Security and privacy mechanisms specification

AUTHORS : KRZYSZTOF PIOTROWSKI
IGOR KOROPIECKI

DATE : 31.01.2021

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grand agreement N°864283

Technical References

Project Acronym	ebalance-plus
Project Title	Energy balancing and resilience solutions to unlock the flexibility and increase market options for distribution grid
Project Coordinator	CEMOSA
Project Duration	42 months

Deliverable No.	D5.2
Dissemination level [1]	PU
Work Package	WP5 Communication Platform and system integration
Task	T5.2 Security and privacy mechanisms
Lead beneficiary	IHP
Contributing beneficiary(ies)	IHP
Due date of deliverable	31 January 2021
Actual submission date	01 February 2021

[1] PU = Public
PP = Restricted to other programme participants (including the Commission Services)
RE = Restricted to a group specified by the consortium (including the Commission Services)
CO = Confidential, only for members of the consortium (including the Commission Services)

Document history

V	Date	Beneficiary	Author
0	11/01/2021	IHP	Krzysztof Piotrowski
1	27/01/2021	IHP	Igor Koropiecki, Krzysztof Piotrowski

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

Summary

1.1 Summary of Deliverable

This document defines the security context of the energy management platform by analysing the potential attacks and proposing solutions to protect against these. Additionally, the document describes the privacy protecting mechanism to be implemented in the middleware. These mechanisms allow the data owners to define the list of other stakeholders to be allowed to access the owner's data. Together with security protecting mechanisms this supports secure, reliable and fair data handling within the energy management platform.

Disclaimer

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

Table of Contents

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

Table of tables

Table of figures

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

List of Abbreviations

Abbreviations	Definitions
CA	Certification Authority
CMU	Customer Management Unit
DER	Distributed Energy Resources
DERMU	Distributed Energy Resources Management Unit
DMU	Device Management Unit
DSO	Distribution System Operator
ESCO	Energy Service Company
LVGMU	Low Voltage Grid Management Unit
MU	Management Unit
MVGMU	Medium Voltage Grid Management Unit
PS	Primary Substation
SG	Smart Grid
SS	Secondary Substation
TLGMU	Top Level Grid Management Unit
TSO	Transmission System Operator

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

1 Introduction

The goal of the ebalance-plus project is to provide a flexible and powerful energy management platform to enable the flexibility in the energy sector (the main focus in this project is the electricity grid, but sector coupling is supported by the platform as well). This task requires to capture, exchange and process large amount of data. Depending on the algorithms to be implemented the data may allow to profile the energy users with respect to different aspects of their lives. It is also crucial for the stability of the energy grid that no one is able to interfere with the system and can, for instance, obtain data she is not allowed to or insert falsified data to influence the system functioning. This document does an investigation on the possible threats and proposes the countermeasures.

The following section introduces the ebalance-plus architecture, introduces the computational units that constitute the system and the stakeholders that participate in it. Then, the possible attacks are listed together with possible ways to protect against these attacks. Section 4 explains the way the identification of the devices and users is performed. Then Section 5, Section 6 and Section 7 explain the protection measures applied on the different parts of the system and on the interfaces between them. Section 8 presents the privacy protection mechanism. The document is concluded by Section 9.

2 The ebalance-plus architecture

This section presents the generic view on the ebalance-plus architecture. The main focus here is to name all the devices and stakeholders that are involved in the system and are important from the security and privacy perspective, as well as the relations between them.
In the ebalance-plus architecture, similar as it was defined in the previous e-balance project [1], we have a distributed set of computers, called management units (MU) that are connected with each other and with sensors and actuators installed within the energy grid allowing the system to interact with the grid – monitoring and controlling it. A connection between two MUs is defined on the parent-child basis, meaning that a parent MU is also managing all its children. This allows creating hierarchical management structures with specific levels and level naming. For instance, the lowest level MUs are installed within end user households and such a customer management unit (CMU) manages all the appliances of this particular household. Going higher in the hierarchy we can define the next level at the secondary substation level and here a low voltage grid management unit (LVGMU) manages all the CMUs in its area (substation coverage) and all the sensors and actuators installed in this branch of the grid. An even higher level can be the primary substation level, where the medium voltage grid management units (MVGMU) manage all the LVGMUs in their respective area. If we define then one more level to manage all the MVGMUs and define a top-level grid management unit (TLGMU) that does the task, we have a hierarchical energy management platform that covers the distribution grid. It is of course possible to define additional levels, either in between these just mentioned or higher levels covering, for instance, the transmission grid. Similar, we can also define management units below the CMU level by defining a device management unit (DMU) to manage the components of a single device or appliance, and a distributed energy resources management unit (DERMU) to manage a stand-alone DER device located on different levels of the energy grid. In general, the management units always process their local data and generate local knowledge and decisions, thus, to keep the data paths short (and whenever possible), they should be located close to the area they manage within the energy grid.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

The example hierarchical architecture composed on these mentioned management units and other components they interact with is presented in Figure 1 together with the interconnections between these and the networking details as stated in deliverable D5.1.

