TagItSmart!
Smart Tags driven service platform for enabling ecosystems of connected objects

Grant agreement 688061

TagItSecure!
Final TagItSecure! Prototype

Deliverable ID:	**D 2.1**
Deliverable Title:	Final TagItSecure! Prototype
Deliverable Type	DEM
Dissemination Level:	**Public/Confidential**
Responsible beneficiary:	FINCONS GROUP AG
Contributing beneficiaries:	FINCONS GROUP AG
Contractual date of delivery:	31/05/2018
Actual submission date:	02/07/2018
Start Date of the Sub-Project	01/02/2018
Duration:	5 Months

PROPRIETARY RIGHTS STATEMENT
This document contains information, which is proprietary to the TagItSmart! Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or in parts, except with prior written consent of the TagItSmart! consortium.

Scope of the document

This document has been produced to provide a short description of the software delivered as output of the WP 2, *Integration and Testing*, wich led to the finalization of the demonstrator delivered as *D2.1 Final TagItSecure! Prototype* and implemented according to the design illustrated in *D1.1 TagItSecure! Reference Architecture.* The code produced for the demonstrator integration – essentially the MQTT-SeDEM library, along with other components (namely *ABE-Proxy, Device Entity* and *CPABE*) that have been adjusted or extended for the demonstrator integration – is publicly available on the GitHub repository at https://github.com/FINCONS-IBD/MQTT-SeDEM , along with the related Javadoc as well as configuration and installation instructions, available in the README file of the repository and reported below for convenience.

Software Release Notes

MQTT-SeDEM

MQTT based Secure Data Exchange Midlleware (SeDEM). A Java Library to perform secure publish/subscribe operations against an MQTT-complaint Broker.

This library provides Java classes to perform publish and subscribe MQTT operations by means of the Event class it defines, which provides an Event creation method and implements an event callback listener. It makes easy all the operations against the MQTT Broker Server.

Building and installation procedure

The library is developed as Maven Project then the building procedure simply implies to use the .pom file as usual in a Maven project. The result of the maven build operation is a mavenized JAR that includes the dependencies; the application that use the library, then, must import also this jar.

Initial Configuration

To permit a dynamic configuration of Logger and Application, the configuration files are external to the application jar. Then it is important using the following VM Arguments while running the main application (either in a .bat / .sh run file or as VM Arguments specified in the IDE).

```
-Drabbitmq_config_file=C:\resources\conf.properties -
Dlog4j.configuration=file:C:\resources\log4j.properties
```

NOTE: if the library has to be installed on a Linux O.S. it is important to use a Linux well formed path

- **Application configuration**

The first step before using the library is the editing of the configuration parameters detailed below. The file named conf.properties is located in the external resource folder of the library (see the "Initial Configuration" introduction section). Follow an example of configuration file:

```
MQTT_USERNAME=username
MQTT_PASSWORD=password
MQTT_CLIENT_ID=client_id_####
MQTT_QOS=1
MQTT_BROKER=broker.url.com
MQTT_PORT=8883
MQTT_PROTOCOL=ssl
MQTT_CLEARSESSION=true
MQTT_RETAINED=true
MQTT_TIME_TO_WAIT=100000
MQTT_TIMEOUT=30
MQTT_KEEP_ALIVE=125
CPABE_AES_ENCRYPTION=true
CPABE_POLICY=c:IT
CPABE_ANTICIPATED_KEY=true
CPABE_ANTICIPATED_KEY_IN_SECONDS=20
KEY_TYPE=EC
CRYPTOGRAPHIC_CURVE=P-256
ALG=ECDH-ES
ENC=A128GCM
PROXY_PROTOCOL=http
PROXY_IP=(abe-proxy-ip)
PROXY_PORT=(abe-proxy-port)
PROXY_ID=ABE-Proxy
STORAGE_TYPE=database
DB_IP=(key-storage-ip)
DB_PORT=(key-storage-port)
DB_AUTH_USER=(key-storage-username)
DB_AUTH_PWD=(key-storage-password)
DB_DATABASE=(key-storage-database)
DB_TABLE=(key-storage-table)
PAYLOAD_PATTERN=[{"value": "{"headers":{},"payload":""}"}]
PLACEHOLDER_PATTERN="{"headers":{},"payload":""}"
CONSUMER.PUBLIC_PARAM=C:/resources/Key/pub_10
CONSUMER.CPABE_PRIV_KEY=C:/Key/personal_key
```

