

14. Now, Charlie wants to access the resourceB1 from Bob. Within the request to the SocIoTal Context Manager, Charlie attach the token of the community beta.

15. The token is verified by the Authorization block to find if the token allow the access to the community beta. As it is true, the response to the SCM will be positive.

16. The SCM will check now if the requested resource belongs to the community beta. As this premise is not true, the SCM will deny Charlie the access to the resourceB1. However, if he requests again with the token from the community alfa, he will be able to access the resource.

4.3 Secure group communications for bubbles

Unlike communities, bubbles are meant to be dynamic and created spontaneously without the need to create such bubble itself, by means of linking particular users or smart objects to the bubble. Instead, to be part of the bubble, devices or users only have to demonstrate that they satisfy the required identity attributes combination (or policy) for such a bubble. This proving process enables smart objects to obtain the proper cryptographic material associated to their attributes, which in turn allows them to decrypt the data exchanged in the bubble.

Nonetheless, users may need to be able to discover the bubbles along with the identity attributes that are needed to be part of the bubble. To this aim, the bubble can be registered in the centralized Context Manager as a special kind of entity, attaching the identity attributes that will be needed to demonstrate.

The bubble registration allows target devices to be aware of the existence of a bubble, while ensuring, at the same time, security and privacy. Anyone can discover the bubble and the required attributes that are registered in the Context Manager, but only those that satisfy the policy of attributes can decrypt the shared data. Thus, since users can discover the required attributes for that bubble they know the policy of attributes and the keys that must be used to encrypt and decrypt the data exchanged in the bubble.

It should be noticed that the process of obtaining the cryptographic keys is the same for any bubble and it is carried out once, during the first time the user obtains the keys associated to all the attributes of his whole identity. Then, he can use the same cryptographic material for different bubbles that may require a different combination of attributes over the whole identity.

Notice that since users are not directly associated to bubbles, no one can discover the users that are associated to an identity, which is suitable to preserver user's privacy.

4.3.1 Secure Data Sharing for bubbles

Starting from the main bubbles definitions of D2.1 [67] and D2.2 [68], in this section we provide a more detailed overview about the main process for realizing secure group communications. This description is based on the SocIoTal security framework interactions, which were presented in previous section. In addition, a correspondence between such security functional components in hardware elements is provided in order to clarify the main interactions.

On the one hand, Figure 35 shows the process to share information through the use of DCapBAC as a mechanism to get access to a group of smart objects. In this case, it is assumed that there is a bubble (bubble A), which was previously established, and a Issuer of such bubble that is responsible to generate proper credentials to authorized users to be able to get access to the information provided by bubbleA's smart objects. Furthermore, this entity can register the required identity attributes combination to be part of such bubble. Therefore, other smart objects can be aware of the identity attributes that need to be satisfied to be part of the bubble. In the case of the use of DCapBAC for secure group communications, the membership to a bubble is realised by the possession of a DCapBAC token that is associated to that bubble. An example of DCapBAC token that can be employed for this purpose was shown in Section 2.4. More specifically, according to Figure 35, the Issuer entity is endowed with a *Capability Manager*, as the entity responsible for generating DCapBAC tokens for bubbles, and an Idemix Verifier, in order to enable an optional anonymous authentication to get such token.

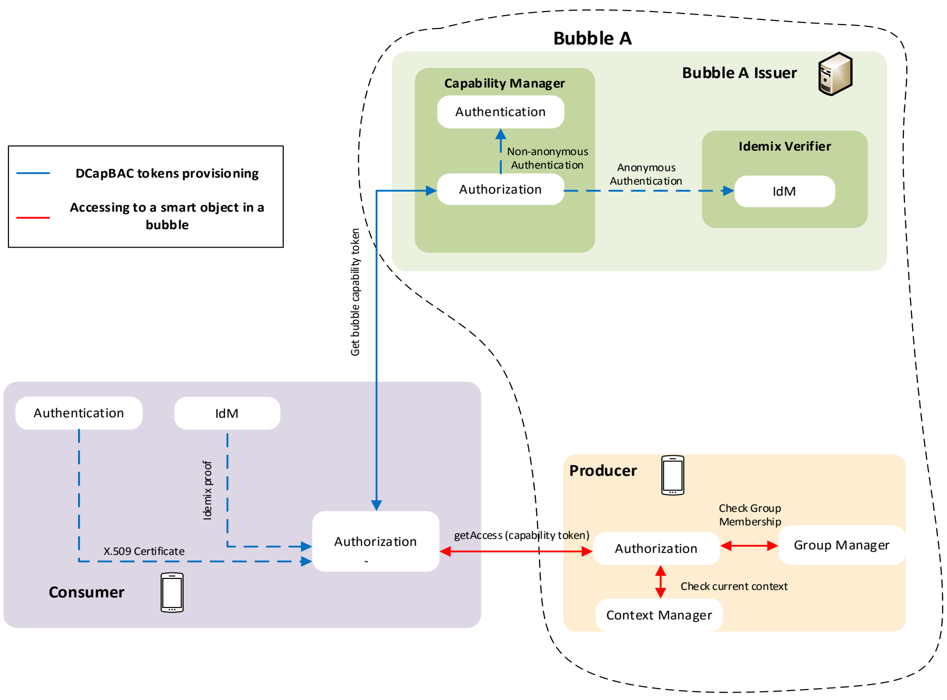

Figure 35. DCapBAC-based Secure Data Sharing for SocIoTal bubbles

The main flow of the scenario starts with a *consumer* entity trying to get access to a resource being hosted in a *producer* entity within bubbleA. For this purpose, it contacts with the Issuer of bubbleA, to get a suitable DCapBAC token. The process for obtaining the token could follow a similar approach to the description provided in Section 2.2 to get CP-ABE keys. As the Issuer stores the identity attributes that are required to be part of bubbleA, the consumer can prove that specific attributes combination in a privacy-preserving way. For that, it can use an Idemix proof (generated from its credential) demonstrating that it is an entity with those attributes. On the contrary, the consumer can make use of more traditional authentication schemes (e.g., by making use of its X.509 certificate in a TLS exchange) to authenticate against the Issuer. However, in this case, the consumer`s privacy would not be preserved. After the consumer is successfully authenticated (preserving or not its privacy), the *Authorization* functional component of the framework (instantiated by the Capability Manager element) is responsible for generating the DCapBAC token. This token is associated to get access to resources being hosted by devices in bubbleA. It should be noticed that this process could require the intervention of a *Policy Decision* Point (PDP), which is in charge to evaluate authorization policies denying or granting the access to the bubble.

Once the consumer device obtains the proper DCapBAC token, it can make use of it to get access to devices within bubbleA. Specifically, according to the figure, this credential is attached in the access request, and evaluated by the producer entity. In this case, this entity should check if it itself is part of the bubble specified in the token (see Figure 12), in order to deny or grant the access. Furthermore, it could check contextual conditions, which could be optionally defined in the token, are fulfilled when receiving the access request.

Moreover, Figure 36 shows the scenario for secure group communications based on the CP-ABE cryptographic scheme. Unlike symmetric key based approaches, in this case there is no need to generate new keys to enable a secure sharing within a bubble. Indeed, information can be encrypted under different CP-ABE policies and decrypted by the same CP-ABE key. In this way, during the sharing process, any third party is involved, enabling a secure and ad hoc communication between devices. For this scenario, two entities are considered; a *producer* that is intended to share or communicate data, and a *consumer*, as the device that wants to get access to information provided by producers. It should be noticed that, although it is not shown for the sake of clarity, both entities have already obtained the cryptographic material required to realize the CP-ABE based group communication. For this purpose, such entities can make use of any of the mechanisms described in Section 2.2. In addition, D2.2 [68] specified the main required interactions to generate CP-ABE keys.