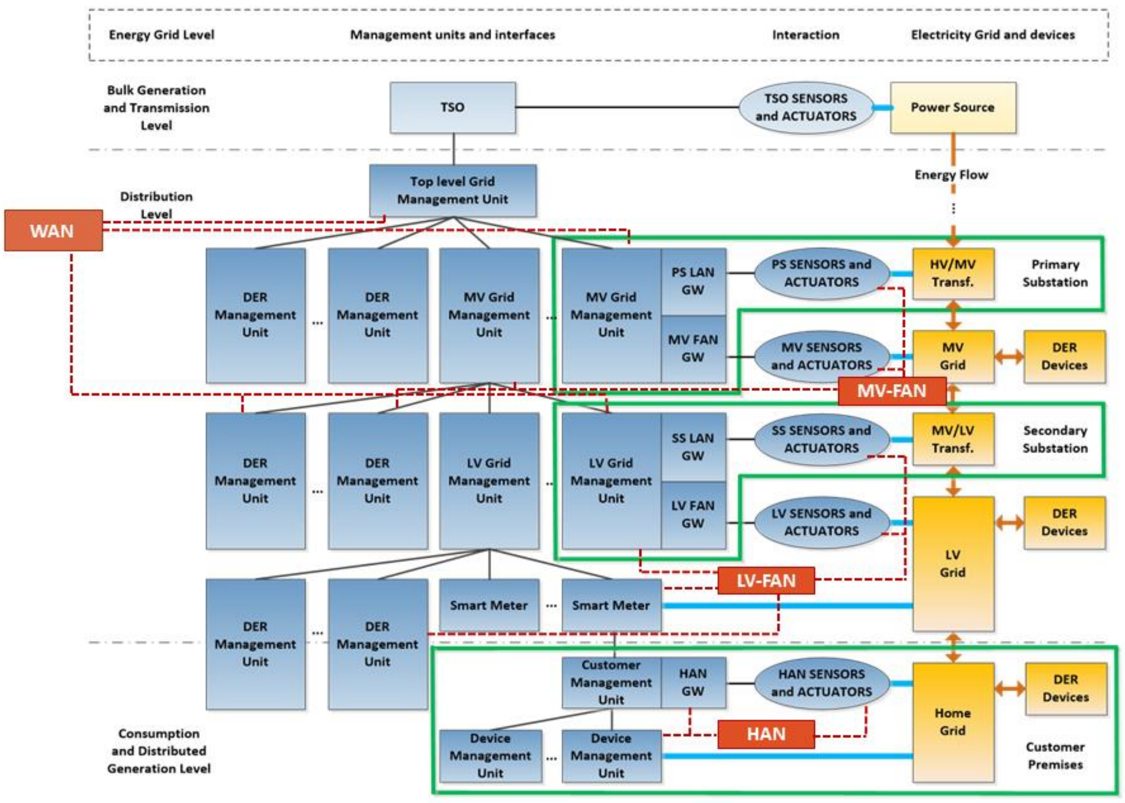

Figure 1 An example of ebalance-plus hierarchical architecture

The ebalance-plus energy management software runs on the management units. This software consists of two layers. The lower one is the distributed middleware that is installed on each management unit and provides a common platform for the data exchange between the management units. The middleware instances act as if they were a single, but distributed database containing all the data. And the data is then further processed by the second software layer – the services that also run on the management units and can access the data in the middleware using the Data Interface (see Figure 2). The services are installed on the management units depending on the system requirements. They implement the management algorithms and are placed within the hierarchical structure, according to their goals. This means that not all the services run on all the MUs. Further, the services are like programs that are executed by specific users – the stakeholders. Thus, each running instance of a given service runs on behalf of a specific stakeholder, realizing the tasks for this particular stakeholder or her clients.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

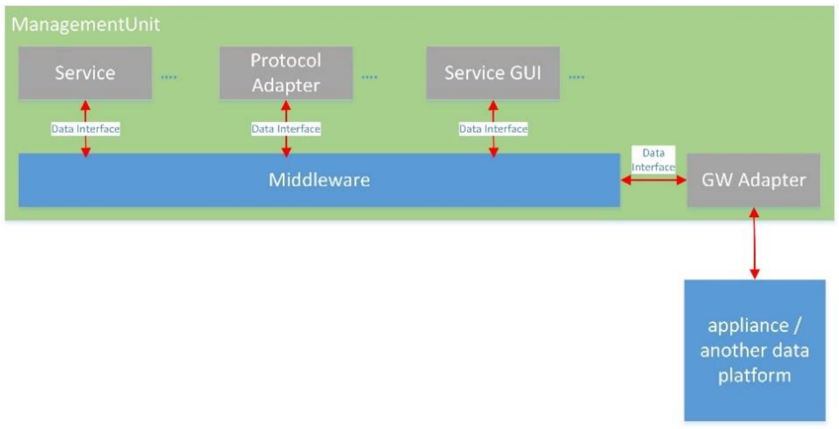

Figure 2 The architecture of a management unit

For the later discussion we will distinguish between the technical users (operators that install and configure the ebalance-plus infrastructure) and the energy grid stakeholders. The technical users can install services, setup middleware instances and configure them, while the stakeholders rely on solutions that were configured for them and operate on the data level.

The set of stakeholders is not limited in any sense. It includes the typical types of energy grid users, like the end customers, the ESCOs, the DSOs etc., but it also includes ones, like aggregators, IT service providers and similar. The ebalance-plus system does not categorize the stakeholders in any sense, at least not at the level of data exchange. But we need to define some groups of stakeholders for the further analysis of their interactions.

So, we can define the simple example of the stakeholder classes as given in the Table 1. This list gives us an overview on the relations between the stakeholders and their potential interests that can be expressed by the services they run/use. Further, the ownership of the premises or assets defines the possible relations between the stakeholders and the locations, where management units are installed, taking the example hierarchy levels mentioned previously. This also defines the physical access to the devices and due to that sometimes the stakeholder that holds the management unit can be considered as the one running the middleware on this MU. Which is actually reflected as the fact that this stakeholder decides about the set of services that can be executed on the given MU.

Table 1 Example classes of stakeholders

Stakeholder class	Comment
DSO	Owns the distribution infrastructure. Is interested in its easy maintenance and in keeping the grid parameters below threshold levels, e.g., by energy production and consumption control at the customers. The LVGMUs, MVGMUs and the TLGMU are installed within the premises of the DSO. These become by that the part of the DSO infrastructure. The DSO runs services related to the monitoring of the grid and its maintenance. But also energy accounting. The DSO operates in the scope of the whole distribution network it owns.
ESCO	Offers energy related services. May be interested in detailed knowledge on the past, current and future behaviour of the

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

	customers, in order to be able to better optimize the processes. Runs services related to energy optimizations. The ESCO operates in the different DSO grids, where its clients are.
Energy supplier	Buys energy from energy producers and sells it to the customers. Runs services related to energy supply optimization. The energy supplier operates in the different DSO grids, where its clients are.
Aggregator	Acts as a large customer by representing a group of (smaller) customers. Runs services related to the energy related aspects where it represents its clients. The energy supplier operates in the different DSO grids, where its clients are.
Customer	Consumes (and also sometimes produces) energy delivered by the DSO infrastructure and sold by the energy supplier. The CMUs and DERMUs are located at the customer premises. A customer runs services related to her energy behaviour and monitoring and controlling of her appliances. The customer operates in her premises.

Thus, to summarize, we can say that in the ebalance-plus energy management system we have distributed management units, each running an instance of the middleware and different sets of services, with each service run by a given stakeholder. This system allows implementing distributed algorithms for energy management. This distribution of the devices and algorithms allows processing the data locally, generate local control signals and is by that very scalable. However, if not protected properly, such a system opens new ways to threats.