Details about parameters:

- *MQTT_USERNAME* - username authorised to connect to MQTT Broker Server endpoint
- *MQTT_PASSWORD* - the password to authenticate against the MQTT Broker Server
- *MQTT_CLIENT_ID* - the unique client string identifier
- *MQTT_QOS* - Quality of Service level: 0 – at most once or 1 – at least once
- *MQTT_BROKER* - the endpoint address of the MQTT Broker Server
- *MQTT_PORT* - the port number where the MQTT Borker Service is listening on
- *MQTT_PROTOCOL* - the protocol used to perform the MQTT Broker connection
- *MQTT_CLEARSESSION* - the flag to specify if the persistence of the messages and subscriptions is required
- *MQTT_RETAINED* - the flag to specify the persistence of the message. All the messages flagged as retained will be stored by the broker
- *MQTT_TIME_TO_WAIT* - measured in milliseconds, it is the maximum time to wait for an action to complete
- *MQTT_TIMEOUT* - measured in seconds, it defines the maximum time interval the client will wait for the network connection to the MQTT server to be established
- *MQTT_KEEP_ALIVE* - measured in seconds, defines the ping interval to keep the connection alive. It is important to know the max keep alive of the server (e.g. 120s on Mosquitto)
- *CPABE_AES_ENCRYPTION* - flag to enable secure exchange of events
- *CPABE_POLICY* - the personal cpabe policy
- *CPABE_ANTICIPATED_KEY* - flag to enable the calculation of the anticipated key (the next key to be used)
- *CPABE_ANTICIPATED_KEY_IN_SECONDS* - the number of seconds before the key expiration date to start the anticipated key calculation
- *KEY_TYPE, CRYPTOGRAPHIC_CURVE, ALG, ENC* - details about the key type
- *PROXY_PROTOCOL, PROXY_IP, PROXY_PORT, PROXY_ID* - Abe Proxy endpoint details

- *STORAGE_TYPE, DB_IP, DB_PORT, DB_AUTH_USER, DB_AUTH_PWD, DB_DATABASE, DB_TABLE* - Key Storage endpoint details
- *PAYLOAD_PATTERN* and *PLACEHOLDER_PATTERN* - the message pattern details
- *CONSUMER.PUBLIC_PARAM* - the location of cpabe public key
- *CONSUMER.CPABE_PRIV_KEY* - the location of cpabe private key

- **Logging configuration**

In the external resource folder of the library (see the "Initial Configuration" introduction section) is present a logging configuration file named *log4j.properties*. It is possible to edit this configuration file to change the file size and location and dimension, as well as logging level of the logger. Here follows an example of the config logging file:

```
# Root logger option
log4j.rootLogger=INFO, stdout, file
# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
%c{1}:%L - %m%n
# Redirect log messages to a log file, support file rolling.
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=C\:\\rabbitmq_client.log
#log4j.appender.file.File=/opt/logs/rabbitmq_client.log
log4j.appender.file.MaxFileSize=5MB
log4j.appender.file.MaxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L
- %m%n
```

NOTE: if the library is installed on a Linux O.S. it is important to change the log file path (see the commented line)

• Software dependencies

The library has 2 internal dependencies:

- log4j, a Java Logging Library
- Eclipse Paho library to interact with the MQTT Broker Server

The other folders present in the repository (ABE-Proxy, CPABE and Device-Entity) represent the other components/libraries involved in the MQTT-SeDEM architecture then it is important to setup the entire architecture to permit a right interaction between the MQTT-SeDEM and the other components.

- ## An example of use

In the *com.fincons.mqtt.client.test* package is present a simple example of Publish and Subscribe Scenario including a Simple MQTT Callback class that is invoked when a new MQTT message arrives. Please refer to the classes included in the package above for more details.

NOTE: For more details about the library usage please refers to JavaDoc folder.

This work is part of **TagItSmart!** H2020 European Project, and in particular is developed in the **TagItSecure!** project funded in the frame of the 2nd TagItSmart! Open Call.