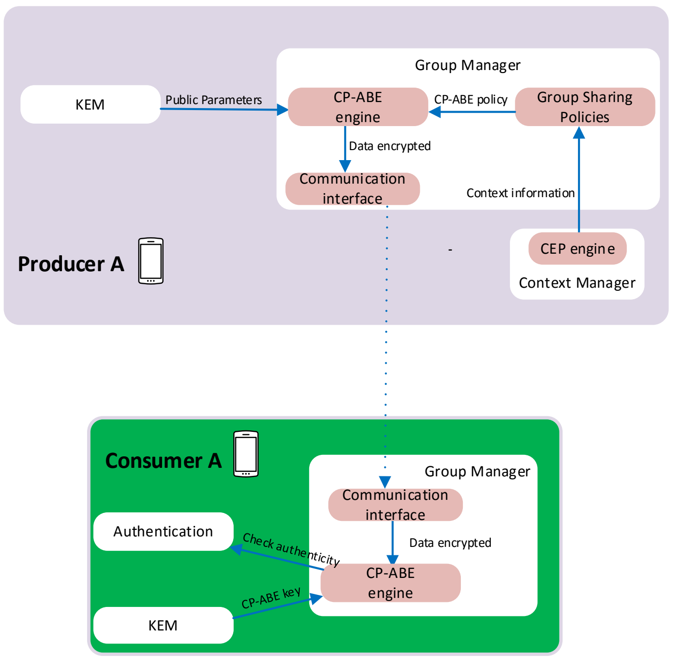

Figure 36. CP-ABE based Secure Data Sharing for SocIoTal bubbles

According to this figure, the main flow of the scenario starts with a producer entity sharing information, which is encrypted with a specific CP-ABE policy. For the selection of this policy, the producer can make use of the *Group Sharing Policies* subcomponent, as part of the Group Manager, in order to select a proper CP-ABE policy. These policies are evaluated according to contextual information that is detected by the Context Manager functional component. Such information can be received from other smart objects or the centralized SocIoTal Context Manager. Optionally, the Context Manager can be provisioned with a CEP engine, in order to infer or reason high-level contextual information. The result of the evaluation of the Group Sharing Policies is, in turn, a CP-ABE policy, which is used by the CP-ABE engine subcomponent to encrypt data. In addition, this process requires the Public Parameters, that are delivered by the Attribute Authority and stored in the KEM functional component. Once information is encrypted, a *Communication interface* subcomponent is responsible for communicating the corresponding data. This interface can be based on protocols, such as CoAP or Bluetooth, in the case of more opportunistic bubbles.

After the consumer smart object receives the encrypted data through its Communication interfaces, such data is sent to the CP-ABE engine subcomponent in order to try the decryption process. For this purpose, the task of this entity is twofold. On the one hand, it checks against the Authentication component if data comes from a legitimate user (e.g. using a signature scheme). Optionally, it can contact with other functional components, such as the Trust & Reputation component, in order to assess of the information received comes from a trustworthy entity. On the other hand, it queries the KEM component in order to get a suitable CP-ABE key to decrypt the information being received. In this way, if this component stores a key satisfying the CP-ABE policy that was used to encrypt data, the consumer entity will be able to get access to such information.

4.3.2 CP-ABE based Publish/Subscribe Secure Data Sharing

Taking into account the requirements and challenges of typical IoT scenarios, the well-known publish/subscribe interaction pattern is receiving a significant interest from academia and industry, in order to realize different IoT use cases. On the one hand, this pattern allows smart objects to be decoupled, which is a relevant feature in scenarios with a potential huge number of entities interacting each other. On the other hand, it allows information to be shared with a group of entities through the use of a single message, which is sent to the broker entity.

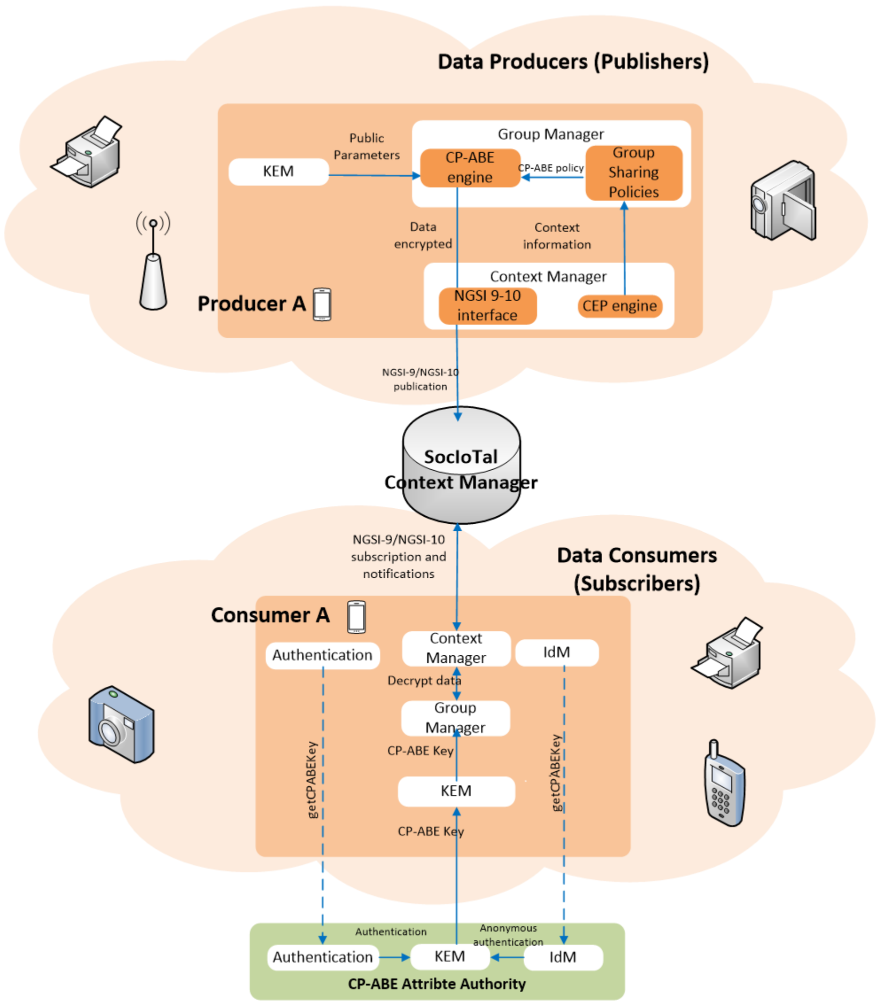

Figure 37. CP-ABE based secure group communication through publish/subscribe

In this way, Figure 37 shows the scenario in which the CP-ABE cryptographic scheme is applied for secure group communications through the use the publish/subscribe pattern. Specifically, this scenario is built on top of the SocIoTal Context Manager that is employed as a broker to communicate information coming from producers to potential consumer entities. Although the figure shows producer and consumers as separate entities, the same entity can act as a publisher (producer), or subscriber (consumer). In addition to such entities, a CP-ABE Attribute Authority is used as the main entity to generate and deliver the proper CP-ABE keys, according to the description provided in Section 2.2.

As it is shown in the previous figure, producers and consumers make use of the functionality provided by the Group Manager, in a similar way to the previous scenario. However, in addition to that case, the Context Manager, which is deployed on devices, is provisioned with NGSI-9/NGSI-10 API interfaces, in order to enable the communication with the SocIoTal Context Manager.