Before we start explaining the security measures to protect our approach, we need to introduce some additional elements in the system architecture that need to be highlighted from the security perspective. The top level of the hierarchy is indeed structured as depicted in Figure 3. The SuperUnit is the MU on the highest level of a given deployment. It may be the TLGMU, but in a setup with many DSO grids connected, it will be an even higher level MU that is above the TLGMUs. Another additional element is the discovery server that is responsible for maintaining the list of registered stakeholders and MUs together their respective certificates, which it also issues. By that it is has the role of being the certification authority (CA) in the ebalance-plus system. The last element is the proxy server that is responsible for buffering requests targeted at management units located behind network elements, like routers or firewalls that prevent these MUs to be directly addressed from outside. Such MUs need then to collect their requests from the proxy server.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

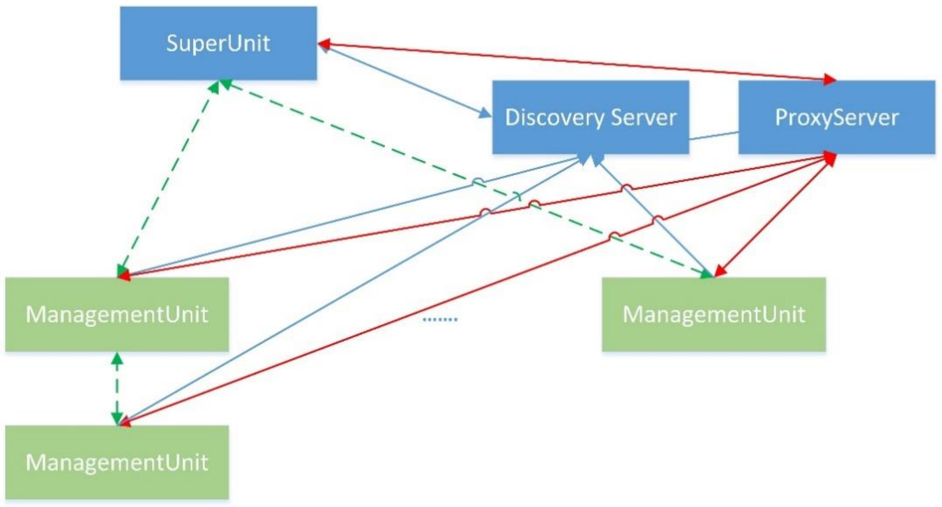

Figure 3 The structure of the top level in the ebalance-plus system architecture

3 Security threats

This section presents the analysis of the potential attacks that can be executed against the ebalance-plus platform by adversaries in order to achieve different goals. The attacks will also be categorized by the ease of execution and by the gain they can bring, if successful.

First, we can divide the attackers in two groups, i.e., internal and external. The internal ones have already some knowledge about the system given, while the external ones need to gather the information they need from scratch. An internal attacker can perform an attack also unintentionally, e.g., by trying to perform an action that is not allowed, i.e., that she is not authorised to do. This is then indeed not an attack, but misbehaviour and the system should simply reject such an action. Anyway, in order to define what is allowed and what not, an access control policy is needed.

Further, we can identify the part of the system that is under attack. Here it is crucial to mention if the attacker has physical access to the target, what actions she performs and what is the intended goal. The attack can be performed on the physical devices, but also on the communication medium that is used to transmit packets between devices. Depending on the effort needed to perform an attack approach, the attack can be categorized as easy, medium or hard. The effort is a mixture of the necessary investments and knowledge. In order to create a metric here, we can define the monetary equivalent needed to perform an attack. What is important here, we ignore the social aspect (exploiting the human factors) and luck in obtaining the needed knowledge, as these factors are hard to measure. Thus, if the attack can be performed using simple tools available in the Internet (at no costs or for less than 500€) and it does not require expert knowledge, it is considered easy. On the other end if the costs go beyond 500.000€ the attack is considered hard. Anything in between is considered medium.

But, even if it is easy to perform the attack, it does not mean that it will be successful. And depending on the effects a successful attack can cause, its severity can be negligible, medium or critical. Here we can also define a monetary metric to measure the effect. And the thresholds could be defined in a similar way, i.e., below 500€ could be negligible, above 500.000€ critical, and medium in between. But indeed, no successful attack should be considered negligible – the fact that it succeeded is a medium danger already.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

Table 2 sketches the security threats we considered for the system. Of course this list can be extended by additional attacks, but we tried to consider the most relevant ones specific for the ebalance-plus system.

Table 2 Security threats

No	Attacker	Target and goal	Ease	Severity
1	external	Physical or remote security exploit in order to get full access to the MU operating system.	medium to hard	medium to critical
2	internal	Physical or remote security exploit in order to get full access to the MU operating system.	easy to hard	medium to critical
3	external	Physical or wireless attempt to eavesdrop or influence the communication between management units.	easy to hard	medium to critical
4	internal	Physical or wireless attempt to eavesdrop or influence the communication between management units.	easy to hard	medium to critical
5	internal	Preparing a service that tries to access data the attacker should not have access to.	easy to medium	medium to critical
6	external	Preparing a management unit that acts as part of the system,	easy to medium	medium to critical
7	internal	To impersonate a stakeholder, service or device.	easy to medium	medium to critical
8	external	To impersonate a stakeholder, service or device.	medium to hard	medium to critical

As mentioned before, all the attacks have a severity level at medium to critical. There are none labelled as negligible. That is because once an attack becomes successful, it also becomes a legal data protection breach (medium), and the actual effects depend on the accessed data, kind of data access, the attack scale and duration, but it can even cause grid instability or destruction (critical).

4 Secure identification by certificates

The hardware, software components and stakeholders introduced in the previous section need a secure way to identify themselves. The platform uses the concept of public key infrastructure [2] (PKI) to introduce security and privacy in communication between participating stakeholders. The public key infrastructure ensures the confidentiality, integrity and authenticity of the messages exchanged on the middleware platform.

The PKI is based on public key encryption [3] which is a cryptographic system based on mathematical problems in which each participant has two keys – public and private. The public key can be distributed publicly and freely to everyone willing to communicate with the

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

participant, while the private key has to be kept as secret as possible by the client, because it can be used to decrypt a message that was previously encrypted with a corresponding public key of the same participant.

Here we deliberately do not specify the security levels in terms of key length or the technology (cryptography) used, as these are parameters that shall be specified for a deployment on the basis of current requirements (standards and values considered secure). At the moment of writing this document, PKI certificates are based on RSA or ECC cryptography with key lengths of at least 2048 bits (RSA) or 256 bits (ECC). In the case of symmetric encryption, AES with key length of at least 128 bits, is considered secure. The PKI introduces a concept of certificate that most importantly contains the owner information and the public key used in the encryption process. Each certificate has a corresponding private key that is kept separate from the certificate and can be used to decrypt any message that was encrypted with the corresponding public key. Each actor participating in the communication is required to have a valid certificate that was issued (generated and signed) by a trusted entity. The actors include middleware servers, services or middleware proxy servers. The certificate can be used to prove:

a) **Identity of the actor participating in the communication,**
b) **ownership of the public key.**

In order to be able to issue certificates for the entities, a trusted entity has to be established. The technical name of such entity in the PKI is Certification Authority (CA). The certification authority is responsible for:

a) **verifying the identity of participating actors,**
b) **issuing and storing the certificates for verified actors,**
c) **maintaining a list of revoked certificates,**
d) **providing a mechanism for checking revoked certificates.**

The certification authority has its own certificate with a separate corresponding private key that is kept secret. Each actor has a local copy of the certification authority certificate (authority identification + authority public key).