To demonstrate the main functionalities of the MQTT SeDEM, a real use-case was developed within the TagItSecure project; the demonstrator consists in the following main components:

- *a data source component* that collects sensor measurements and publishes these event measurements in a secure way using the MQTT-SeDEM encrypting/publishing features. For this reason a MQTT-compliant Broker is required; for this purpose the Evrthyng IoT Platform was used (see here for more details). In the MQTT-SeDEM project (see here), and in particular in the *com.fincons.mqtt.client.test* package is present a simple example of data source Java Publisher.
- *the ABE-Proxy component* (see here) to perform the heavy CPABE encryption operations. In particular to optimize the encryption operations, the encryption of data is based on AES symmetric keys; the ABE proxy executes the CP-ABE encryption algorithm to protect and save these encrypted symmetric keys. For this purpose, the ABE-Proxy interacts with a *Key Storage Service*. In the project the NOSQL OrientDB database has been identified to fulfill these needs.
- *a data consumer* that performs subscribe operations on the MQTT topic through the MQTT-SeDEM library and decrypts the AES encrypted sensor measurements retrieving the CPABE Encrypted AES key from the *Key Storage Service* instance shared with the ABE-Proxy component described above. In the MQTT-SeDEM project (see here), and in particular in

the *com.fincons.mqtt.client.test* package is present a simple example of data consumer (a simple Java Subscriber).

For more details about the building procedures required to set-up the single components please refer to the single projects descriptions.

ABE-Proxy

The ABE Proxy is the new architectural component responsible to perform the heavy ABE encryption operations and the integration of CP-ABE and AES encryption techniques, so to speed up the encryption/decryption processing and reduce resource needs on the client side.

It is a web based application that uses novel attribute-based encryption techniques in order to protect access to information, which is encrypted according to user's provided access policies and the generation of decryption keys tied to the user's profiles.

Installation guide

In order to install the application, it is necessary to deploy the war obtained after the build process in the Tomcat environment, and to copy the Public Key generated by CPABE Service in the folder ABE-Proxy\WebContent\WEB-INF\cpabe_keys\pub_10 specified.

Another important configurable parameter is the expiration time that indicates the life-time of the associated symmetric key. When this life-time expires, it is strongly recommended to renegotiate a new symmetric key between data source and ABE proxy in order to guarantee that the communication continues reliable and secure, and to achieve the well-known Perfect Forward Secrecy. This parameter is set in the ABE Proxy's *web.xml* file, and is in millisecond format. Following an example:

```
<init-param>
        <param-name>expiration_time</param-name>
        <param-value>5000</param-value>
</init-param>
```

The ABE Proxy provides two public endpoint through two different sockets:

- *ABE-Proxy/generate_shared_secret*: performs the agreement of a symmetric key between the data source and the ABE-Proxy using *Elliptic Curve Diffie-Hellman Ephemeral Static*, combined with the *Concat Key-Derivation Function*.
- *ABE-Proxy/cpabe_information*: the data source delegates the CP-ABE encryption operations to the ABE-Proxy.

Software dependencies

The ABE-Proxy component has two important dependencies:

- *CPABE library* (see here) that provides the CP-ABE encryption and decryption algorithms and entities.
- *Key Storage Service* component: in which the ABE-Proxy saves the CP-ABE encrypted AES symmetric keys. It is a fully configurable component (specified in the *web.xml* configuration file); in the project the NOSQL OrientDB database has been identified to fulfill these needs.

For more details about the building procedures required to set-up the single components please refer to the single projects descriptions.

Device-Entity

This library provides support to any client device that needs to implement CPABE encryption and decryption and needs to interact with the ABE-Proxy component.

The library provides the high level algorithms and entities to perform AES or CPABE encryption and decryption operations and the methods to interact with the ABE-Proxy through the http/https protocol.

Installation guide

The library is provided as mavenized project, then it is easy to build the library jar with dependencies using the *pom.xml* file as usual in a maven project.

Software dependencies

To use this software, you will need to have the *CPABE library* (see here) referenced in the *pom.xml* file.

CPABE

This software is a Java realization for "ciphertext-policy attribute based encryption" (CP-ABE).

It is based on the https://github.com/Bysmyyr/cpabe implementation.

You are responsible for the problem caused by using the code.

Installation guide

To use this software, you will need to have the Java Pairing Based Cryptography Library (jPBC) library installed. You can get it from the following page:

http://gas.dia.unisa.it/projects/jpbc/

The library is provided as mavenized project, then it is easy to build the library jar with dependencies using the *pom.xml* file as usual in a maven project.

Important note

This library is used by the Device-Entity (see here) and ABE-Proxy (see here) components. To permit a right interaction between these two components it is important to use the same version of the CPABE Library.