Payload:
```
{
    "entities": [
    {
        "type": "bedroom",
        "isPattern": "false",
        "id": "Hab1"
    }
    ],
    "attributes": [
        "temperature"
    ],
    "reference": "http://83.33.158.192:9000/entity",
    "duration": "PT5M",
    "notifyConditions": [
    {
        "type": "ONCHANGE",
        "condValues": [
            "temperature"
        ]
    }
    ],
    "throttling": "PT5S"
}
```

Figure 38. Subscription message example for CP-ABE based group communication

Specifically, consumer entities subscribe against such entity to receive data from producers. An example of this message is shown in Figure 38. Furthermore, Figure 39 and Figure 40 show an example about how the information that is shared with the broker is communicated under the NGSI-9/NGSI-10 API through update and notification messages, respectively.

Payload:
```
{
    "contextElements": [
    {
        "type": "bedroom",
        "isPattern": "false",
        "id": "Hab1",
        "attributes": [
        {
            "name": "temperature",
            "type": "string",
            "value": "temperature_value_ecncryptedd",
            "metadatas": [
            {
                "name": "cph",
                "type": "string",
                "value": "encrypted_value"
            },
            {
                "name": "signature",
                "type": "string",
                "value": "signature_value"
            },
            {
                "name": "pub",
                "type": "string",
                "value": "public_key_value"
            }
            ]
        }
        ]
    }
    ],
    "updateAction": "UPDATE"
}
```

Figure 39. Update message example for CP-ABE based group communication

```
Payload:
{
    "subscriptionId" : "51c0ac9ed714fb3b37d7d5a8",
    "originator" : "localhost",
    "contextResponses" : [
    {
        "contextElement" :
        {
            "attributes" : [
            {
                "name" : "temperature",
                "type" : "string",
                "value" : "temperature_value_encrypted",
                "metadatas": [
                {
                    "name": "cph",
                    "type": "string",
                    "value": “encrypted_value"
                },
                {
                    "name": "signature",
                    "type": "string",
                    "value": "signature_value"
                },
                {
                    "name": "pub",
                    "type": "string",
                    "value": "public_key_value"
                }
                ]
            }
            ],
            "type" : "bedroom",
            "isPattern" : "false",
            "id" : "Hab1"
        },
        "statusCode" :
        {
            "code" : "200",
            "reasonPhrase" : "OK"
        }
    }
    ]
}
```

Figure 40. Notification message example for CP-ABE based group communication

Section 5 - Evaluation Results

This section delivers some experimental results carried out with the aim of validating and showing the feasibility of the proposed secure sharing mechanisms. Firstly, the section shows the performance results obtained to generate and deliver group keys used to share data using symmetric encryption. Then, this section also devotes a subsection about the evaluation of the process aimed to communicate information securely by mean of the CP-ABE cryptographic scheme. Namely, these evaluations have been performed relying on the SocIoTal Context Manager in order to share information with groups of entities in a secure fashion, through the well-known publish-subscribe pattern.

5.1 Physical layer key generation for group key distribution results

The evaluations presented in this section are based on Matlab simulations, considering IR-UWB channel probes according to the following equations:

$$\boldsymbol{y_{AB}(t) = (h_{AB} * p)(t) + n_B(t)}$$

$$\boldsymbol{y_{BA}(t) = (h_{BA} * p)(t) + n_A(t)}$$

where $h_{AB}= h_{BA}$ is the reciprocal channel impulse response generated using the IEEE 802.15.4a statistical model for LOS indoor environments (CM1), $p(t)$ is the transmitted pulse waveform (bandwidth 1 GHz, center frequency 4.5 GHz), $n_i(t)$ is the thermal noise at receiver i according to the SNR defined as the ratio of the pulse power and the noise power:

$$\boldsymbol{SNR = \frac{P_{pulse}}{P_{noise}}}.$$

The duration of the observation window is set at 50 ns (usual duration at UWB Rx so as to capture most of the multipath energy in typical indoor environments) and the start sample in the observation window at each node (A or B) is determined independently using a threshold crossing rule (i.e., emulating imperfect synchronization based on channel leading edge detection between A and B acquisitions). The received signal is then processed by integration over an interval of half the duration of a pulse and normalized to the maximum value (i.e., w.r.t. to the maximum energy sample observed over the observation window).

The quantization cells are $\{(0 - 0.25), (0.25 - 0.5), (0.5 - 0.75), (0.75 - 1)\}$ and the corresponding binary dictionary is $\{'00', '01', '11', '10'\}$. The employed guard-bands vary between 0 and 0.05 (GB) around the thresholds of the quantization cells and the samples that fall in these bands are dropped. Reconciliation is achieved with a Reed Solomon code with a message length of 127 codewords and data length 124 codewords after padding the quantized binary sequence with dummy bits in order to obtain the needed block length for encoding.

The simulations presented herein take into consideration a group of three nodes in a full mesh configuration and only one channel probing on each directed link (i.e., three links in total). The results are obtained by averaging over 1000 three-node configurations with independent channels.

If the channels change, several probes can be concatenated and longer keys could be generated on each link of the mesh (this case is not studied herein). However, the main considerations about key generation are the same:

- Reciprocity measured by the key agreement ratio (i.e., the ratio of agreed upon keys after reconciliation, considering the three links of the mesh in the same time) and bit matching rate (the ratio of identical bits before reconciliation averaged over the three links). The sources of mismatching are represented by the effect of the noise on the signal amplitudes and on the synchronization.

- The length of the group key that can be distributed after the establishment of the three keys which is limited by the second shorter key. A small guard-band for quantization implies less dropped samples, which implies more bits per key but also a smaller bit matching rate.
- Random aspect of the keys. As the samples of the quantized signal can be correlated or do not have sufficient dynamics, the generated keys can have pattern defects. In this context, we measure the frequency of apparition of '1' over all generated keys.

Even though the bit matching rate is relatively high in all configurations (Figure 41), the probability of generating three successful keys in one trial per link is relatively small for small guard-band intervals (Figure 42). The key agreement is expected to deteriorate with the increase in the number of nodes in the mesh. Alternative techniques that would allow the cooperative generation of the group key directly at the physical layer with the help of signal processing techniques could improve the key agreement rate.

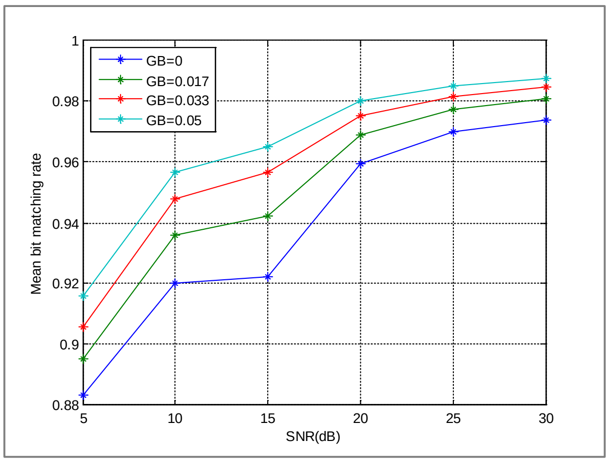

Figure 41: Mean bit matching ratio for different guard-band intervals

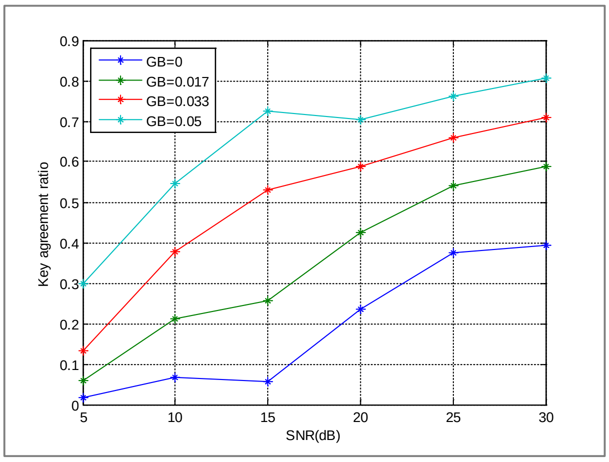

Figure 42: Key agreement ratio simultaneously for different guard-band intervals

Figure 43 represents the length of valid keys, which, as expected, decreases with the increase in the guard-band interval. As the this metric has been computed only on the configurations in which there was key agreement over all of the three links, the fluctuations in the results at low SNR (5-15 dB) can be explained by the small sample size used for averaging.