When an actor is willing to communicate on the middleware platform, she must create a certificate signing request (CSR) which (among others) contains the actor's certificate (identity and public key). The request is forwarded to the certification authority. The authority performs a verification process to confirm the identity of the requesting actor. When the verification process is successful, the certification authority uses its private key to sign the certificate sent in the CSR. A signed certificate is returned to the requesting actor. Disclosing private keys of both parties (even to each other) at any step of the certification process would break the security enforced by the public key cryptography. Therefore it has to be clearly stated that the private keys of both the requesting actor and the certificate authority are not shared at any step of the certification process and:

- **the certification authority uses its private key only to sign certificates,**
- **the actor uses its private key only to decrypt messages that were encrypted with the corresponding public key.**

Without knowing the certification authority's private key, it is computationally extremely expensive (for most of actors/scenarios considered impossible) to create a valid signed certificate (in acceptable period of time) with altered owner information. The actors can use this fact to validate the authenticity of other actors. When attempting to communicate, both actors have to provide their signed certificates, of which the signature can be verified. Any modification attempt of data signed in the certificate will result in signature mismatch which will alert the other actor and the connection can be rejected. After the signature verification, both actors can communicate with the certification authority to check if the certificate is not revoked.

When the private key of an actor's signed certificate is compromised or the actor is behaving maliciously, the certificate can be revoked which equals to blacklisting the certificate by the

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

certification authority. A revoked certificate is no longer considered trusted. Based on the circumstances, the certificate can be reissued or cooperation with the malicious actor can be terminated. The certification authority has no power to physically take away the certificate from the actor, therefore other actors have to check whether their communication participant is identifying itself with a revoked certificate.

The implementation of the security features is modular and configurable. It allows to quickly update the underlying mechanisms or chosen cipher suite in case that the currently chosen option is no longer considered secure.

In the ebalance-plus system the following certificate classes are issued by the certification authority:

- **middleware instance / management unit certificate,**
- **stakeholder / user certificate,**
- **certificate to prove the identity of a service running on behalf of a given stakeholder**

We do not certify the services individually, because the stakeholder and the service identities together define the compound identity that defines what the given instance of a service can access (do) and how.

5 Data Interface security

The Data Interface security scenario distinguishes three classes of actors – middleware server, service and the certification authority. Generally, a single local middleware server and its local services run on behalf of their stakeholders, these may be different. The services perform various operations (always) on the local middleware server, such as data collection, signal processing or management. A service and the middleware server communicate through a channel called the Data Interface. The certification authority is used to authorize services and middleware servers. The relations have been shown on Figure 4.

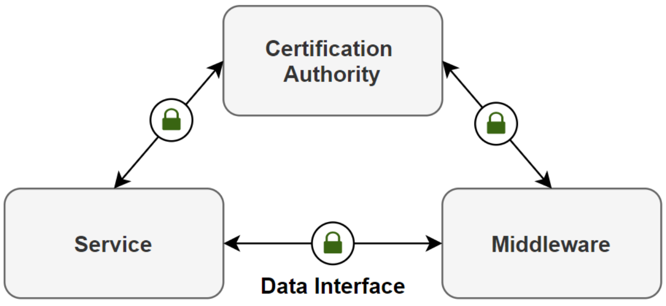

Figure 4 Relations between actors in the Data interface scenario

5.1 The Data Interface

The Data Interface is a channel that is used by the services run by the stakeholders to access the middleware platform and the functionalities that are provided. The access to the Data Interface is supplied as a library in two programming languages – Java and Python. In order to use it, a programmer has to create a service (Java or Python application), embed the provided library and use the provided functions to create an instance of the Data Interface. The instance connects to a given middleware server. Once the connection is established, the service can use the Data Interface to:

a) **Create variables**

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

Each stakeholder can create variables with a given name, value, type and a list of sub-values with corresponding names and types. After the variable is created, the stakeholder is given full permissions to her data-space in the variable.

b) **Read and manage data in a variable**
Each variable has a virtual and dedicated space for each stakeholder where the stakeholder can write, read, update and clear data. The data is stored as a set of rows.

c) **Grant or revoke access to data in a variable**
In order to access data that belongs to a different stakeholder, the accessing stakeholder has to ask the target stakeholder for permission or wait until the target stakeholder grants access manually.

d) **Subscribe to changes in a variable**
Each stakeholder can subscribe to events that are related to variables, such as read, write, update or clear.

e) **List variables, permissions and subscriptions**
Each resource can be listed with pagination and sorting support.

f) **Obtain system status information**
Each stakeholder can fetch status of the target middleware instance that allows to check the instance condition or server load.

5.2 The Security and Privacy Module

The middleware introduces a concept of a Security and Privacy Module (SPM). It is like a key store, but with additional functionality. The technical users (certified technicians) setup the necessary hardware and software in the stakeholder premises. The SPM stores data in an encrypted file in a location only accessible by the service running on behalf of the stakeholder, of whom the data is protected by the SPM. The encrypted file contains the signed certificate (by the certification authority) and the corresponding private key. The SPM can be used to store further files or values that require secure storage.

5.2.1 Initial configuration

When a stakeholder (here specifically an end customer) desires to participate in the ebalance-plus platform, a certified technician has to perform initial setup and configuration at the stakeholder premises. Before the technician pays a visit, the stakeholder is verified and an agreement is made. On successful verification and agreement, necessary files and documents are generated:

a) **The encrypted SPM file,**
b) **An envelope with password that unlocks the SPM file.**

Based on the necessary level of security, the envelope is sent by post or delivered when the technician visits the stakeholder premises to perform the initial configuration.

When the certified technician arrives at the stakeholder premises, the necessary hardware (the CMU) is placed and connected to the necessary networks, as stated during the verification and agreement process. The hardware comes with the correct software preinstalled and the technician places the encrypted SPM file on the hardware which is left inside stakeholder premises. At no point during the configuration phase, the password is exposed to the technician. Along with the password, the envelope contains instructions for the stakeholder on how to unlock the SPM file and connect to the system.

In case when the envelope with password or the SPM file is lost or compromised, a new SPM file with a different password has to be generated and the envelope delivery procedure

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

has to be repeated. If the hardware is already installed at the customer premises, a certified technician has to replace the encrypted SPM file with the new one.

After the initial configuration is complete and the stakeholder successfully unlocks the SPM file, the communication with the system can take place. In order to communicate with the middleware, a stakeholder needs a service that will provide functionalities (background work or graphical user interface). Such services can come pre-installed when the certified technician performs the initial configuration and use automatic update system to stay up-to-date.

The installed certificate can be renewed in two cases: 1) it is about to expire, or 2) it has been put on a black list. In the first case it can be made automatically, while the latter actually requires an installation from scratch (by a technician).

5.2.2 Establishing secure channels

The Figure 5 presents all of the modules, actions and relations that are present in a successful connection attempt between two actors.