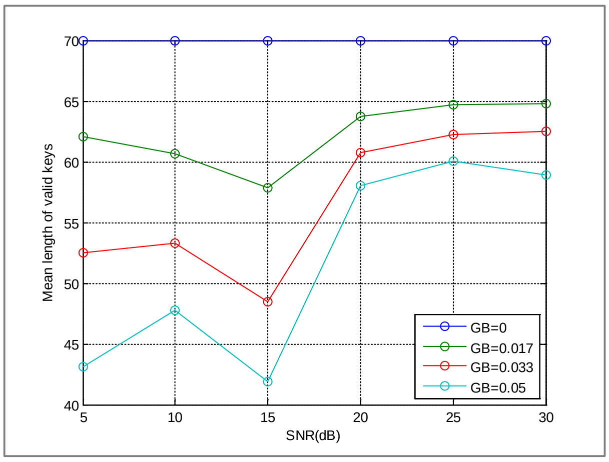

Figure 43: Mean length of possible group keys

Finally, we evaluate the frequency of apparition of ‘1’ in each generated bit sequence over all channels and configurations and plot the histograms of these values. Preferably, this value is around 0.5 (i.e., equal number of ‘1’ and ‘0’ in a key). This metric is influenced by the dynamics of the signal (i.e., whether there are sufficient fluctuations in the quantized signal) and by the particular quantization dictionary. We conclude that all guard-band intervals have the same behavior, while the SNR and therefore the signal dynamics influence a lot on the results. At low SNR (Figure 44), after normalization, we find a high proportion of ‘1’ in the binary sequences, corresponding to a preponderant quantization on the last cells (i.e., the signal is too uniform). At high SNR (Figure 47), we find the reverse, meaning that apart a few samples that will “stand out” and be quantized in the last cell of the quantization grid, a lot of the remaining samples will be quantized in the first quantization cells. Medium SNR levels provide good diversity in terms of frequency of apparition of ‘1’ (Figure 45, Figure 46) but are not, however, free from other predictable aspects (e.g., too regulate oscillations between ‘1’ and ‘0’).

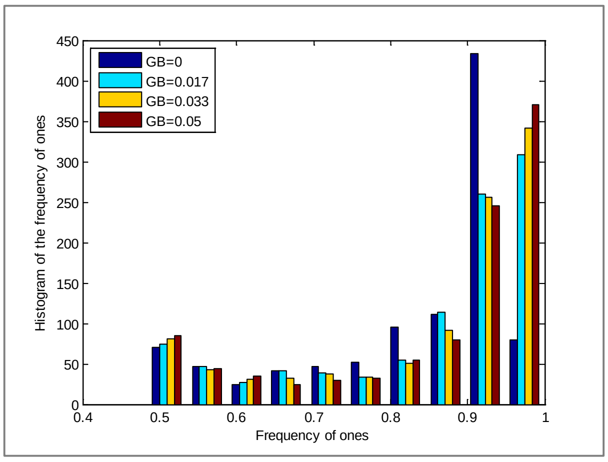

Figure 44: Histogram of frequency of '1' at SNR = 5 dB

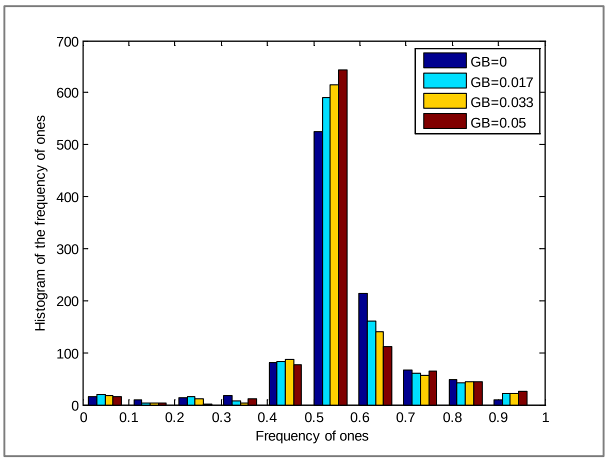

Figure 45: Histogram of frequency of '1' at SNR = 10 dB

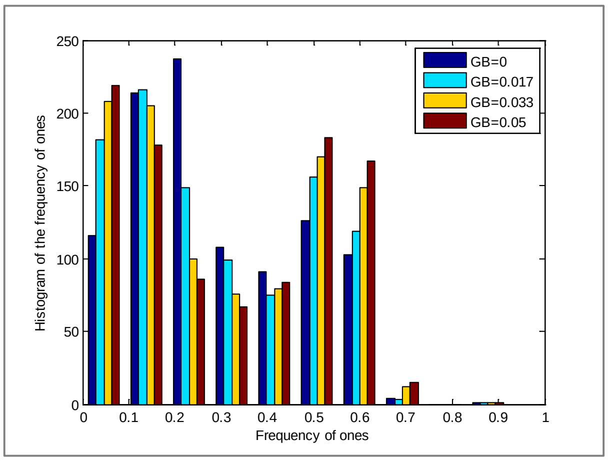

Figure 46: Histogram of frequency of '1' at SNR = 15dB

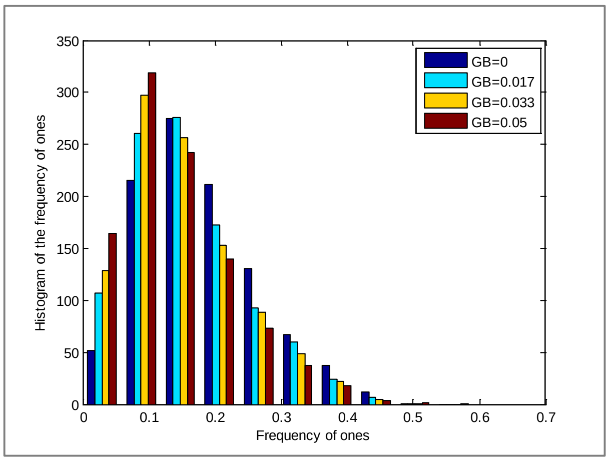

Figure 47: Histogram of frequency of '1' at SNR = 20 dB. Similar results for higher SNR values

In conclusion, IR-UWB signals provide a richer signal in terms of number of samples from one channel probing, but the direct quantization of this type of signals still suffers from defects in the random aspect of keys. One solution would be to use decorrelation transformations before quantization, which is also applicable to RSS measurements. This option, unfortunately, will also have an impact on the reciprocity of the measurements. Another solution would be to estimate other channel metrics that are less sensitive to correlation between samples (e.g., estimate the time of arrivals of the multipaths and quantize this information instead of the amplitude information).

Again, due to a patent application in progress regarding related concepts, in this section, we have only accounted here for preliminary evaluations of an initial solution enabling group key generation using the IR-UWB physical layer. Further evaluations, still based on the same Matlab synthetic environment and possibly real channel estimates (i.e., produced at integrated radio devices) shall be reported as soon as possible in one suitable upcoming SocIoTal deliverable, according to the patent application acceptance and timeline. However it is to be noted that this non-conventional solution will not be physically implemented in SocIoTal.

5.2 Secure Data Sharing experiment results

According to the publish/subscribe secure data sharing scenario presented in the Section 4.3.2, below we provide a performance analysis taking into account different practical aspects (e.g., runtime, memory consumption, etc.), in order to evaluate the feasibility of this mechanism. It should be pointed out that this scenario has been tested by making use of the SocIoTal Context Manager acting as a broker entity. Producer and consumer entities were implemented as v.5 (Lollipop) Android applications on a Smartphone LG-G3 with a Qualcomm MSM8975AC Snapdragon 801 chipset, a Quad-core 2.5 GHz Krait 400 CPU, and 2 GB RAM. This evaluation was carried out by making use of PowerTutor [62] and System Monitor applications.

Before the discussion of the experimental results, it is required the explanation of some aspects related to the CP-ABE scheme that is used in this scenario. Specifically, performance results are mainly influenced by the desired security level (that depends on the cryptographic parameters being employed), as well as the number of attributes used to define the access policy. We deployed a Java-based CP-ABE library on Android, which we used to implement the core functionality of the sharing scheme. In particular, this library is based on the functionality provided in [63], which implements the CP-ABE scheme provided by [36]. It is built on top of the Java Pairing Based Cryptography library (jPBC) [64], and uses type A pairings, which are built on the supersingular curve $y^2 = x^3 + x$ over the field F_p for some prime $p = 3 \bmod 4$. In this case, let p be the prime order of F_p, and $E(F_p)$, the additive group of points of affine coordinates (x, y) with x, y in F_p, that satisfy the curve equation, q represents the order of the cyclic subgroup of interest in $E(F_p)$. As stated in [65], the security level of the scheme depends on the size of primes p and q. It should be pointed out that this evaluation considers an 80-bits security level (i.e., $|p| = 512$, $|q| = 160$), as the minimum security level recommended by NIST [66]. Furthermore, the policies defined for this evaluation only consider AND connectives for one level of attributes. Therefore, the CP-ABE key used by the subject device to resolve the challenge, should be associated to (at least) all the attributes that were used to define the access policy in the encryption process.

Under these considerations, Figure 48 shows the delay required by a producer entity to generate a Context Manager message (e.g., an update of an entity). These results were obtained by averaging the values of 10 executions for such operations. The series “Encrypt” makes reference to the delay required to encrypt data under a specific attribute policy. Moreover, the series “Other operations” is related to other required functionality for the scenario (e.g., message signature, Base64 encoding or HTTP packet generation). In addition, these results were obtained by modifying the number of attributes defined in the access policy from 1-10, since we consider this range expressive enough for most scenarios and use cases. Thus, in the case of a 1 attribute access policy, the time required to generate a message takes 1547 ms, while for a 10-attribute policy, it takes around 8288 ms.

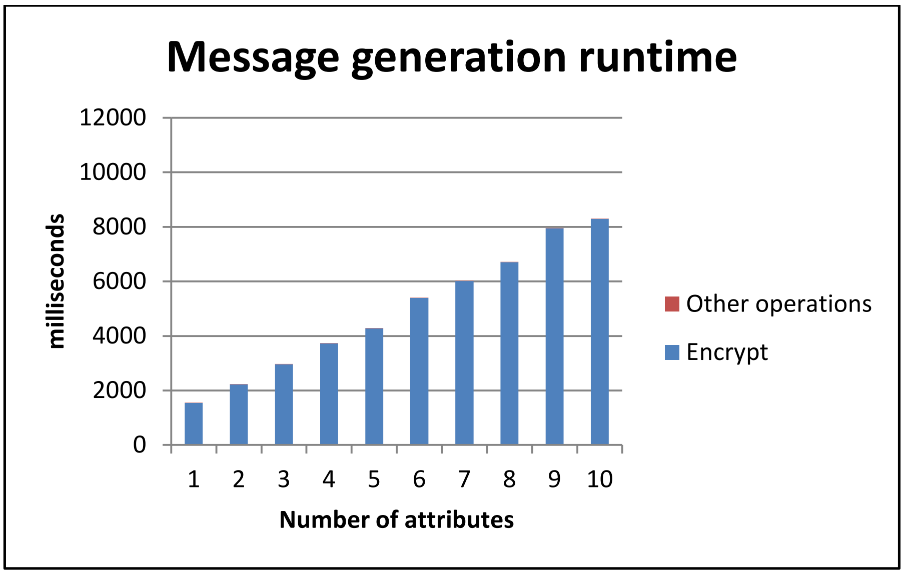

Figure 48. Runtime to generate a Context Manager message (Producer)

Regarding the process related to the reception and decryption of a message, which is carried out by a consumer entity, Figure 49 shows the delay required for that process. In this case, it is assumed, a specific consumer entity is subscribed to the SocIoTal Context Manager in order to receive the corresponding notifications from a producer entity. The series "Decrypt" makes reference to the delay required to decrypt data making use of CP-ABE key satisfying the CP-ABE policy that was used to encrypt. In addition, the series "Other operations" is related to other required functionality for the scenario (e.g., ECDSA signature validation, Base64 decoding or HTTP packet parsing). In the case of a 1 attribute access policy, the time required to receive and decrypt data is around 1784 ms, while for a 10-attribute policy, it takes around 11543 ms by using a 10-attribute CP-ABE key satisfying that policy. It should be pointed out that, in case CP-ABE key does not satisfy the policy, this time will be significantly lower.

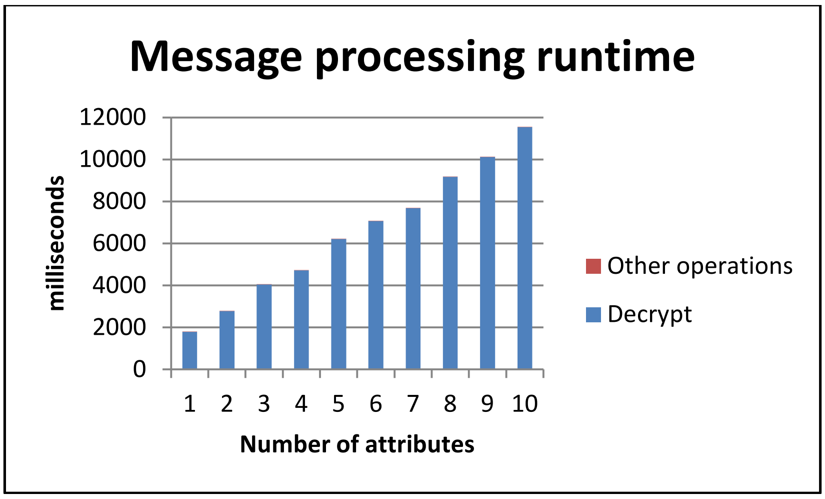

Figure 49. Runtime to process and decrypt a Context Manager message (Consumer)

In addition to the execution time, we also measured the average memory space that is required by the application, in the case of producer and consumer entities. Like in the case of the runtime results we use a range between 1 and 10 attributes. Thus, Figure 50 shows the memory space required by a producer entity to generate a message by fluctuating the number of attributes. According to it, depending on the number of attributes used to specify the access policy, the required memory is between 75 and 118 MB, which is reasonable for the most of current smartphones.