When a service (Actor #1) connects to the middleware server (Actor #2), it presents the signed certificate that is stored in the SPM. The middleware server verifies the signature issuer and queries the Certification authority to check if the certificate is revoked. If all checks are successful, that is:

a) **The calculated signature of the service signature is correct,**
b) **The certificate has been signed by a trusted certification authority,**
c) **The certification authority has not revoked the certificate,**

the connection attempt progresses. In the next step, the middleware presents its signed certificate to the service. The service performs the same checks as the middleware server, signature is verified, certification chain and revocation status is checked. If any check fails, the connection attempt is rejected. The middleware servers and services cache the result of previous queries to certification authority for short and configurable periods of time in order to prevent service downtime due to connection issues. If the connection attempt is successful, a private channel for message exchanging is established where each stakeholder is certain of the identity of the other participant and messages can be exchanged securely. After a secure channel is established, the actors can exchange messages.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

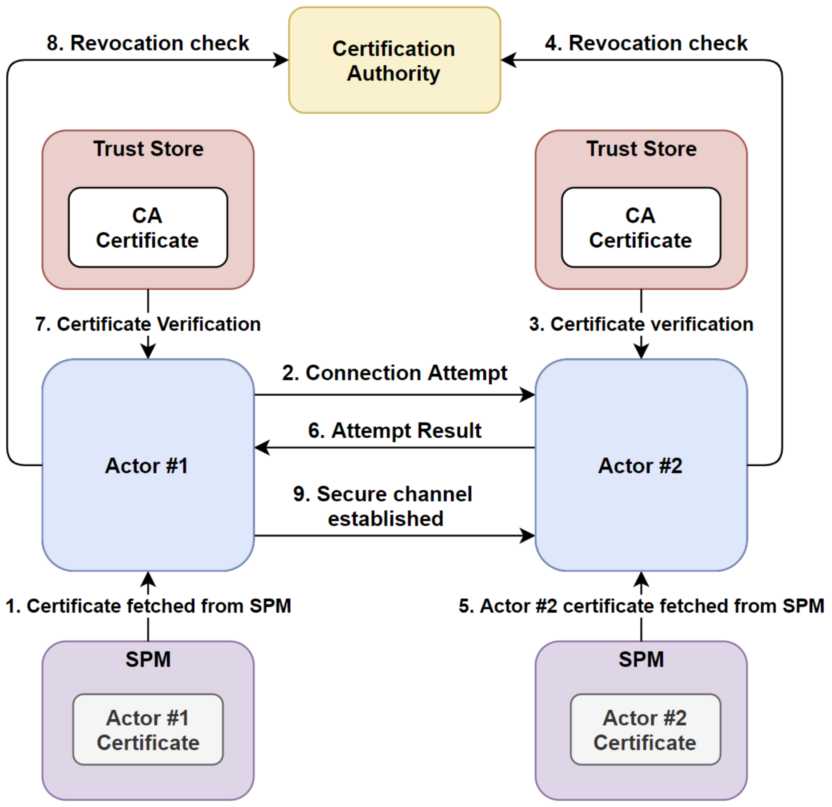

Figure 5 Connection attempt between two actors

5.3 Adapter security

It is important to remember that services can be used as adapters that pass messages from/to multiple destinations or provide user interfaces that allow to interact with the data or other stakeholders. The channels that provide adapter functionalities must also be protected in order to prevent passive listening or active data altering attacks. The service acting as an adapter can work as a server, a client or a peer. Adapters are like services, thus can also be provided by third parties and in that case their implementation is out of scope of the ebalance-plus project. However, for the services and adapters to be implemented by the project the defined rules need to be followed. For adapters specifically, each case includes different security issues that must be addressed.

1) **Case #1 – the service acts as a server**
   In this case, the service can be characterized as a functionality provider and is responsible for providing and enforcing security measures. For example, when a service provides a graphical user interface in the form of a web application. In order to display the interface, the necessary data has to be transmitted over a medium to the client running a web browser.
2) **Case #2 – the service acts as a client**
   The service connects to servers that provide functionalities. Most often, the service uses the security measures provided by the server that it connects to. For example, communication with cloud-based providers.
3) **Case #3 – the service acts as a peer**
   The service is a part of a peer-to-peer network where each peer is responsible for securing the communication. For example, the service is an adapter that enables participation in a generic peer-to-peer message exchanging network.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

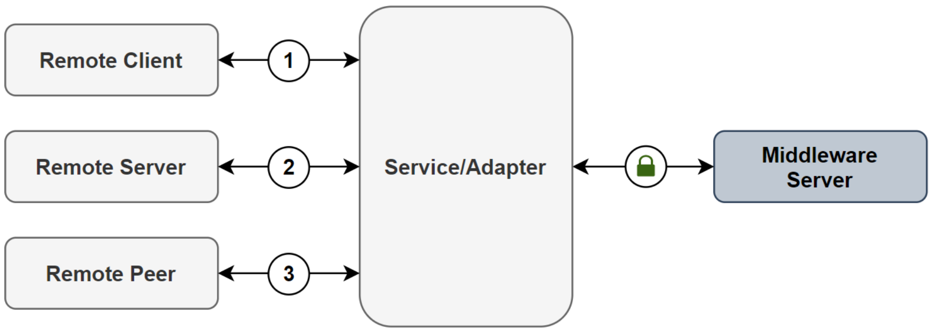

Figure 6 Services as adapters

The mechanisms used to protect the communication channels in described cases are not controlled by the middleware platform. The only way for the middleware platform to protect itself from services that use unsafe communication is to revoke a certificate of the unsafe service and cease all communication. Most of the mechanisms used for securing the communication channels make use of the public key encryption.

6 Secure bootstrap and runtime

The services that make use of the functionalities provided by the middleware platform are generic applications. By default, there are no limitations (other than limitations enforced by the operating system) in actions they may perform. In cases where a single device runs a middleware server and services that belong to a single stakeholder or services that communicate with the middleware server are run on physically separate hardware from the server and each other, the threat is minimal. However, in cases where a single device hosts the middleware server and services that belong to different stakeholders, there is a risk of malicious services that have the possibility to obtain confidential information such as private keys, databases, passwords or source code of services that belong to competing stakeholders.

In order to protect against malicious services, the middleware platform implements a utility that is responsible for:

a) **bootstrapping secure environments for services,**
b) **running services with adequate privileges.**

The utility creates a directory for each stakeholder (Figure 7). Each service that runs on behalf of a stakeholder has a corresponding sub-directory inside the stakeholder's directory. The service directory is a secure space for the service to store any files necessary to run the service, including the executable file, certificates, private keys, databases or passwords. Based on the scenario, a service can have access to all service folders of its stakeholder or only to its own folder. Access to folders of other stakeholders is denied.