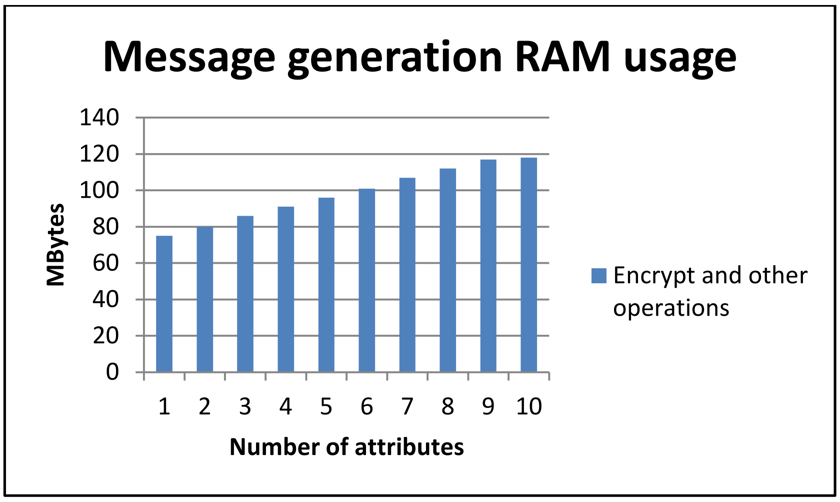

Figure 50. RAM usage to generate a Context Manager message (Producer)

In the case of a consumer entity, Figure 51 shows the memory space required to receive and decrypt the message from the SocIoTal Context Manager. According to the results, this process is heavier than the generation procedure, and it requires between 81 and 127 MB, for 1-attribute policy and 10-attributes policy, respectively.

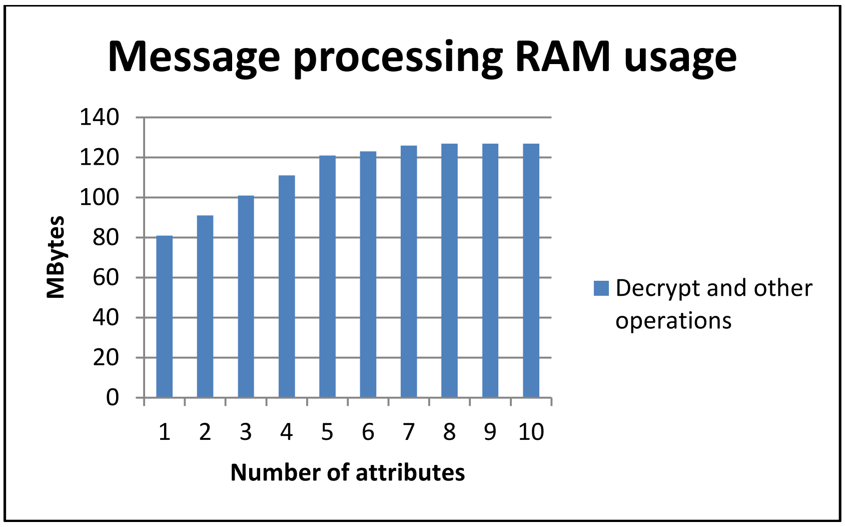

Figure 51. RAM usage to process and decrypt a Context Manager message (Consumer)

The aspects related to energy consumption are a major concern for IoT devices. Because of it, we provide a set of experimental results regarding the energy consumption of the producer and consumer applications. In the case of a producer entity, Figure 52 shows the energy consumption related to the process to generate an encrypted message to the SocIoTal Context Manager requires between 1 and 8,86 J. These values are obtained, again, by fluctuating the number of attributes that are used to specify the access policy.

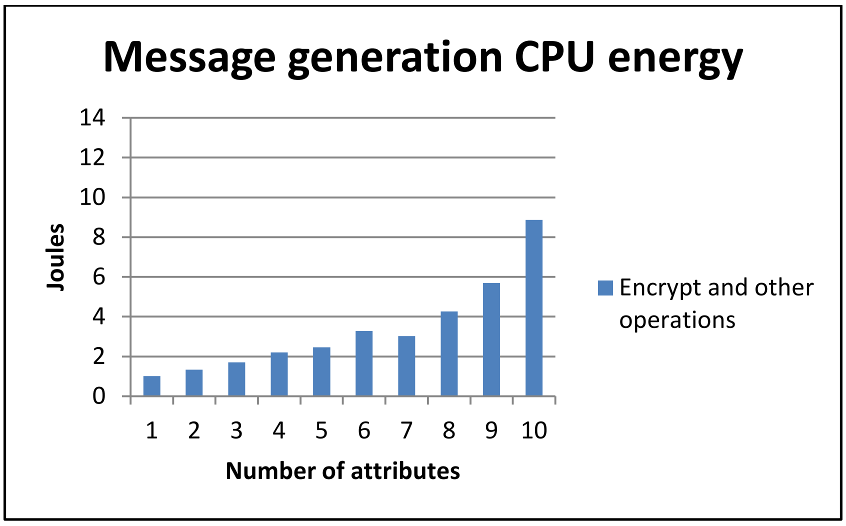

Figure 52. Energy consumption to generate a Context Manager message (Producer)

For a consumer entity, Figure 53 shows the energy consumption related to the process required to process the message, which is sent from the SocIoTal Context Manager. For these operation, the CPU energy fluctuates between 1,4 and 13,9 J, by considering different numbers of attributes to specify the access policy.

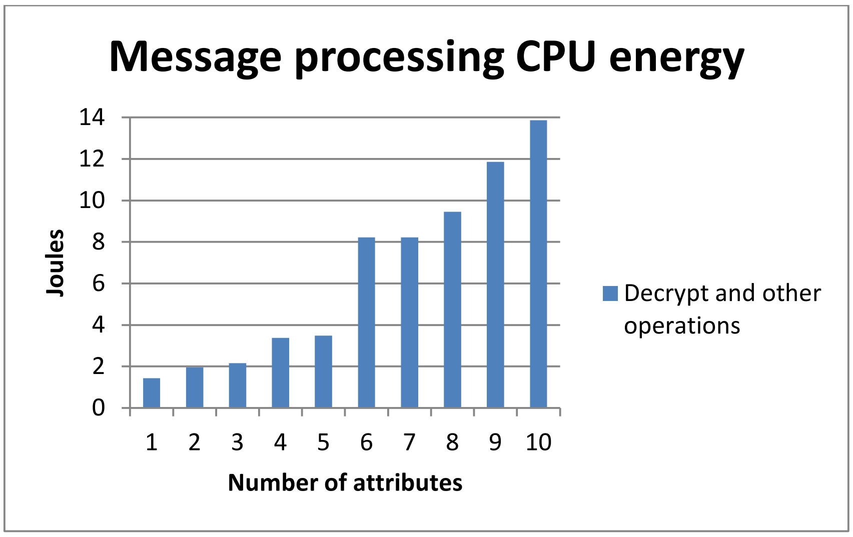

Figure 53. Energy consumption to process and decrypt a Context Manager message (Consumer)

According to these results, the energy consumption for both applications remains low, which makes this mechanism suitable for this kind of devices. An aspect to be considered about these results is, as it is shown, the energy consumption shoots up under a specific number of attributes. For example, in the case of a consumer entity (Figure 53) the energy consumption goes from 3.48 J (with 5 attributes) to 8.22 (with 6 attributes). This fact is due to the operating system assigns a maximum RAM size for the application, and consequently, when it reaches this size, the operating system uses a paging memory-management scheme, retrieving data from the secondary storage. In this way, when the RAM stabilizes (i.e., it reaches the maximum allocated for the application), the energy consumption is greatly increased. Indeed, as it can be seen in Figure 51, the required RAM is almost stable when 6-attributes policies are used.

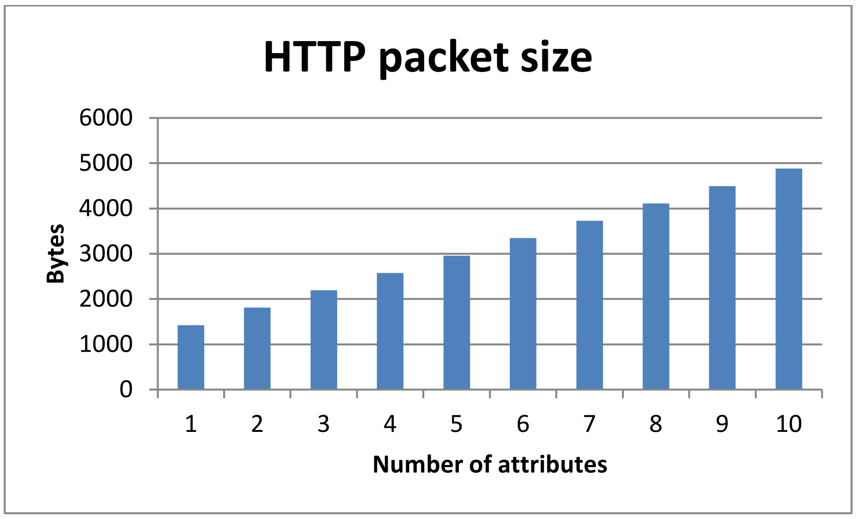

Figure 54. HTTP packet size depending on the attributes of acces policy

An additional aspect to be considered is related to the increase in the size of the message when the CP-ABE cryptographic scheme is used. In fact, for this scheme, the size of the ciphertext directly depends on the number of attributes, which are used to specify the access policy that is, in turn, employed to encrypt the information. This fact is reflected in Figure 54. According to these results, depending on the number of attributes for the same message, the size increases considerably, ranging from 1424 bytes by using a 1-attribute policy, to 4882 bytes for the same message being encrypted with a 10-attributes policy.

Section 6 - Conclusions

This deliverable has provided the description of the main interactions of the SocIoTal security framework, in order to realize different mechanisms for secure group communications. The concept of group is an important aspect to be considered in IoT scenarios, as a way to manage the communications among billions of interconnected smart objects. Starting from the definition of the ARM-compliant security framework presented in D2.1, as well as the main security and privacy aspects reported in D2.2, in this deliverable different low-level encryption techniques have been explored as a foundation to enable secure communication in the envisioned SocIoTal communities and bubbles. On the one hand, different options for generation and delivery of group keys have been presented by considering the use of Symmetric Key Cryptography. On the other hand, other recent schemes, such as IBE and CP-ABE have been analysed as a more flexible (and heavier) alternative enabling these communications. In the case of CP-ABE, this document has provided an overview of different alternatives addressing the whole lifecycle of the scheme.

The proposed mechanisms have been framed within the SocIoTal security framework, which is intended to provide a holistic security and privacy architecture in order to foster large-scale IoT deployments. In this way, the main interactions of the security functional components of such framework have been provided to enable the realization of the secure group communications for SocIoTal. These mechanisms have been designed by considering contextual information provided by different SocioTal enablers in order to enrich group communications with context awareness features.

In addition, this deliverable have provided a set of experimental results related to the secure group communication mechanisms. Specifically, the physical layer key generation for group key distribution mechanism has been evaluated through the use of Symmetric Key Cryptography. This process is a low-level and lightweight alternative to the application of the CP-ABE scheme for secure group communications. In this case, the CP-ABE secure group mechanism has been evaluated on Android devices through the well-known publish-subscribe pattern. This scenario has been developed by using the SocIoTal Context Manager, in order to evaluate its feasibility with a set of experimental results under the main foundations and developments of the project.

References

[1] Mittra, S. (1997, October). Iolus: A framework for scalable secure multicasting. In *ACM SIGCOMM Computer Communication Review* (Vol. 27, No. 4, pp. 277-288). ACM.

[2] Shields, C., & Garcia-Luna-Aceves, J. J. (1999). *KHIP—a scalable protocol for secure multicast routing* (Vol. 29, No. 4, pp. 53-64). ACM.

[3] Challal, Y., Bettahar, H., & Bouabdallah, A. (2004). SAKM: a scalable and adaptive key management approach for multicast communications. *ACM SIGCOMM Computer Communication Review*, *34*(2), 55-70.

[4] Ballardie, T. (1996). Scalable multicast key distribution (RFC 1949). *IETF Request For Comments*.

[5] Ballardie, A. (1997). Core based trees (CBT) multicast routing architecture. (RFC 2201). *IETF Request For Comments*.

[6] Hardjono, T., Cain, B., & Monga, I. (2000). *Intra-domain group key management for multicast security*. IETF Internet draft.

[7] Setia, S., Koussih, S., Jajodia, S., & Harder, E. (2000). Kronos: A scalable group re-keying approach for secure multicast. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on* (pp. 215-228). IEEE.

[8] Briscoe, B. (1999). MARKS: Zero side effect multicast key management using arbitrarily revealed key sequences. In *Networked Group Communication* (pp. 301-320). Springer Berlin Heidelberg.

[9] Harney, H., & Muckenhirn, C. (1997). Group key management protocol (GKMP) specification. (RFC 2093). *IETF Request For Comments*.

[10] Harney, H., & Muckenhirn, C. (1997). Group key management protocol (GKMP) architecture. (RFC 2094). *IETF Request For Comments*.

[11] Wallner, D., Harder, E., & Agee, R. (1999). *Key management for multicast: Issues and architectures*. (RFC 2627). *IETF Request For Comments*.

[12] Harding, M., McGrew, D. A., & Sherman, A. T. (1997). A new key-management algorithm for large dynamic groups. *Tansparencies form talk given by Alan Sherman at NSA (November 19, 1997)*.

[13] U. Maurer, Secret Key Agreement by Public Discussion from Common Information, IEEE Trans. on Information Theory, vol. 39, pp. 733-742, May 1993.

[14] N. Patwari, J. Croft, S. Jana, and S. K. Kasera, High-Rate Uncorrelated Bit Extraction for Shared Secret Key Generation from Channel Measurements, IEEE Trans. on Mobile Computing, vol. 9, pp. 17-30, Jan. 2010.

[15] S. Severi, G. Abreu, G. Pasolini, and D. Dardari, A Secret Key Exchange Scheme for Near Field Communication, in Proc. IEEE Wireless Commun. Netw. Conf. (WCNC), Sept. 2014.

[16] Y. El Hajj Shehadeh, O. Alfandi, and D. Hogrefe, Towards Robust Key Extraction from Multipath Wireless Channels, Journal of Communications and Networks, vol. 14, pp. 385_395, Aug. 2012.

[17] G. Pasolini and D. Dardari, Secret key generation in correlated multi-dimensional Gaussian channels, in Proc. IEEE Int. Conf. Commun. (ICC), pp. 2171-2177, June 2014.

[18] I. Tunaru, B. Denis, and B. Uguen, Reciprocity-Diversity Trade-off in Quantization for Symmetric Key Generation, in Proc. PIMRC'14, (Washington DC, US), Sept. 2014.

[19] F. Marino, E. Paolini, and M. Chiani, Secret Key Extraction from a UWB Channel: Analysis in a Real Environment, in Proc. IEEE Int. Conf. Ultra-Wideband (ICUWB), Sept. 2014.

[20] M. Pezzin and D. Lachartre, A low Power, Low Data Rate Impulse Radio Ultra Wide Band Transceiver, in Proc. FUNEMS'10, (Florence, Italy), June 2010.

[21] W. Burr, D. Dodson, and W. Polk, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, tech. rep., Information Technology Laboratory, NIST, Gaithersburg, Maryland, 2010.

[22] M. Madiseh, M. McGuire, S. Neville, L. Cai, and M. Horie, Secret Key Generation and Agreement in UWB Communication Channels, in Proc. IEEE GLOBECOM'08, (New Orleans, LO, USA), pp. 1-5, Dec. 2008.

[23] L. Lai, Y. Liang, and W. Du, Cooperative Key Generation in Wireless Networks, IEEE Journal on Selected Areas in Communications, vol. 30, pp. 1578-1588, Sept. 2012.

[24] N. Wang, N. Zhang, and T. Gulliver, Cooperative Key Agreement for Wireless Networking: Key Rates and Practical Protocol Design, IEEE Transactions on Information Forensics and Security, vol. 9, pp. 272-284, Feb. 2014.