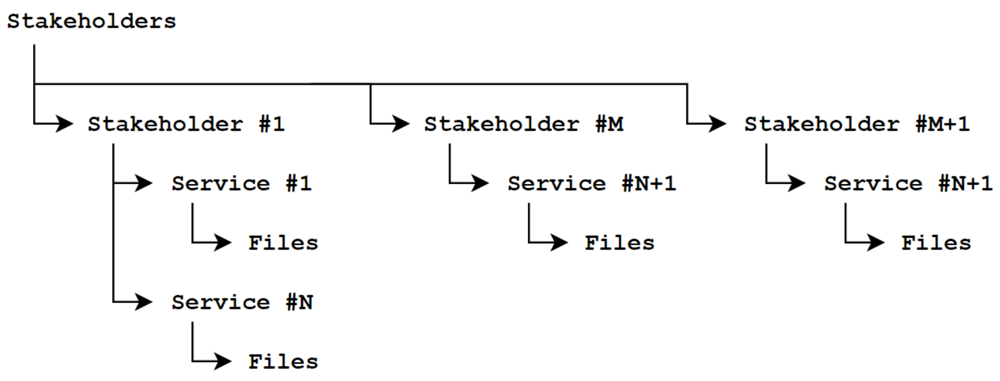

Figure 7 Stakeholder and service separation

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

The first layer of security is the file system permissions that provide protection from the outside. For the Java services, the second layer of security is Java's Security Manager [4] (SM) that provides protection from the inside. The SM allows to define policies for applications (services) and allows to protect any meaningful resource and enforce limited or specific usage (Figure 8). A similar approach will be defined for Python services.

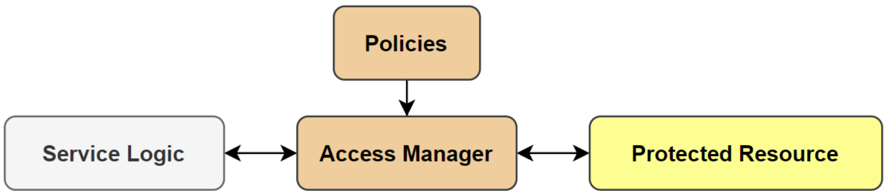

Figure 8 Relation between a service, Access Manager, policies and protected resources

The bootstrapping utility is also used to run the services. After the secure environment for a service is set up and the corresponding policies are created, the utility runs the service executable with the Security Manager enabled that controls the service behaviour in accordance to defined policies. The services are never executed without the assistance of the bootstrapping utility. Doing so could compromise the system security and render the implemented measures useless.

It is also crucial that the execution environment protects against other attacks, like those based on memory leaks. But the fact that the services are to be implemented in higher level programming languages like Java and Python may limit the danger as these are executed in their execution environments and not directly in the processors.

7 Inter-MU security

The inter-middleware (or inter-MU) communication scenario distinguishes four classes of actors – middleware instance, middleware proxy, middleware discovery server and the certification authority. The total amount of actors in the example scenario describe in the following paragraphs is six. Three middleware servers, a single proxy server, a single discovery server and a single certification authority. The Figure 9 presents each actor and relations between them (Data Interface scenario shown on the left side for context).

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

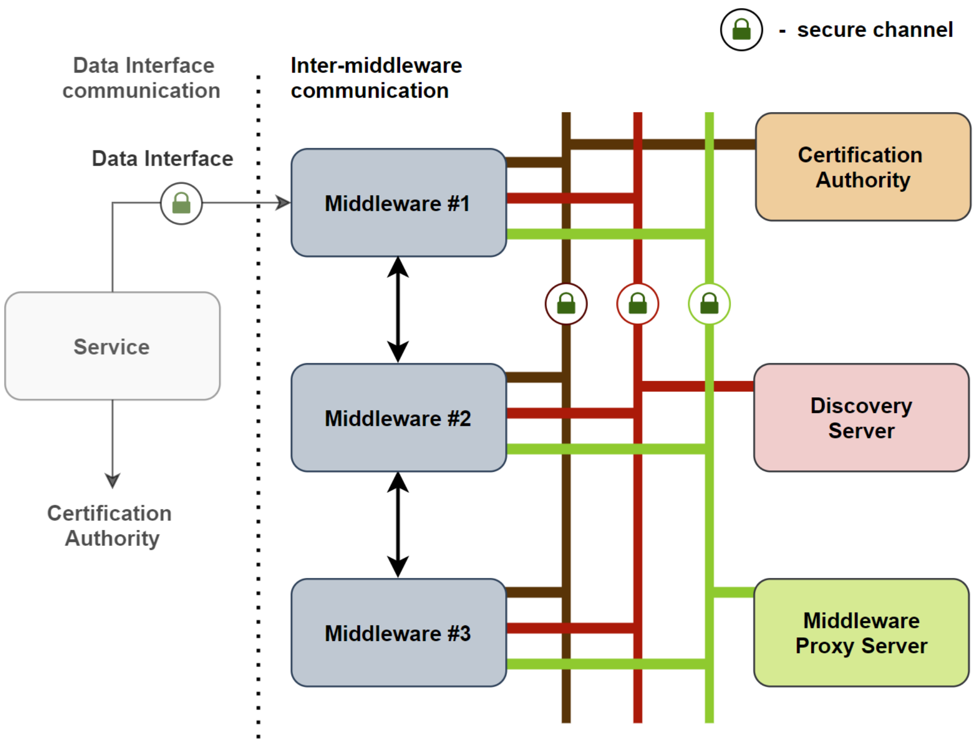

Figure 9 Inter-middleware communication

The scenario can be extended by appending further middleware instances that can be involved and exchange data either directly or through the proxy server.

The scenario has two cases in which the communication between middleware instances can happen – direct and using the proxy. When a service wants to access data from a remote middleware, the local middleware that handles the request looks into its routing table to find whether there is a direct route to the destination middleware. If a route is found, the communication happens directly. If no route is found, the middleware attempts to communicate with the destination middleware through a proxy server.

7.1 Direct communication

Each middleware contains four tables in the database – Parents, Children, Proxies and Peers. The Figure 10 shows the relation between the database and the middleware in context of storing routing information.

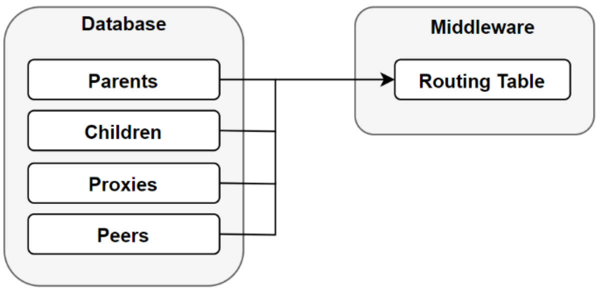

Figure 10 Routing information in the middleware

Each table contains routing information required to reach the destination device (proxy) or the middleware instances, i.e., in its simplest form as the URL. The information is handled differently, based on the table. The middleware instance may store different number of entries (details on how to reach a given target), depending on the routing table:

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

a) **Parents –** the middleware stores only the active parent,
b) **Children –** the middleware stores all entries that are configured as its children,
c) **Proxies –** the middleware stores all proxy servers,
d) **Peers –** the middleware stores all entries pointing at peer middleware instances.

From the middleware point of view, the Children and Peers tables store the same type of routing information – routes to all middleware instances given in the table. However, from the administrator point of view, the tables allow to categorize routing information.

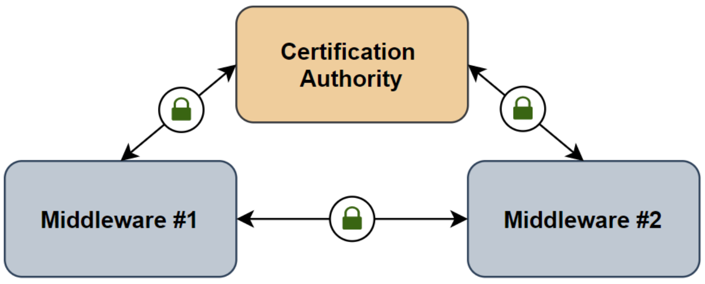

Figure 11 Direct communication

In the direct communication (shown on Figure 11), message exchanging happens between two middleware instances without the use of a proxy server. In order to start the communication, the requesting middleware has to have routing information in the routing table to the destination middleware. If the routing information is present, the requesting middleware opens a secure channel with the destination middleware, similar as it was described in Section *5.2.2 (Establishing secure channels)*. If no direct connection is present to the destination middleware, the communication cannot happen directly.

7.2 Proxy communication

In the proxy communication (shown on Figure 12), where two middleware instances cannot communicate directly, message exchanging is handled through an additional server – the proxy server.

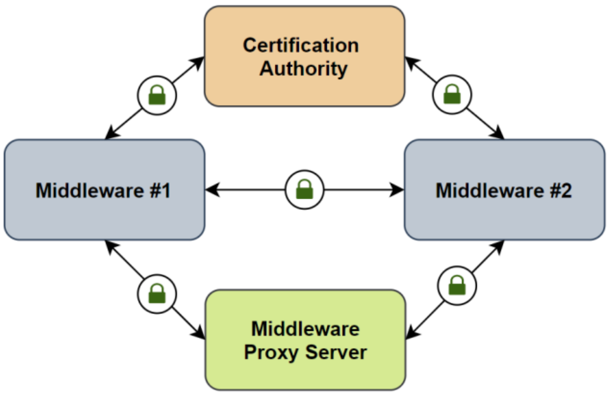

Figure 12 Proxy communication

The requesting middleware opens a secure channel (as described in Section *5.2.2* Establishing secure channels) and asks the proxy to relay a message to the destination middleware. If the proxy can satisfy the request, the message is forwarded and the proxy server waits for a response. Once the response arrives, it is forwarded through to the requesting middleware. The messages exchanged between requesting and destination middleware are not visible to the proxy server, as it is protected by public key encryption. If the

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

request cannot be satisfied by the proxy server, mainly because there is no route to the target, an appropriate error message is sent to the requesting middleware.

8 Privacy Protection (User Policies)

The middleware provides mechanisms for the stakeholders to control access to their data. These mechanisms are responsible for allowing or denying access from services based on the policies defined by the stakeholders (owners of data). Technical users can use the mechanisms directly (with an API) while the stakeholders have to rely on services that provide access to these mechanisms through graphical user interfaces.

After the connection between two management units is established (certificates signatures are verified and secure channel is created), the middleware can receive requests from local services or remote middleware servers. When a request is received, the middleware queries the access manager that is responsible for allowing or denying a request. The manager uses an internal permission system to handle each request.

The permission system implemented in the middleware is a modular permission system that uses a dot notation to represent action(s) to be performed. The dot notation system allows to represent a single action or set of actions which makes it possible to grant access to different parts or actions within the system with a single permission. The Figure 13 shows a single permission and names each component that builds the permission.

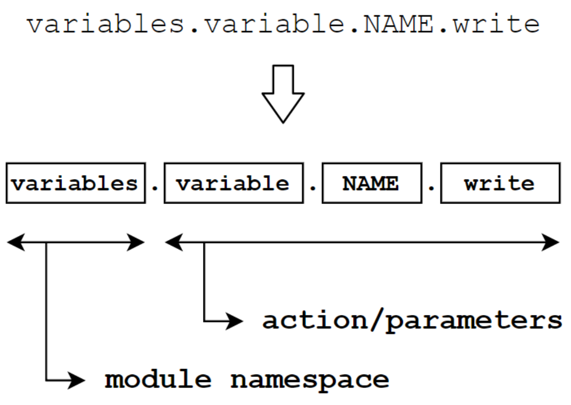

Figure 13 A single permission within the permission system

Each module implementing the middleware operations logic has its own namespace in the permissions module. The namespaces are: *variables*, *policies*, *subscriptions* and *system*. Each permission within a namespace contains at least a single action that represents an action to be performed in the related middleware logic module (e.g.: *variables.list*). In some cases, after the action, the permission contains further parameters (actions related to a single variable, e.g.: *variables.variable.NAME.write*). The approach simplifies certain activities related to granting access within the system, for example when a stakeholder consents to data access (and access kind – read/write/update/clear) by a different stakeholder, the action can be stated in a single request and stored as a single entry (regardless of access kind) in the permission system which reduces the complexity, simplifies interpretation and allows to easily revoke the consent.

When the middleware platform is started, the database-related module performs a check for missing database tables. If the permissions table is missing, the table is created by the database-related module and the middleware logic is notified. After receiving a notification on a missing permission table, the middleware inserts default permissions that allow each stakeholder to perform basic operations. The default permissions (Table 3) are generic and do

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

not grant any special privileges. It is up to the system/administrator that uses the middleware instance to grant any special permissions that allow to perform privileged operations.

Table 3 Default permissions

Permission	Comment
`variables.[add,list]`	Allows any service to list existing variables and create the space to write new values in the context of given variable for given stakeholder.
`policies.list`	Allows any service to list existing permissions.
`subscriptions.[add,remove,list]`	Allows any service to add, remove and list subscriptions.
`system.status`	Allows any service to fetch system status.

The Table 4 shows the structure of the permissions table used by the middleware with some example settings. Table 5 contains the description of the columns shown in Table 4.

Table 4 The permissions table (Omitted columns: id, query_service, query_stakeholder)

permission	owner	owner_unit	service	stakeholder	min_delay
**	*	*	*	*	0
variables.variable.NAME.*	ACME	MU1	Example Service	ACME	0
variables.**	*	MU1	OWNR Service	OWNR	0

Table 5 Permissions table columns description

Column Name	Description
`permission`	The string representing the given permission.
`owner`	When specifying permissions in the variables namespace, this field makes it possible to grant access to variable values that belong to specific owners.
`owner_unit`	Used to grant access only from specified unit (another middleware).
`service`	Specifies the service that given permission has been granted to.
`stakeholder`	Contains the stakeholder that given permission has been granted to (running the defined service).

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

`min_delay`	Specifies the minimal delay in seconds between accesses (cooldown).

The permissions module supports two types of syntaxes: permission and entity. The permission syntax is valid only in the permission column while the entity syntax can be used in the columns: *`owner`*, *`owner_unit`*, *`service`* and *`stakeholder`*. The selectors for both syntaxes are shown in Table 6 and in Table 7.

Table 6 Permission selectors

Selector	Description
`*`	Matches a single permission nesting permission level.
`**`	Matches multiple permission nesting permission levels.
`[a,b]`	Matches any permission level given in the list.
`<a,b>`	Matches any permission level that is not present in the list.
`?`	Matches a single character in a permission level.

Table 7 Entity selectors

Selector	Description
`*`	Matches one or more characters in a single entity name.
`?`	Matches a single character in a single entity name.

The Table 8 and

Table 9 show example expressions for both the permission and entity matching syntax.

Table 8 Example permission expressions

Expression	Valid	Comment
`**`	Yes	Matches any permission.
`a.b`	Yes	Matches a permission that is exactly *`a.b`*.
`a.*`	Yes	Matches any single next level in a permission that starts with *`a`*. Matches: *`a.b`*, *`a.b2`*, *`a.b3`* Does not match: *`a.b.c`*
`a.**`	Yes	Matches all levels in a permission that start with *`a`*. Matches: *`a.b`*, *`a.b.c`*, *`a.b.c.d`*
`a.*.c`	Yes	Matches any single level in a permission that starts with *`a`* and ends with *`c`*. Matches: *`a.b.c`*, *`a.b2.c`* Does not match *`a.b.zzz.c`*

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

`a.[b,b2].c`	Yes	Matches a permission that is either *a.b.c* or *a.b2.c*.
`a.<b,b2>.c`	Yes	Matches a permission that does not contain *b* or *b2* in the exact level between *a* and *c*. Matches: *a.zzz.c* Does not match: *a.b.c*, *a.b2.c*
`variables.**`	Yes	Grants full access to the variables module.
`variables.list`	Yes	Allows to list variables.
`variables.*`	Yes	Allows to add a variable or list variables.
`variables.variable.*`	No	The permission is incomplete and will never match any request.
`variables.variable.*.read`	Yes	Allows to read all variables.

Table 9 Example entity expressions

Expression	Valid	Comment
`*`	Yes	Matches any entity name.
`a*`	Yes	Matches any entity name that starts with *a*. **Example matching names:** *abc*, *abcd*
`a?`	Yes	Matches any entity name that starts with *a* and ends with any single character. Example matching names: *aa*, *ab*, *ac*
`a*z`	Yes	Matches any entity name that starts with *a* and ends with *z*. **Example matching names:** *abz*, *aabbbz*
`a?c`	Yes	Matches any entity name that starts with *a*, contains any single character and ends with *c*. **Example matching names:** *abc*, *aac*, *acc*
`a,b,c`	Yes	Matches any entity name that exists in the list separated by commas. **Example matching names:** *a*, *b* or *c*
`a,b*,c`	Yes	Matches any entity name that exists in the list separated by commas. Additionally, the second selector matches any name that starts with *b*. Example matching names: *a*, *b*, *bbb*, *bbbbbb*, *bcbcbc*, *c* **The entity syntax supports separating selectors with a comma.**

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283

The Table 10 presents a complete list of the permissions available in the middleware. The permissions have been fragmented for better readability (see Figure 14).

`variables.variable.N.read` ➔

variables.	variable.	FRUITS.	read

Figure 14 Fragmentation of permissions shown in Table 10

Table 10 Complete permissions list

Permission				Comment
variables.	**			Full access to the variables module.
	add			Permission to add variables.
	list			Permission to list variables.
	variable.	**		Permission to fully control all variables.
		<NAME>.	*	Permission to fully control a single variable.
			read	Permission to read a variable.
			write	Permission to write to a variable.
			remove	Permission to remove a variable.
			update	Permission to update a variable. (Existing entry)
			clear	Permission to clear a variable table from data.
policies.	*			Full access to the policies module.
	add			Permission to add permissions.
	remove			Permission to remove permissions.
	list			Permission to list permissions.
subscriptions.	*			Full access to the subscriptions module.
	add			Permission to create subscriptions.
	remove			Permission to remove subscriptions.
system.	*			Full access to the system module.
	status			Permission to fetch middleware status.

9 Conclusions

This document provides the definition of the security mechanisms to be applied in the ebalance-plus platform to protect the system, the stored data and the user privacy. These presented mechanisms aim at making the distributed data exchange platform (the middleware) securely accessible from all the services that implement the energy management algorithms and are running on the different management units distributed within the energy grid. It should work as if the data storage would be centralized and locally protected. This requires the unsecure communication channels to be secured (encrypted communication) and the communication peers to be easily and securely identifiable.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

Further, the services, as the data access requesters are identified as well. Using the identity of a service together with the identity of the stakeholder that runs this specific instance of the service, the data owner can define the desired use of her data, stating who (stakeholder) can use the data for what purpose (service) and how the data may be accessed (e.g. for reading or writing). The identities are checked at the connection initialization and can also be verified on the basis of every access or a session can be maintained to optimize the effort.

These presented mechanisms need to be combined with standard IT security approaches, when it comes to securing the management units. These have to be protected against unauthorized access on the OS level, as well as by physical protection (attacks number 1 and 2 from *Table 2*). The presented approach allows to minimize the effects of a successful attack. If a MU was accessed by an attacker and it has been compromised, putting it on a black list limits the damage to that device only. And the attacker cannot connect to the system with its credentials. And even if the attacker would manage that, in order to access data on remote MUs the attacker would need valid credentials of a service/stakeholder, for which the data owner granted access (attacks number 5, 6, 7 and 8). Encrypting the communication channels protects against eavesdropping and modifying the exchanged data (attack number 3 and 4).

But besides the security aspects it is also crucial to provide reliable communication. If it is based on common communication channels it can be often attacked by attacking the infrastructure. In this case the system in question will also not work properly as the data cannot be exchanged, what can also lead to critical effects. Further aspect is related to human factors. Even if the system is protected like a tank, a weak password or a bad configuration can make it vulnerable to attacks.

The security and privacy mechanism, introduced in this document, combined with the communication protocol stacks identified before, during the specification of the networking technologies (project deliverable D5.1), will be used by the implementation of the middleware that will be further used to implement the energy management platform. The platform prototypes will be evaluated in the different demo sites.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement N°864283

References

[1] e. consortium, “The e-balance project homepage,” 2017. [Online]. Available: http://ebalance-project.eu/. [Accessed 01 02 2021].

[2] IETF, “RFC-3647 - Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework,” November 2003.

[3] W. Stallings, Cryptography and Network Security: Principles and Practice, 1998.

[4] Oracle, “The Security Manager,” [Online]. Available: https://docs.oracle.com/javase/tutorial/essential/environment/security.html. [Accessed 14 01 2021].

This project has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under Grant Agreement N°864283