[25] Q. Wang, K. Xu, and K. Ren, Cooperative Secret Key Generation from Phase Estimation in Narrowband Fading Channels, IEEE Journal on Selected Areas in Communications, vol. 30, pp. 1666-1674, Oct. 2012.

[26] Y. Wei, C. Zhu, and J. Ni, Group Secret Key Generation Algorithm from Wireless Signal Strength, in Sixth International Conference on Internet Computing for Science and Engineering (ICICSE), pp. 239-245, Apr. 2012.

[27] C. Ye and A. Reznik, Group Secret Key Generation Algorithms, in IEEE International Symposium on Information Theory (ISIT 2007), pp. 2596-2600, June 2007.

[28] Al-Riyami, S. S., & Paterson, K. G. (2003). Certificateless public key cryptography. In *Advances in Cryptology-ASIACRYPT 2003* (pp. 452-473). Springer Berlin Heidelberg.

[29] Luo, M., Tu, M., & Xu, J. (2014). A security communication model based on certificateless online/offline signcryption for Internet of Things. *Security and Communication Networks*, *7*(10), 1560-1569.

[30] Shi, W., Kumar, N., Gong, P., Chilamkurti, N., & Chang, H. (2014). On the security of a certificateless online/offline signcryption for Internet of Things. *Peer-to-Peer Networking and Applications*, 1-5.

[31] Marin, L., Jara, A. J., & Skarmeta, A. F. (2011). *Shifting primes: extension of pseudo-mersenne primes to optimize ECC for MSP430-based future internet of things devices* (pp. 205-219). Springer Berlin Heidelberg.

[32] Boneh, D., & Franklin, M. (2001, January). Identity-based encryption from the Weil pairing. In *Advances in Cryptology—CRYPTO 2001* (pp. 213-229). Springer Berlin Heidelberg.

[33] Shamir, A. (1985, January). Identity-based cryptosystems and signature schemes. In *Advances in cryptology* (pp. 47-53). Springer Berlin Heidelberg.

[34] Sahai, A., & Waters, B. (2005). Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005* (pp. 457-473). Springer Berlin Heidelberg.

[35] Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006, October). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security* (pp. 89-98).

[36] Bethencourt, J., Sahai, A., & Waters, B. (2007, May). Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP'07. IEEE Symposium on* (pp. 321-334). IEEE.

[37] Guo, R., Wen, Q., Shi, H., Jin, Z., & Zhang, H. (2014). Certificateless public key encryption scheme with hybrid problems and its application to internet of things. *Mathematical Problems in Engineering*, *2014*.

[38] Adiga, B. S., Balamuralidhar, P., Rajan, M. A., Shastry, R., & Shivraj, V. L. (2012, August). An identity based encryption using elliptic curve cryptography for secure M2M communication. In *Proceedings of the First International Conference on Security of Internet of Things* (pp. 68-74). ACM.

[39] Ahmad, A., Biri, A., Afifi, H., & Zeghlache, D. (2009, September). TIBC: Trade-off between Identity-Based and Certificateless Cryptography for future internet. In *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on* (pp. 2866-2870). IEEE.

[40] Touati, L., Challal, Y., & Bouabdallah, A. (2014, June). C-CP-ABE: Cooperative Ciphertext Policy Attribute-Based Encryption for the Internet of Things. In *Advanced Networking Distributed Systems and Applications (INDS), 2014 International Conference on* (pp. 64-69). IEEE.

[41] Santesson, s., Nystrom, M., & Polk, T. (2004). *Internet X.509 Public Key Infrastructure: Qualified Certificates Profile*. (RFC 3739). *IETF Request For Comments*

[42] Dierks, T., Rescorla, E. (2008). The transport layer security (TLS) protocol version 1.2. (RFC 5246). *IETF Request For Comments*

[43] Rescorla, E., & Modadugu, N. (2006). Datagram transport layer security. (RFC 4347). *IETF Request For Comments*

[44] Shelby, Z., Hartke, K., & Bormann, C. (2014). The Constrained Application Protocol (CoAP). (RFC 7252). *IETF Request For Comments*

[45] Camenisch, J., & Van Herreweghen, E. (2002, November). Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM conference on Computer and communications security* (pp. 21-30). ACM.

[46] Camenisch, J.; Lysyanskaya, A. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Advances in Cryptology - EUROCRYPT 2001; Springer, 2001; pp. 93–118.

[47] Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008, January). MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on* (pp. 791-798). IEEE.

[48] Conti, M., Giordano, S., May, M., & Passarella, A. (2010). From opportunistic networks to opportunistic computing. *Communications Magazine, IEEE*, *48*(9), 126-139.

[49] Guo, B., Zhang, D., Wang, Z., Yu, Z., & Zhou, X. (2013). Opportunistic IoT: exploring the harmonious interaction between human and the internet of things. *Journal of Network and Computer Applications*, *36*(6), 1531-1539.

[50] Maji, H. K., Prabhakaran, M., & Rosulek, M. (2011). Attribute-based signatures. In *Topics in Cryptology–CT-RSA 2011* (pp. 376-392). Springer Berlin Heidelberg.

[51] Su, J., Cao, D., Zhao, B., Wang, X., & You, I. (2014). ePASS: An expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the Internet of Things. *Future Generation Computer Systems*, *33*, 11-18.

[52] Hernández-Ramos, J. L., Jara, A. J., Marín, L., & Gómez, A. F. S. (2014). Dcapbac: Embedding authorization logic into smart things through ecc optimizations. *International Journal of Computer Mathematics*, (ahead-of-print), 1-22.

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

[54] Karp, A. H., Haury, H., & Davis, M. H. (2010). From ABAC to ZBAC: the evolution of access control models. In *Proceedings of the 5th International Conference on Information Warfare and Security, ed. EL Armistead* (pp. 202-211).

[55] Crockford, D. (2006). The application/json media type for javascript object notation (json). (RFC 4627). *IETF Request For Comments*

[56] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999, January). Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing* (pp. 304-307). Springer Berlin Heidelberg.

[57] M. J. Covington, P. Fogla, Z. Zhan, and M. Ahamad, “A context-aware security architecture for emerging applications,” in Computer Security Applications Conference, 2002. Proceedings. 18th Annual. IEEE, 2002, pp. 249–258.

[58] Langheinrich, M. Privacy by design—principles of privacy-aware ubiquitous systems. Ubicomp 2001: Ubiquitous Computing. Springer, 2001, pp. 273–291.

[59] Pfitzmann, A.; Hansen, M. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, 2010.

[60] Bassi, A.; Bauer, M.; Fiedler, M.; Kramp, T.; van Kranenburg, R.; Lange, S.; Meissner, S. Enabling Things to Talk, 2013.

[61] Vučinić, M., Tourancheau, B., Rousseau, F., Duda, A., Damon, L., & Guizzetti, R. (2014). OSCAR: Object security architecture for the Internet of Things. *Ad Hoc Networks*.

[62] Yang, Z. (2012). PowerTutor-A Power Monitor for Android-Based Mobile Platforms. *EECS, University of Michigan, retrieved September*, *2*.

[63] Java CP-ABE library.https://github.com/junwei-wang/cpabe

[64] 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.

[65] Brown, M.; Hankerson, D.; López, J.; Menezes, A. Software implementation of the NIST elliptic curves over prime fields; Springer, 2001.

[66] Guideline, N.E.A. NIST Special Publication 800-63 Version 1.0. 2, 2006.

[67] SocIoTal Deliverable 2.1. "D2.1: IoT Communities and Identity Management"

[68] SocIoTal Deliverable 2.2. "D2.1: Framework Specification for Privacy and Access Control"

[69] SocIoTal Deliverable 3.1.1. "D3.1.1: Device centric enablers for privacy and trust"

[70] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.
http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML