POLITECNICO DI TORINO

MASTER's Degree in Mechatronic Engineering

MASTER's Degree Thesis

Integrating ISO15118 standard to implement V1G and to unlock Energy Flexibility in the automotive

Supervisors

Prof. Marcello CHIABERGE

Ph.D. Massimo REINERI

Candidate

Alessandro GRASSO

October 2022

Abstract

The protection of the planet and particularly the control of climatic alterations is a theme that is becoming more and more neuralgic in today's society and is the reason that pushes for new developments and changes in everyday activities. Since vehicles emit CO2 and many other greenhouse gases, being one of the major pollution causes all around the world, electric vehicles are becoming more and more crucial to partially solve the climatic change problem. The electric vehicles and all the e-mobility panorama are in their first years, therefore it is all evolving fast, trying to accomplish user and energy needs. The objective of this thesis is to integrate ISO15118 to create a Proof of Concept (PoC) focused on the V1G and the unlocking of energy flexibility in the automotive. The ISO15118 standard main benefits are the improvement of the quality of the user journey during the charging process with the new feature called "Plug & Charge", which allows the user to start the electric vehicle charging just by plugging the cable in, without the use of any easy-to-lose cards; the scheduling of the charging process based on the e-mobility needs and the renewable energy availability, integrating "smart charging" and "energy flexibility"; stronger data security with TLS 1.2 protocol, ensuring confidentiality, data integrity, and authenticity. The PoC implements the communication between the electric vehicle and the charging station, following the ISO15118 requirements, with the implementation of the open-source protocol stack "RiseV2G". The graphical user interfaces are implemented with the use of two tablets (one for the electric vehicle and one for the charge point), in order to allow the user to express his/her preferences about the payment option and the charging profile. The electric vehicle and the electric vehicle supply equipment operations regarding the charging communication are implemented with the use of two micro-computers and two tablets. An accurate testing procedure has been made to validate the PoC behavior in all the possible use cases.

Acknowledgements

I would like to thanks my supervisors Ph.D. Massimo Reineri and prof. Marcello Chiaberge for the opportunity to collaborate with Accenture and for their support during the thesis months.

This project would not have been possible without the support of the "Team Max", I am very fortunate to have been a part of this group.

I am grateful for my family’s unconditional and loving support, which kept me always motivated and excited during these five years. Without them, this graduation would have remained a dream.

Thank you to my partner, Greta, for always listening to me, even after long days at work and during difficult times, for cracking jokes when things became too serious, and for giving me the best advice everyday and the love to be strong in front of any difficulties encountered during the journey.

Lastly, I would like to thank my friends and every single person that I met during the journey who helped me reaching this goal.

If at first you don’t succeed, laugh until you do...

Table of Contents

List of Figures

Acronyms

EV	Electric Vehicle
BEV	Battery Electric Vehicle
HEV	Hybrid Electric Vehicle
PHEV	Plug-in Hybrid-Electric Vehicle
FCEV	Fuel-Cell Electric Vehicle
ICE	Internal Combustion Engine
AC	Alternate Current
DC	Direct Current

V2G	Vehicle to Grid
PFC	Power Factor Converter
BMS	Battery Management System
AMI	Advanced Metering Infrastructure
RTU	Remote Terminal Unit
IoT	Internet of Things
G2V	Grid To Vehicle
V2X	Vehicle To Everything
V4G	Vehicle For Grid
V2V	Vehicle To Vehicle
V2L	Vehicle To Load
V2H	Vehicle To Home

SoC

State Of Charge

EMS

Energy Management System

CP

Charging Point

WPT

Wireless Power Transfer

DWPT

Dynamic Wireless Power Transfer

BSS

Battery Swapping Station

EVSE

Electric Vehicle Supply Equipment

CCS

Combined Charging System

LLC

Low Level Communication

PWM

Pulse WIdth Modulation

PnC

Plug and Charg

BPT

Bidirectional Power Transfer

PKI

Public Key Infrastructure

ACD

Automated Connection Device

OEM

Original Equipment Manufacturer

HLC

High Level Communication

RPT

Reverse Power Transfer

EVCC

Electric Vehicle Communication Controller

SECC

Supply Equipment Communication Controller

EMSP

E-Mobility Service Provider

EIM

External Identification Mode

EXI

Efficient XML Interchange

V2GTP

Vehicle-to-Grid Transfer Protocol

UDP

User Datagram Protocol

TCP

Transmission Control Protocol

TLS

Transport Layer Security

ECC

Elliptic Curve Cryptography

CA

Certificate Authority

CPO

Charge Point Operator

CSO

Charging Station Operator

CPS

Certificate Provisioning Service

GUI

Graphical User Interface

BL

Buisness Logic

Chapter 1

Introduction

The protection of the planet and particularly the control of climatic alterations is a theme that is becoming more and more neuralgic in today's society and is the reason that pushes for new developments and changes in everyday activities. Since the industrial revolution, the human behaviour is dramatically speeding up global warming, causing the increasing of the global average temperature, extreme weather, rising sea levels, disappearing glaciers, and disruption of habitats that could cause many plant and animal species to extinction. Global warming is caused by a high concentration of greenhouse gases in the atmosphere, and one of the primary ways these gases are released is by burning fossil fuels. As stated by the World Health Organization (WHO), 91 % of the people in the world live in a place that has a pollution level that far exceeds the threshold set by the WHO[1].

Vehicles emit CO_2 and many other greenhouse gases, being one of the major pollution causes all around the world. This is why electric vehicles are becoming more and more crucial to partially solve the climatic change problem. In the last years, vehicle manufacturers are making a very significant effort moving toward the electric world, but the change can not be immediate: many years are required to let the infrastructure be ready to support a large number of users with electric cars. The electric vehicles and all the e-mobility panorama are in their first years, so it is all evolving fast, trying to accomplish user and energy needs. The objective of this thesis is to create a PoC focused on the ISO 15118 standard, which the main benefits are the improvement of the quality of the user journey during the charging process and the scheduling of the recharge based on the e-mobility needs and the renewable energy availability.

1.1 The electric vehicle

The electric vehicle solution was born almost simultaneously with the first thermal engines at the end of the 800. After a first phase in which both solutions managed to coexist, the invention of the quadricycle with an internal combustion engine, invented by Herry Ford, such vehicles took over the automotive market. That is because of the limited technology related to heavy, bulky, and with little autonomy batteries, and the absence of control systems. Thanks to the great availability of energy on board, low refueling times, high technological and economic development, the internal combustion machines dominated the automotive market throughout the twentieth century. Today, thanks to the technological development of batteries and electronics, the growing demand for electricity on board, and the introduction of new legislation that aims to both reduction of polluting gases and CO_2, both on the increase in efficiency and reduction of the primary energy used, new automotive solutions have been introduced: hybrid vehicles and electric vehicles.

Hybrid electric vehicles(HEV) have the possibility to obtain energy from two different energy sources, usually a battery and a fossil fuel. The scope of this type of vehicles is to combine the advantages of both drive systems and balance out their disadvantages: reduction of the fossil fuels use and of the greenhouse emission, acceleration faster and more dynamic thanks to the electric motor, reasonable achievable distance range. During braking or coasting, the kinetic energy is converted back into electricity – this is known as recuperation. HEV are composed by the following main components:

- **ICE**: Internal combustion engines provide exceptional drivability and durability, getting its energy from diesel, gasoline or alternative fuels (such as natural gas);
- **Electric Motor**: it can act as motor, drawing energy from the battery to drive the vehicle, or as a generator recovering energy from the car to refuel the batteries;
- **Inverter**: it connects the electric motor with the battery. Its scope is to

convert the DC voltage coming from the battery into the AC voltage needed for power generation in electric motor;

- **Electronic control device**: hybrid vehicles are very complex, so the presence of a controller to automatically switch the energy source from electric to ICE and vice-versa, in order to guarantees that the vehicle runs efficiently.

Plug-in Hybrid-Electric Vehicle (PHEV), are part of the HEV family, and they can recharge their batteries at any charging point. Unlike the HEV, the PHEV have a much more efficient battery, allowing the vehicle to cover more than 100 kilometers.

Fuel-Cell Electric Vehicle(FCEV) are hydrogen-powered vehicles, which use a fuel cell [3] to generate electricity which is then used for propulsion through an electric motor. Today, hydrogen fuel cells are used almost exclusively for vehicular applications. Usually, these vehicles are also hybrids, in the sense that they also have a battery that allows them to exploit braking energy. The battery is necessary as fuel cells (at least those in use today) are not reversible, therefore they are not able to convert energy from electricity into chemistry (i.e. to generate hydrogen).

Battery Electric Vehicles (BEVs) have only one energy source, namely an electric battery. BEVs need to be frequently connected to an electric power source to recharge the battery pack. The typical range of a full charged BEV is estimated to be around 100-150 km, but high-end models of the BEV can cover about 300-350 km. The simplest form of a BEV comprises a motor, clutch, a multi-speed gearbox, and a differential. The latest models of BEVs consist of the electric motor being directly placed in the wheel, thereby reducing the overall size and weight of the entire vehicle [1]. A simple design of the electric vehicle can be composed by a **battery charger**, a **battery**, a **controller** and obviously the **electric motor**.

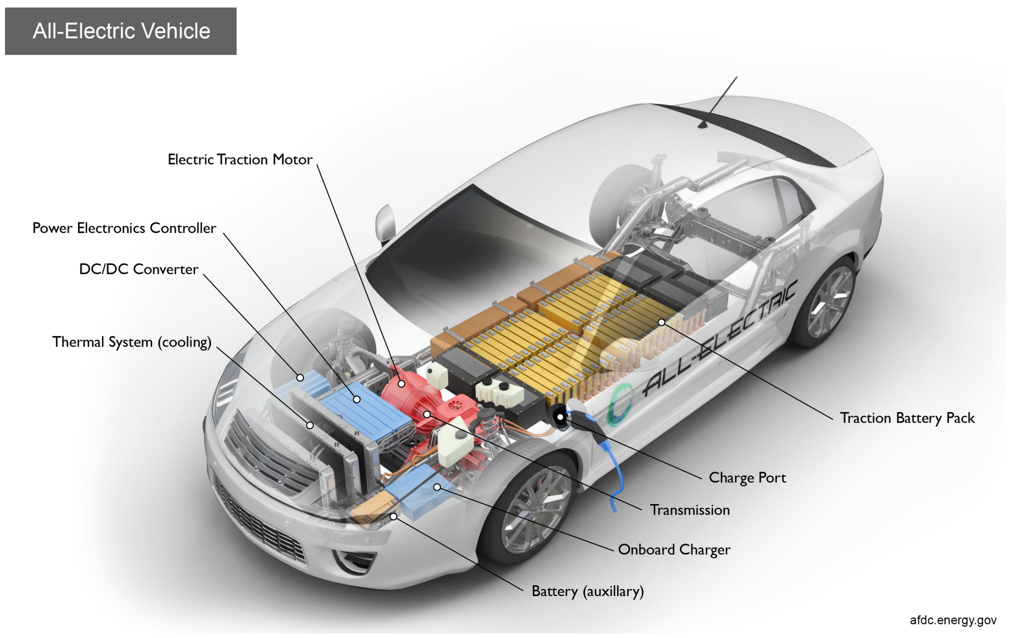


Figure 1.1: The structure of an Electric Vehicle (BEV) [1]

As described in [1], there are different **motors** that can be used in electric vehicles word. The most suitable ones are DC motors, permanent magnet synchronous motors (PMSMs), induction motors, switched reluctance motors (SRMs), and brushless DC motors. DC motors were used heavily in HEVs and EVs thanks to their simple construction, lower cost, and better speed regulation, but the presence of commutators and brushes boosts the maintenance costs and decreases the number of applications in high-speed scenarios. DC motors also have low efficiency, low reliability, and bulky construction which guided the car manufacturers to look for better alternatives for electric vehicles. Induction motors have been considered as traction motors due to their robustness, reliability, and low maintenance. The problem is the reduction in efficiency when used in high-speed applications, the lower efficiency compared to permanent magnet synchronous motors, and the low power factor. PMSMs have a simple structure, high efficiency, and high power density. The main disadvantage is that this type of motor can also be demagnetized due to the armature reaction. Anyway, PMSMs are being used commercially by some car manufacturers. SRMs are robust and have exceptional speed-torque characteristics. Unfortunately, the high noise, the torque ripple, and the tendency of electromagnetic interference make this motor category less employed than the others. BLDC motors are a variation of permanent magnet DC motors in the regard that the stator and rotor of the motors are interchanged. The lack of brushes, their high performance, low weight, durability, and excellent controllability make BLDCs the best option for electric vehicles. In fact, PMSMs and Brushless DC motors, compared to the other type of motors, consume less fuel and, consequently, contribute less to pollution thanks to a less percentage of greenhouse gas released.

The **battery** is one of the most important components of the electric vehicle, since it affects the performance, the maximum range, and the time spent at the charging point. The cost of the battery is high, and their lifespan is not so long, so the researchers are looking for new technologies to make a more efficient battery pack. Nowadays, the most implemented batteries are the nickel-metal hydride (NiMH), the lithium-ion (Li-ion) and the lead-acid batteries.

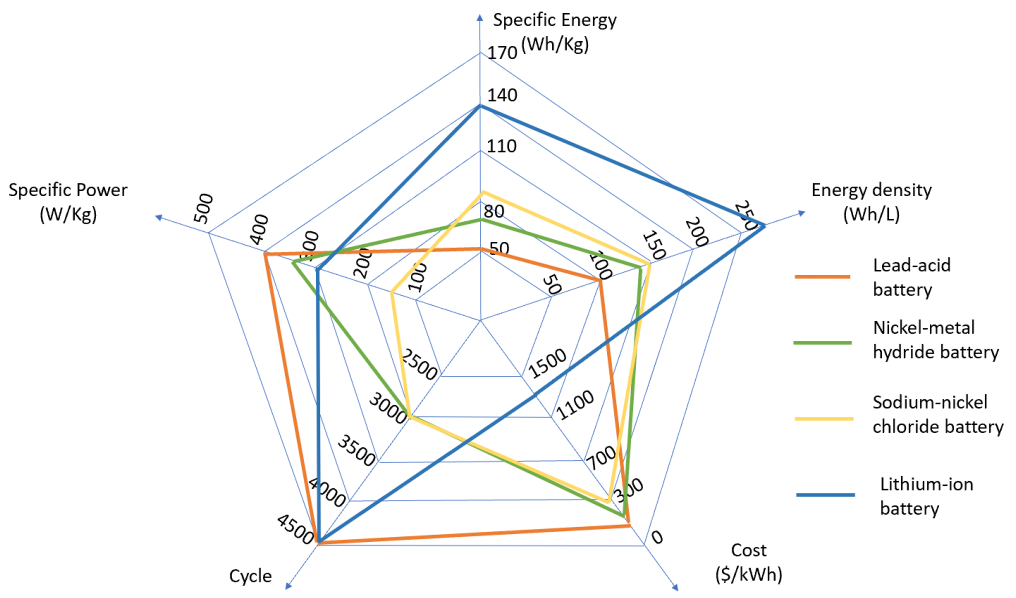


Figure 1.2: Comparison of the most used batteries [2]

According to [2], the **lead-acid batteries** were the first to be implemented, and they are still one of the most widely used types of rechargeable battery. This is because they are the cheapest type of battery available on the market. Lead-acid batteries are classified into two categories: engine starter batteries and deep cycle batteries. As the name may suggest, the first ones come into action when the motor start, providing a peak of current. They are recharged by the automobile's alternator (component able to convert mechanical energy into electrical energy). Deep cycle batteries are designed to be regularly deeply discharged, and they are used as a source during the normal ride. Their impact on the environment is low, thanks to the fact that in the EU and USA, the collection and recovery rate of lead-based batteries has now reached more than 99%. Nevertheless, they are not so implemented in the electric vehicle field due to their cost, their lower specific energy, and lower energy density.

Even if they are not the most used, the **nickel-metal hybrid** batteries have up to double the specific energy and a greater energy density than the lead-acid ones. Unfortunately, according to figure 1.2, they are expensive and have lower charging efficiencies than other batteries. For this reason, they are not so used by the car manufacturer.

Even if the nickel-metal hybrid batteries are safer than the **Li-ion** batteries, these are the most used in the EV world. Thanks to their higher energy density and efficiency, their long cycle life, and their compactness, they dominate the market and are used by the most important car manufacturer, such as BMW, Tesla, and Nissan. As described in [1], studies show that these batteries had the least wheel-to-wheel energy consumption as well as the lowest greenhouse gas emissions. Unfortunately, Li-ion batteries are quite expensive due to the scarcity of lithium, so extensive research is being conducted, in order to find an alternate energy source such as supercapacitors or graphene batteries.

As defined in [2] and in [3] , the **charging system** play a key role in the vehicle and in the battery functionalities. Charging time, battery life and maximum range of kilometers achievable strictly depend on the quality of the battery system. Modern charging systems are composed of a boost converter, that is capable of realizing a power factor correction (PFC) and then adapting the current value to the desired one.

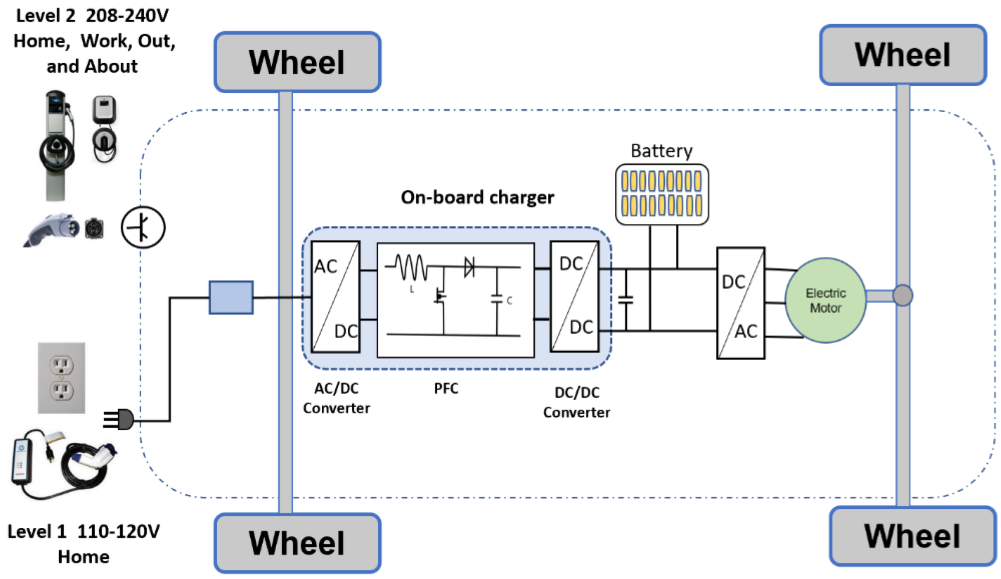


Figure 1.3: Layout of the on-board battery charger [2]

They can be classified as off-board or on-board chargers with one-way or two-way power flow. The on-board charger is installed in the vehicle, and its scope is to convert the AC arriving from the charging station into DC, since the battery can only be charged with direct current. The on-board charger reduces the complexity of the charging station, however, it increases the weight of the vehicle, causing the reduction of the maximum range achievable. Furthermore, the presence of the on-board charger means a higher cost for the car manufacturer. With the off-board charger, instead, the AC-DC converter is installed in the charging station and allows DC fast charging. The user can connect his/her car to an urban area or a highway refueling station. As will be discussed later in this document, the two-way power flow charger allows the V2G technology, where the energy can flow back from the vehicle to the grid. These types of chargers are more complex than the one-way chargers, which only allow the energy to flow from the charging station to the vehicle.

Another fundamental device inside the EV is the **battery management system (BMS)**. It is an embedded system composed of custom-designed electronics, with specific algorithms that manage the EV rechargeable battery pack. Its main objective is to preserve the battery life and to let it work at its maximum efficiency. The fundamental functions of a BMS are:

- Monitoring of both the single cells and the voltage and current across the battery. These measurings, adequately reprocessed by the microcontroller, are used to determine the battery state of charge and the maximum charge/discharge current that the battery can withstand in those operating conditions. In fact, an incorrect charge or a deep discharge could damage the battery;

- Protect human safety of device's operator, detecting unsafe operating conditions;

- Communicate to the vehicle controller how to use the energy available from the battery pack, limiting the maximum power provided by the battery pack

1.2 Smart Grid

The actual electrical grid is aging and outmoded, becoming impractical for the coming future. Energy shortage, climate change, and environmental pollution have become global issues affecting people's quality of life and world economic sustainability. Therefore, there is an absolute need to change the current energy development and utilization patterns and find a green development path.

A smart grid can be defined as an electricity network that, supported by innovative technologies such as mobile applications, Internet of Things (IoT), big data analytics (BDA), and cloud computing, can integrate the behavior and actions of all users connected to it, monitoring and controlling the power system in both ways from power stations to end-users or vice-versa [4]. Unlike traditional grid, the smart grid provides several aspects that are becoming fundamental in the e-mobility word, such as availability, effectiveness, accuracy, controllability, flexibility, interoperability, maintainability, measurability, reliability, sustainability, stability, and security, as defined in [4]. These requirements of the smart grid allow the power network to be more stable, robust and reactive to any failure that can occur in the grid, detecting in advance vulnerabilities, power outages and blackouts.

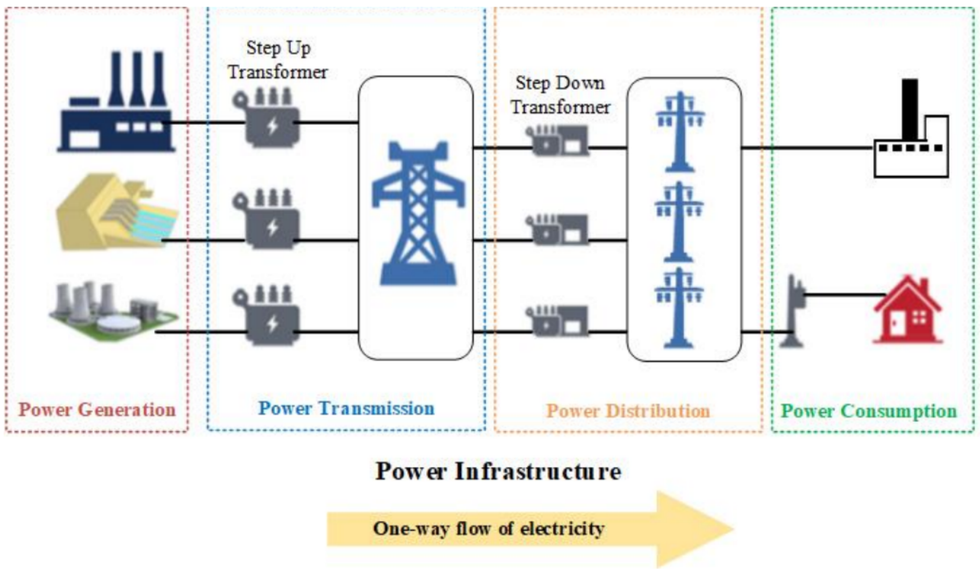


Figure 1.4: Structure of the power grid [4]

As depicted in figure 1.4, from a power point of view, the smart grid can be divided in four main parts:

- **Power generation**: generation of electricity used by all the elements in the smart grid. Their scope is to obtain electricity from several sources of energy, renewable and not renewable. The main scope of the smart grid is to increment the use of renewable sources (wind, solar, hydro, biomass, bio-fuel, geothermal, wave, tidal, and fuel cell) more than the non-renewable ones (fossil fuels, coal, petroleum, nuclear and natural gas). As stated in [4], in 2017, the most used power source is fossil fuel, with a production of 66.7% of all energy products in the World. The second energy source in terms of use is the Hydroelectric power source with a share of 16.3%. The third most widely used energy source is nuclear, whereas renewable energy sources are far away from these numbers.

- **Power transmission**: transportation of electricity is a crucial step in the smart grid world: the voltage is at its maximum and the current at its minimum, in order to reduce the costs and losses. One of the jobs of the smart grid is to measure the voltage and current values along the transmission lines, which can be overhead or underground, in order to respond in an adequate manner.

- **Power distribution**: Upstream of the power distribution, there is a step-down transformer, which levels down the voltage and rises the current, in order to provide a suitable form of power to the smart grid consumers.

- **Power consumption**: with power consumption, all the elements that consume energy are represented, such as electric vehicles, houses, and factories. In the future, the number of consumers will significantly increase, especially electric vehicles, so a robust grid is necessary to prevent possible frequent problems.

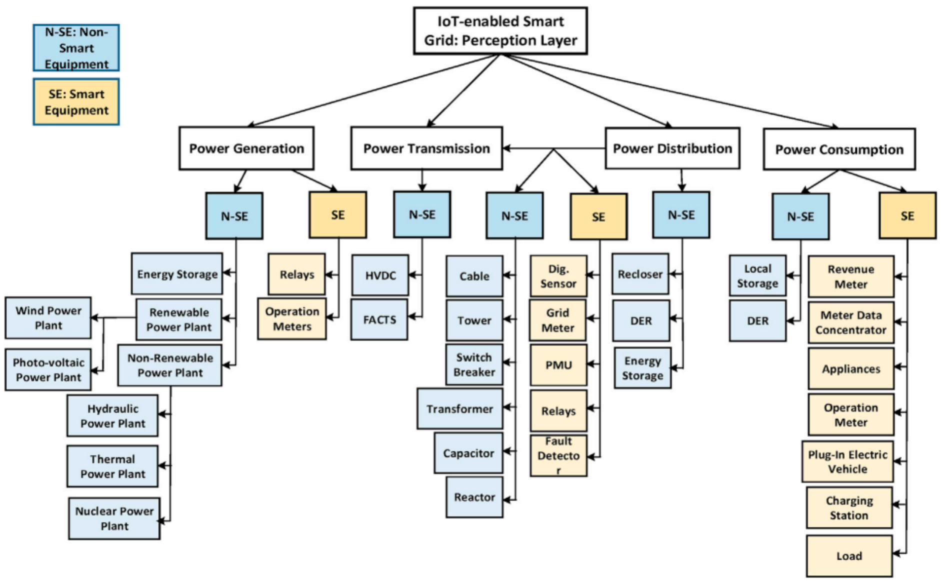


Figure 1.5: Power grid integrated with IT [5]

In the smart grid, the power organization is integrated with the advanced Information Technology (IT) and communication networks to deliver electricity with improved efficiency and reliability, while reducing costs and environmental impacts [6], as shown in figure 1.5. In addition to the power grid, the smart grid is composed of sensing devices, information gateways, and the Advanced Metering Infrastructure (AMI). The smart meters and the sensor continuously communicate with each other and with the smart grid, providing measurements about the energy consumption, energy losses, systems performances, and glitches [7]. These information are also send to the central controller through gateways which function is to collect the data coming from the AMI, accomplishing the conversion of protocol and communication among two heterogeneous networks (wide area network (WAN) - home area network (HAN)), and propagate it to the controller. One of the main advantage of the smart grid is **energy flexibility**: the smart grid has to adapt the energy in order to manage variations in demand and load handling, so a crucial design is required for the communication system, as shown in figure 1.6.

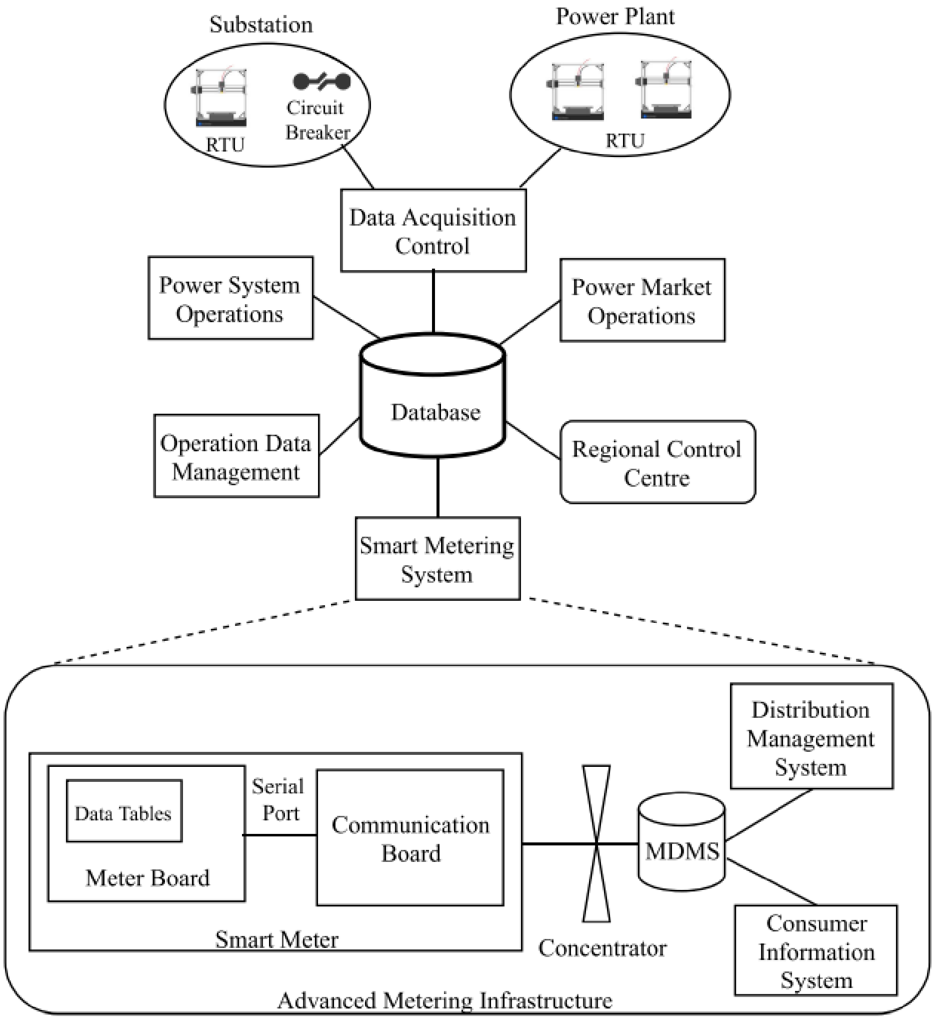


Figure 1.6: Smart grid communication system [7]

The smart grid is organized in different regional control centers, which will communicate with the smart metering system and will supervise different substations and power plants. Meanwhile, all the crucial information are sent to power system and power market operators to handle the energy process, and to Operation Data management which analyzes the data through cloud computing and big data processing. One of the major challenges related to the smart grid is the processing and analyzing of big data coming from all the elements present in the grid. As very well described in [6], all the data captured by the sensing elements and smart meter must be processed in real-time, in order to improve efficiency and reliability. Traditional data-sets are not powerful enough o manage all these heterogeneous data-sets, usually named big data: advanced management approaches are needed, such as big data analytics (BDA).

It's fundamental to organize and identify the features of each stakeholder who contribute to the development of smart grid. As already stated, every component must communicate with the other, so every organization must apply to the rules and make their part. Therefore, complex relations with all economic and social parties such as government, power generation enterprises and users (such as residential, commercial, small and large industrial consumers).

In the smart grid there is a special category of consumers: prosumers. They are those consumers who can produce electricity using small-scale energy generators installed at their houses or vehicles and then feed forward the electricity to the grid. In particularly, vehicle-to-grid, vehicle-to-home and other technologies are analyzed in the next paragraph.

1.3 Vehicle-To-Everything technologies

As reported in [8], the sale of electric vehicles around the globe has increased by 46% during the year 2018-2019, and, in the coming years, a rapid expansion is expected: by 2030, there will be around 140-240 million of EVs in the world. Knowing that EVs spend most of their time parked, they can be used as stationary energy storage. To help mitigate climate change, we need to consume more renewable energy than fossil fuel energy. The main disadvantage of renewable energy sources is their volatility: unlike the fossil fuel energy source, we can't decide when to produce energy and so utilize it. The energy coming from sources such as wind and solar power must be used (or stored) the moment it is produced. Electric vehicles, thanks to their rapidly increasing number, can help with this problem, working as "power banks" of the smart grid, storing energy coming from renewable sources and then giving it back through V2X (Vehicle-To-Everything) technologies.

Before describing the V2X technologies, it is necessary to understand what is defined with **V1G**. V1G is the simplest form of electric vehicle unidirectional smart charging: it allows energy flexibility, adjusting the power demand/response according to the energy availability. With V1G, several data are exchanged between the electric vehicle and the charging station, in order to prefer a charging profile that meets the user needs but also the renewable energy availability. The EV is still not capable of returning its energy back to the smart grid. If the EV state of charge (SoC) is above the value requested by the user (e.g. the user wants to charge its vehicle at 80% rate, if the SoC is 81% or above, the vehicle enters the V2G mode) the electric vehicle enters the **Vehicle-To-Grid (V2G)** mode operation, being able to return back its energy to the grid, using the battery pack as an energy source for the grid. The control of the EV operation mode is visualized in figure 1.7, where G2V means Grid-To-Vehicle, so the normal charging of EV. With V2G, there are benefits for every element of the Smart Grid: in fact, the EV owner will be economically rewarded to offer this service.

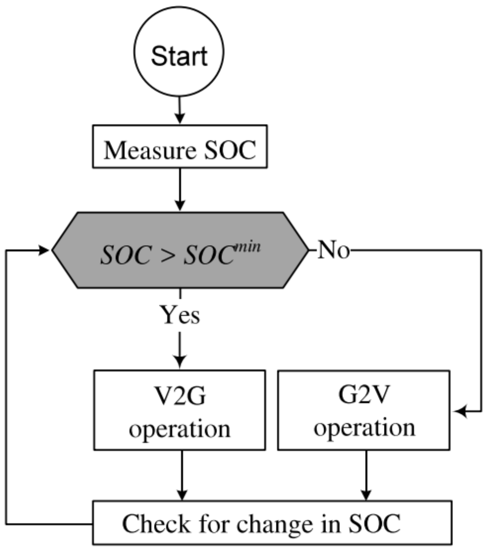


Figure 1.7: Transition between V2G and G2V [8]

As very well described in [8], V2G technology allows the vehicle to provide several services to the grid:

- **Reactive power support**: During the peak load period, the EV injects reactive power into the grid in order to back up renewable energy sources and minimize the difference between reactive power requested by the load, and reactive power that the grid can provide at that moment. During the light load period, instead, the EV is charged. It is very important to highlight that enabling V2G technology doesn't impact the EV SoC and the battery lifetime. Anyway, to ensure reactive power support, sophisticated controllers, AC-DC and DC-DC converters are required and researchers are trying to find the most suitable ones: the unified controller can provide an excellent dynamic response and steady-state performance; the sliding mode controller (SMC) is able to accommodate unknown disturbances during operation; hysteresis current controller resolves the SMC issues, is simple to implement and ensures fast dynamic response. Even if its implementation may be expensive due to the digital platforms required, Energy Management System (EMS) based energy box controller is able to optimize the output of distributed energy sources and ensure smooth control of active and reactive powers in V2G or G2V operation mode. In fact, this technology ensures optimal battery modules charging/discharging profiles, enhancing the reliability of the system.

- **Shaving of peaks and filling of valleys in load demand**: This feature is accomplished by trying to convince the consumer to recharge its EV during the light load periods, and then offer their EVs to give back the stored energy during the periods in which the energy is more required: this is also called **energy flexibility**. Smart charging can help with this problem, as it creates a charging profile based on the user's needs, trying to increase the energy provided to the user during the periods in which the total energy required from the smart grid is expected to be moderate and to decrease the charge during the hours in which the smart grid operators require high quantities of energy.

- **Compensation of harmonics in grid current**: EVs in V2G operation mode behave as an active filter and provides harmonic compensation, improving the grid power quality and acting against the nonlinear behaviour of the smart grid loads. To accomplish this feature, a large number of EVs are required but, they must be very well managed by a central sophisticated controller: lack of coordination between EVs can decrease the power quality of the smart grid.

- **Minimization of generation uncertainty in renewable energy sources**: As already described, one of the main benefits of V2G is the possibility to back up the intermittent output of renewable energy sources. The EV act as a controllable load, in order to require more power when energy from renewable sources is produced but not required from the grid (so, it would be lost) and provide back power when renewable sources are not producing it but the demand is increasing. However, this service contributes to the EV battery degradation, reducing its lifetime, therefore being an important cost for the user. Also, in this case, smart charging comes in help, trying to predict when the EV will act as a load or as a source, and then adjust the charging profile to protect the battery from destructive routines.
- **Minimization of frequency deviation**: The difference between the nominal and actual value of smart grid frequency is called frequency deviation, and this can be caused by an inequality between the active power requested by the loads and the active power provided by the smart grid. Frequency deviation can cause blackouts and voltage fluctuations, as described in [8]. Nowadays this problem is resolved with the usage of a large number of batteries acting as spinning reserve, but since this is not economically convenient, EVs batteries can be used instead.

The **Vehicle for grid (V4G)** technology allows the EV to act as an active power filter, compensating the harmonics in line current and providing reactive power when needed. A vehicle can also work as V4G and V2G simultaneously, since V4G is a special case of V2G, as defined in [8].

Another very important technology is the **Vechicle to home (V2H)** which is formed when the EV owner charges his/her vehicle at home. This network is then formed by a single EV and a smart home and it is very easy to recreate in real life, as described in [9]. V2H is one of the operation modes with the highest efficiency and improves the utilization of renewable sources of energy. The vehicle, connected to the home, can act as an energy source, contributing to reducing the household daily load profile. As discussed for the V2G, the vehicle can inject reactive power into the home grid or even into the community grid. Anyway, every application in

V2H can interact with V2G, V2V and V2L, in order to synchronize every operation and create a more reliable smart grid. If the home is equipped with renewable energy systems, V2H comes in help by acting as a power bank to store the energy produced by the solar panel. The EV battery degradation must be taken into account since a significant reduction in its lifetime is inevitable.

By using **Vehicle-to-Vehicle (V2V)** technology, the vehicles can redistribute their energy between EVs parked at charging points or connected to smart homes. To allow this technology, an aggregator is required: as defined in [9], it is a controller whose objective is to collect information about the EVs and the smart grid and then synchronize the energy transmission between all the elements involved. The aggregator group a specific number of EVs and allow the possibility to take the energy from a fully charged EV and give it back to an electric vehicle that still needs to be charged. When all the EVs under the management of the aggregator have reached their desired SoC, the controller starts the interaction between the smart grid, offering the EVs as energy sources. V2V has simple infrastructure requirements and small transmission losses.

Last but not least, also the **Vehicle-To-Load (V2L)** technology can be implemented by the EV, as described in [8]. It is a special case of V2H and V2V operation modes: the EV is used to supply power to critical users, such as hospitals or data centers, in case of malfunction of the grid.

The potential of V2X technologies is substantial, but there are still a lot of challenges to be solved. First of all, economic terms must be defined: every stakeholder must have a positive income out of the implementation of V2X technology. Another crucial aspect is the battery pack degradation: a lot of services provided by the EVs in V2X operation modes require the charge/discharge of the battery, causing the reduction of the battery lifetime, degradation of the battery, and the reduction of the maximum range of kilometers that the EVs can achieve without charging again. As explained before, the BMS is responsible for controlling the charging/discharging of the battery, protecting it from aging and degradation. Everything in the smart grid is connected to everything, therefore, the whole system is exposed to cyber-attacks. The attacker injects high-frequency signals into the smart grid communication lines, taking control of the system. Now, the controller has no longer access to the smart grid, and the attacker can take all the sensitive data available (e.g passwords, data related to user privacy, ...). Still, the communication between the stakeholders must be regularized by the international standard: as we will widely discuss after, one of the main objectives of ISO 15118 is to establish a sequence of messages between the EV controller and the charging point, that must be followed to allow the V2G technology.

1.4 State of the art of the charging infrastructure

As the number of EVs will rapidly increase all around the world in the coming years, the charging technology will become crucial in the e-mobility panorama, aiming to improve the electric vehicle user journey. As a matter of fact, charging is becoming faster, more convenient, and the number of charging points is very quickly increasing.

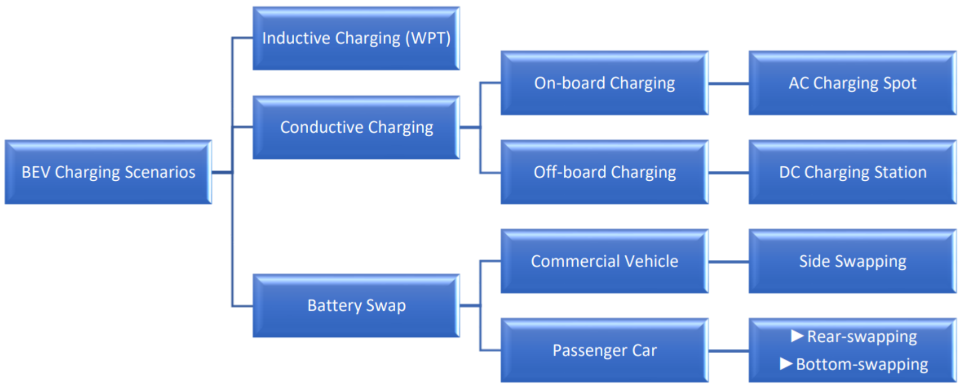


Figure 1.8: Charging solution for electric vehicles [10]

According to figure 1.8, there are three ways of charging the battery: conductive charging, inductive charging, and battery swapping.

- **Conductive charging**: As described in [2], conductive charging consists of the physical connection between the EV and the charging point (CP). The energy can be transferred in AC mode or DC mode. As already described in paragraph 1.1, the EV batteries can only be charged with DC, but the energy from the grid is AC, so an AC-DC converter must be present. If it is on the vehicle, we are talking about an on-board charging solution and AC charging mode: the CP infrastructure is simple, it provides AC energy and then the vehicle charger converts it into DC. If the EV is not equipped with the converter, we are talking of an off-board charging solution: it is mounted in the CP, so the EV receives DC. The last mentioned charging solution is called DC fast charging, since there are no limitations on weight and power of the off-board converter, unlike the on-board one that must be low weight and low seize, due to EV constraints. Therefore, the charging capability of DC charging modes is much higher than the onboard charging technique, which means less time to fully charge the EV battery pack. A simple rappresentation is shown in figure 1.9.

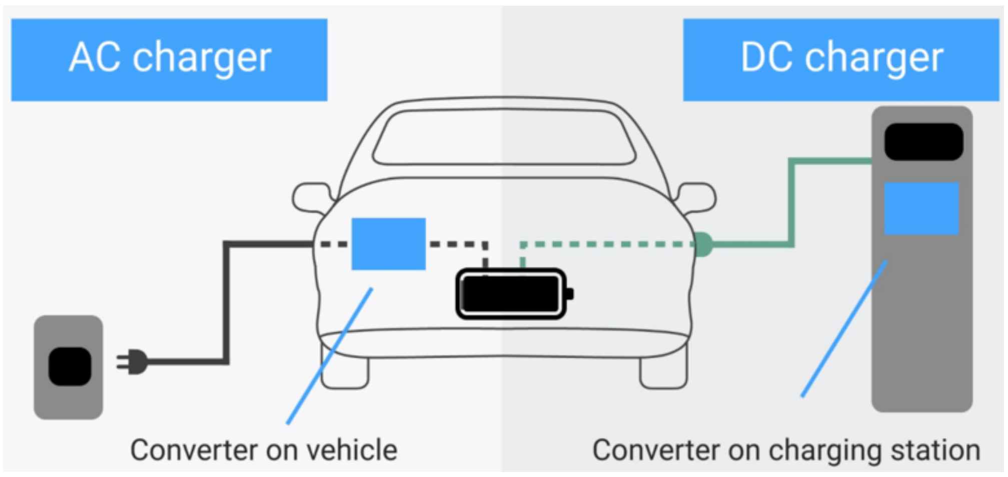


Figure 1.9: AC/DC charging

Different charging techniques can be employed with conductive charging:

– **Constant Current (CC)**: when providing energy to the vehicle, the current is kept constant and the voltage can change. This technique is used for Nickel-cadmium batteries.

– **Constant Voltage (CV)**: when providing energy to the vehicle, the voltage is kept constant and the current is variable.

– **Constant Power (CP)**: The charging point provides energy to the vehicle maintaining the power constant, therefore both the voltage and the current can change over time, but their product must remain the same.

  - **CC-CV**: the combination of the CC and CV techniques is the optimal choice for the lithium-ion batteries. To charge the battery pack, first, the CC technique is implemented, until a defined voltage value is reached. Then, the charging mode is switched to CV, until the current almost drops to zero.
  - **Pulse Charging**: the charging point provides energy through pulses of current. This is a new technology and researchers think it will become the most employed one in DC fast charging.

- **Inductive charging**: Also defined as wireless power transfer (WPT), allows the user to charge his/her vehicle without a physical connection between the EV and the CP. The energy is transferred thanks to a transmitting coil and a receiving coil. WPT can be classified into:
  - **Inductive Power Transfer (IPT)**: can be used for high power applications, with large air gaps.
  - **Capacitive Power Transfer (CPT)**: can provide low power, with tiny air gaps.

  Also, as described in [2], WPT can be classified as:

  - **Stationary charging system**: Also known as "park and charge", the user can simply park their EV and the charging process can start as conductive charging. The only difference is that the user doesn't need to connect the vehicle to the CP, since there are the transmitting coil underneath the parking place and the receiving coil installed in the EV, that will handle the energy transmission without a physical connection.
  - **Quasi-dynamic charging system**: it is similar to a stationary charging system, but it is applied to places where the vehicle stays for a very short amount of time (e.g a bus station, traffic lights)
  - **Dynamic charging system**: Also known as Dynamic wireless power transfer (DWPT), allows the charging of electric vehicles while they are moving. For example, if applied on the highways, there is no need for the user to stop at charging points, and the reachable range of kilometers significantly increases.

- **Battery swapping**: This technique allows the user to arrive at a battery swapping station (BSS) and swap their batteries for fully charged ones. Their depleted batteries are then charged at the station, so the users can always find fresh batteries ready to be mounted. This would incredibly reduce the time spent charging the vehicle, and this could also be an important service for the smart grid since fully charged batteries can stay connected to the grid and act as a power bank until an EV owner comes in and take them. Obviously, there are several aspects to be defined: for example, assume that the EV user battery has a remaining lifetime of 80%, goes to a BSS, and swap his/her battery for a battery with a remaining lifetime of 40

Several standards exist to define different aspects of the charging infrastructures and the charging process. The most popular standards, as stated in [2], are the International Electrotechnical Commission (IEC) standards, the Society of Automotive Engineers (SAE) standards and the Japan Electric Vehicle Association Standards (JEVS). Some of the most important standards related to the charging of the EV are analyzed:

- **IEC 62196**: this standard is divided in three parts, and its scope is to uniform "plugs, socket-outlets, vehicle connectors, vehicle inlets and cable assemblies" for electric vehicles, as defined in [10]. This standard is strictly correlated to IEC 61851 standard that will be discussed later in this list. It is divided in three parts: IEC 62196-1, which is defined for CP with operating voltage not exceeding 690V in AC or 600 V in DC, describes the requirements of the interface between an electric vehicle and a charging station, as well as mechanical and electrical requirements and tests for plugs, sockets, vehicle connectors and sockets of the vehicle to be used for charging EVs. Specific designs are not described, since they can be found in other parts of the standard. The focus is for accessories and cable assemblies that are meant to be used in an ambient temperature of -30 to +50 °C. IEC 62196-2 apply to EVs with operating charging voltage not exceeding 480 V AC. This part define the plug types depending on the application (Type 1- single phase vehicle couplers; Type 2- single and three phase vehicle couplers; Type 3- single and three phase vehicle couplers with shutters). IEC 62196-3 describes specific designs for vehicle connectors and sockets for vehicles to be used for DC charging of electric vehicles in mode 4 as described by IEC 61851-1 and IEC 61851-23. Specific designs are grouped into different configurations, especially CHAdeMO and CSS Combo. The designs are described in sufficient detail to allow compatibility between products from different manufacturers.

- **SAE J1772**: This standard cover the general requirements of the physical, electrical, and communication parts of the charging system in North America and Japan. Different charging solutions are categorized in levels, as described in [10]:

  - **level 1 - AC**: This charging level simply consists of connecting the EV to the household outlet, which will provide 120V, alternating current. This technology is portable and no installation of a charging point is needed. This is used when you want to charge your EV at home during the night: in fact, the time required to fully charge a battery is between 8 hours and 12 hours, depending on the EV battery capacity.

  - **level 2 - AC**: A home charging or a public charging point installation is required. This is much faster than level 1, since just 4-6 hours are required to fully charge the battery, providing a voltage of 240V, AC. This is the most diffused technology and can be found at parking areas, at home or at work place.

  - **level 3 - DC**: This technology refers to DC fast charging: it is able to charge the battery up to the 80% in just 30 minutes. An on-board battery charger is not required, since the charging point already emits energy in DC.

- **IEC 61851**: This standard specifies general requirements about charging electric vehicles in Europe and China. As described in [10], four charging modes are defined:

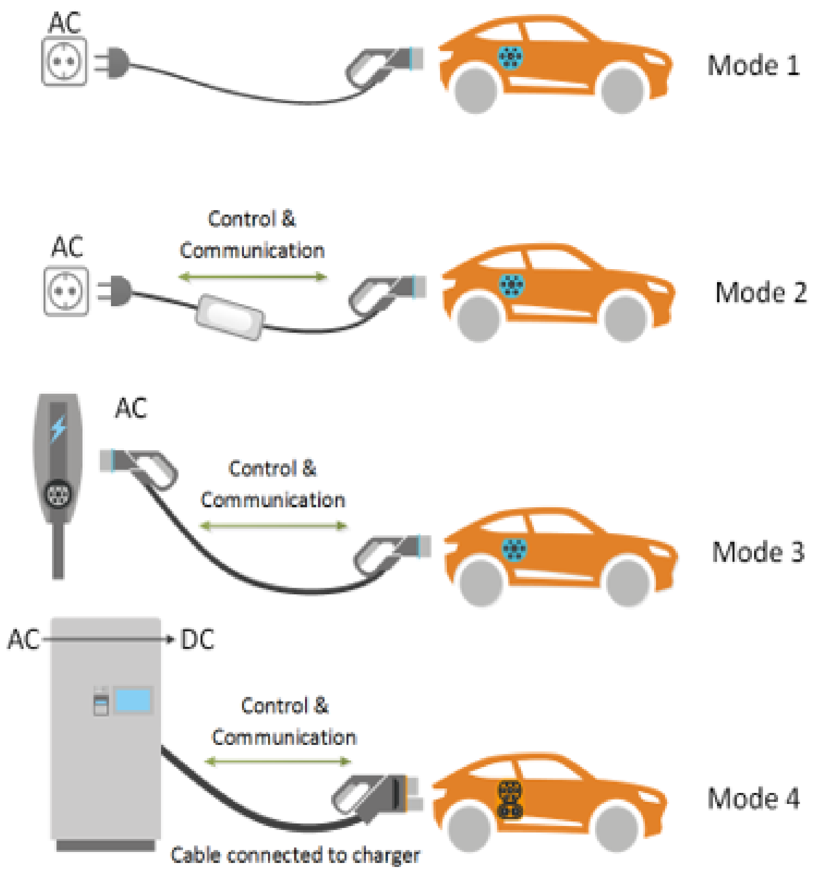


Figure 1.10: Charging modes defined in IEC 61851 standard [10]

– **Mode 1**: The first mode represents the possibility to connect the EV to the household outlet, charging with AC in slow charging. This is the simplest and cheapest way to charge the electric vehicle since no charging point installation is required. The socket must apply to several safety requirements, imposing strict limits to available power in mode 1. As of now, the current limit is 10A, being a good compromise between the user needs and safety regulations: a too high value of current would cause heating of socket and cables, together with possible fire or electric injuries. The voltage can be 250 V if mono-phase or 480 V if three-phase. The connection between the EV and the power source is direct since no control pins are present.

– **Mode 2**: Also this mode refers to AC slow charging mode, but the

Electric Vehicle Supply Equipment (EVSE) is present. This device is directly connected to the EV and to the energy source. Its scope is to provide several safety protections, such as monitoring the presence of the ground connection or the over-temperature protection. Also in this mode, the connection can be mono-phase (250 V) or three-phase (480 V), with a maximum current of 32 A. This solution is more expensive and reliable than the mode 1 solution.

- **Mode 3**: A permanent installation on the wall (EVSE) is required for this charging mode. The EV is connected to the EVSE through a specified socket/outlet. Not only energy is transmitted, but also information regarding the charging process and safety requirements. This charging mode can be slow or fast, depending on the EVSE installed.

- **Mode 4**: This mode refers to the fast charging of the EV with an external charger. The current can be up to 400 A, being an expensive but fast charging process.

All around the world exist different **plug types**:

- **Plug type 1**: It is used in the USA and in Japan, 1-phase connector with 120 V / 240 V.

- **Plug type 2**: It is used in Europe and in China. Connector with 1-phase 230 V and 3-phase 400 V grid access.

- **Combined Charging System (CCS)**: With this plug, two more pins are added to the previously explained plugs, so that with a single connector, both AC and DC charging are available. CCS1 adds the DC pins to the type 1 plug, so it is used in the USA and in Japan. CCS2 is used in Europe and in China, since it is a combination of type 2 plug and DC pins.

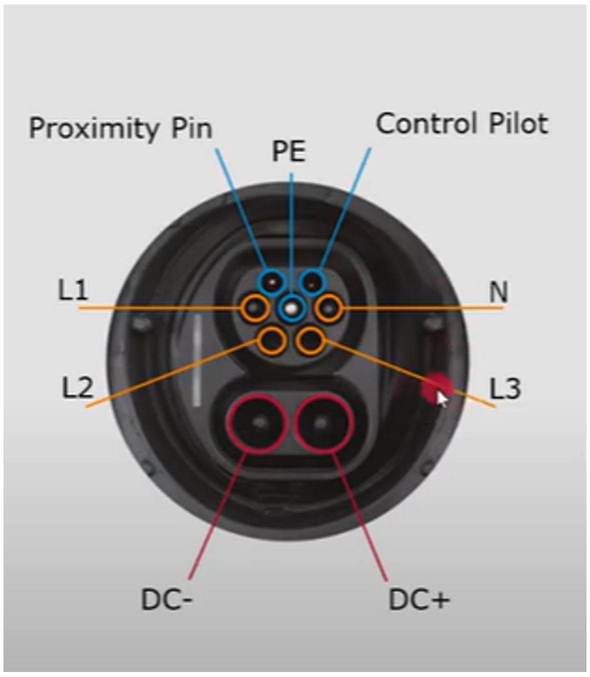


Figure 1.11: CCS2 connector [10]

As can be shown in figure 4.4, the pinout of the connector is the following:

- **L1, L2, L3**: AC charging power pins
- **N**: Neutral pin
- **PE**: Protective earth pin
- **DC-, DC+**: DC charging power pins
- **PP**: Proximity pin,
- **CP**: Control pilot

As described in [10], EVSE and EV communicate with LLC (Low-Level Communication), which doesn't require additional circuits to work. The PP is used by the EV to check the presence of the plug. EVSE and EV exchange information about their state through the CP pin, indicating for example if they are ready or not and the current capacity that the EVSE can provide (this is done through PWM signaling).

Last but not least, the **ISO 15118** standard series defines the requirements for the communication between the EVSE and the EV. Since the thesis is focused on this standard, the next chapter gives an in-depth explanation of the documents and their requirements.

Chapter 2

The ISO 15118 standard

2.1 The main features of the ISO 15118 standard

ISO 15118 standard defines the information and communication requirements to integrate electric vehicles in the smart grid, As described here [11], this standard family primary objective is to create a standardized, secure, energy-convenient and user-convenient way of charging the EV at home or in public.

Through this standardized communication, different important features are available for every part involved in the e-mobility ecosystem:

- **Plug & Charge (PnC)**: This feature allows the user to simply plug in his/her electric vehicle to the CP, and the charging process start automatically, with no need of other actions. The charging experience becomes user friendly thanks to the EV-EVSE communication based on the exchange of digital certificates. The presence of the certificates is required for security reasons and it will be analyzed after in this list. This contract certificates is installed in the EV and it enables the vehicle to automatically authenticate itself to the charging infrastructure, without the need to find the easy-to-lose RFID card to pay for the charge.

- **Smart charging**: Combining information exchanged between the EV and the EVSE, such as the amount of energy needed to fully charge the vehicle and the departure time, with information related to the state of the smart grid, the available energy coming from renewable energy sources, the best charging strategy is provided to the user. Therefore, the energy needs of the user are accomplished, together with an increasing use of renewable energy, thanks to the intelligent load management. Plus, pricing information are also taken into account, allowing the user to spend less but still satisfying his/her energy needs.

- **Strong data security**: The smart grid is a critical infrastructure with a lot of information. For this reason, hacker attacks can happen. The ISO 15118 is based on the OSI model, therefore, strong security on both transport layer and application layer is required. The three pillars of data security are achieved: confidentiality, integrity and authenticity. A Public Key Infrastructure (PKI) enables a secure exchange of information based on asymmetric cryptography (refer to paragraph 2.4 for in-depth explanation).

- **Bidirectional Power Transfer (BPT)**: In this standard, the communication required to use the V2G technology is defined. As well explained later in this document, V2G messages are defined in ISO 15118-20.

- **Wireless Power Transfer (WPT)**: The communication between the EV and the charging infrastructure to realize the "park and charge" feature is specifically defined in this document family. The user, then, can charge its vehicle by simply parking in a specified area, and the EV-EVSE will automatically handle the communication and the charging process.

In the ISO 15118 standard, the communication between the EV and the charging infrastructure is defined. More precisely, the involved parts are the **Electric Vehicle Communication Controller (EVCC)** and the **Supply Equipment Communication Controller (SECC)**. But, to ensure a secure and automatic authentication with related billing process, all the smart grid stakeholders are involved. For this reason, the PKI, which enable a secure exchange of information and certificates, is crucial and is analyzed in the paragraph 2.4 of this document.

2.2 The structure of the standard

As illustrated in the figure 4.4, the ISO 15118 standard is composed of seven plus one documents.

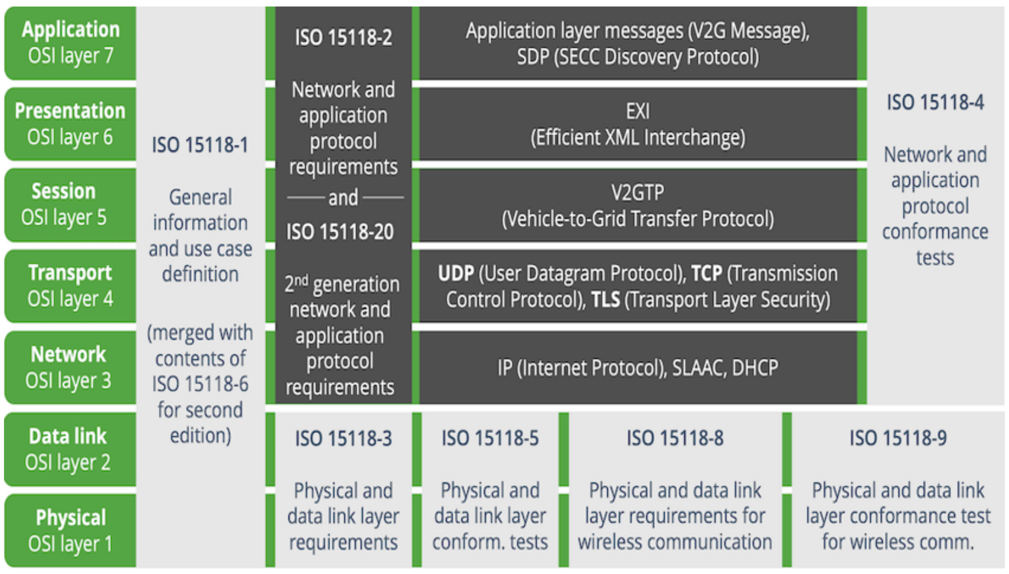

Application OSI layer 7
Presentation OSI layer 6
Session OSI layer 5
Transport OSI layer 4
Network OSI layer 3
Data link OSI layer 2
Physical OSI layer 1
ISO 15118-1
General information and use case definition
(merged with contents of ISO 15118-6 for second edition)
ISO 15118-2
Network and application protocol requirements
and
ISO 15118-20
2nd generation network and application protocol requirements
Application layer messages (V2G Message), SDP (SECC Discovery Protocol)
EXI (Efficient XML Interchange)
V2GTP (Vehicle-to-Grid Transfer Protocol)
UDP (User Datagram Protocol), TCP (Transmission Control Protocol), TLS (Transport Layer Security)
IP (Internet Protocol), SLAAC, DHCP
ISO 15118-4
Network and application protocol conformance tests
ISO 15118-3
Physical and data link layer requirements
ISO 15118-5
Physical and data link layer conform. tests
ISO 15118-8
Physical and data link layer requirements for wireless communication
ISO 15118-9
Physical and data link layer conformance test for wireless comm.

Figure 2.1: ISO 15118 document parts [11]

- **ISO 15118-1 "General information and use-case definition"**: As reported in the official standard document [12], ISO 15118-1 specifies the general requirements and use cases for conductive and wireless High Level Communication (HLC) between the EVCC and the SECC. Indeed, HLC is required to enable features such as identification, the automatic handle of payment, energy transfer control, and all the value-added services. This document is very useful to understand the basics of the ISO 15118 standard series, defining all the relevant aspects of this standard family, such as charging/discharging control, payment, identification, association, cyber-security, and privacy.

  The standard is divided into two main parts: a list of requirements to enable all the features allowed by ISO 15118, and a general definition of the use cases.

  A brief explanation of the main requirements to realize an EV-EVSE communication to enable a charging process that complies with the ISO 15118 standard is given:

  - **Energy transfer schedule**: One of the first things the user does when connecting his/her vehicle to the charging infrastructure is to negotiate a charging schedule that must end before a specified amount of time (user departure time). If for any reason, the smart grid is no longer able to respect the schedule, an error message must be shown to the user. At this point, the user can activate the renegotiation of the charging schedule.

– **Private data protection**: User data shall not be accessible from non-necessary parts. The user must always be informed about who is accessing his/her data.

– **Smart Grid energy limits**: The SECC must inform the EVCC about all the smart grid power limitations. Maximum voltage, maximum current, and maximum power limitations, based on the smart grid or the EVSE, are sent to the electric vehicle controller.

– **Authorization of charging services**: The EVSE checks if the EV is allowed to charge and asks for payment methods (cash, EMAID, or contract certificates). In the case of the contract certificates payment method, the PnC feature is enabled, and the user certificate chain is checked: this is due to privacy manner and it will be explained more in deep in paragraph 2.4 (PKI).

– **Wireless communication requirements**: To enable WPT, additional general requirements shall be specified. The SECCs may broadcast their information to identify themselves, and the EV may associate with the EVSE without the driver's actions.

– **Reverse Power Transfer (RPT) description**: To let the vehicle act as a power supply and feed back energy to the grid (or home), different requirements are specified. First of all, the EVCC must comply with local electrical safety requirements. The communication shall occur with basic signaling plus HLC. The system can be "single channel" if the same channel is used for both charging and discharging, or "dual channel" if two different power channels are used for charging and discharging the EV.

– **Traceability requirements**: During the charging loop, both the EVSE and the EV shall measure the active charging energy, reactive charging energy, active discharging energy and reactive discharging energy. At the ened of the charging process, the EVSE must be able to produce a detailed document with all the energy information related to the charging process just finished (e.g. AC/DC type of charge, total active and reactive energy transmitted, control mode used).

All possible use cases are defined in the ISO 15118-1, in order to enable the communication between the electric vehicle and the EV supply equipment.

- **Start of communication session**: This use case covers the initiation of the EV-EVSE communication session. Here, the EVSE sends a PWM signal with a duty cycle of 5% in order to require HLC: The EVCC and the SECC establish the physical and data link layer connection. If we are in the case of wireless communication, this use case covers the automatic finding of the most suitable SECC to start the charging process.

- **Communication set-up**: This use case covers the creation of a link between the electric vehicle and the EVSE. The goal of this use case is to allow the electric vehicle to send request messages on the application layer to the SECC, according to ISO 15118-2 standard messages.

- **Certificate handling**: Installation and update of the certificate are covered in this use case. The update of the certificate occurs when the current certificate is expired and needs to be replaced with a new and valid one issued from the secondary actor (e.g. e-mobility service provider (EMSP)). Here, the SECC plays a key role: it can provide the certificate itself, or act as a link between the EV and the SA that issue the certificate. To complete this action, the HLC defined in the "Start of communication" use case is used. The certificate installation, instead, is needed the first time the EV connects to a charging point. With the OEM provisioning certificate that was installed by the car manufacturer at the EV production, the EVCC request a new and valid certificate. This can be issued by the SECC or by a secondary actor (in this case, the SECC acts as a link between the EV and the SA).

- **Identification, authentication, and authorization**: Four different authentication methods are defined in this use case. A graphical overview is given in 2.2.

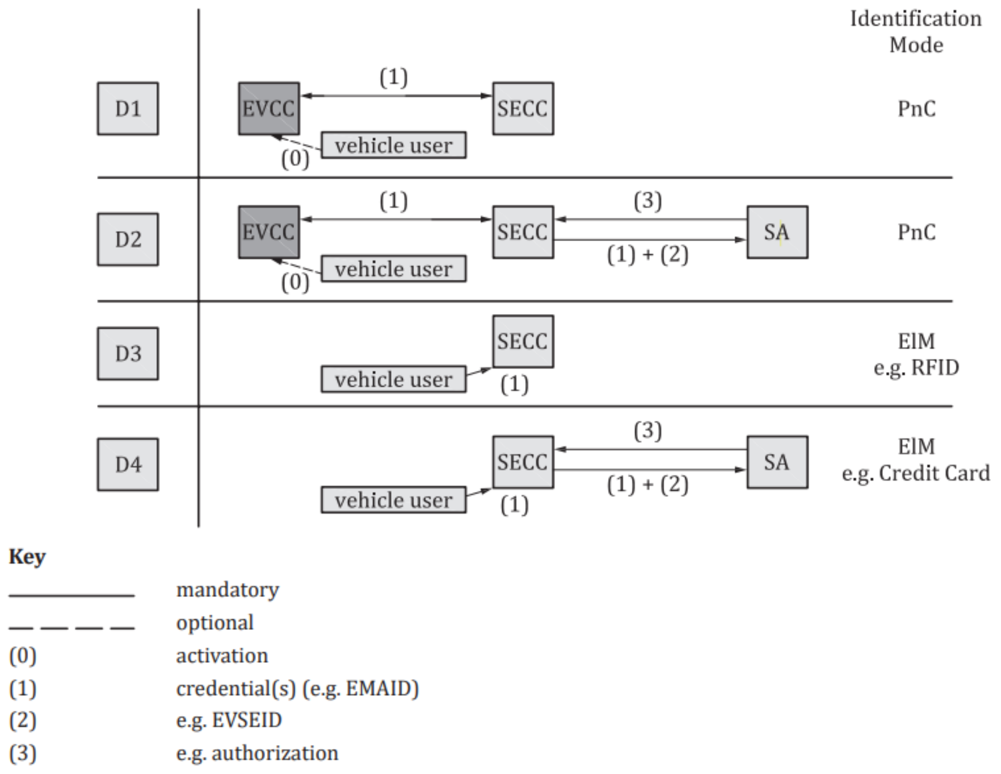


Figure 2.2: Overview of possible authentication/identification modes [12]

- ∗ **D1**: The authorization is made with contract certificates (PnC) that are verified directly at the SECC. The authorization should start automatically but if this doesn't occur, the user can activate it through the EV HMI.
- ∗ **D2**: In this use case, the authorization is made using contract certificates with the help of a SA.
- ∗ **D3**: External Identification Mode (EIM) can also be used by the user to authenticate itself and the vehicle in order to start the charging process. Credentials must be provided by the user through the HMI, for example with an RFID.
- ∗ **D4**: Authorization can be made also with a Credit Card, with the help of the SA.

– **Pairing and fine positioning**: The EVCC sends a request message to the SECC choosing among the possible fine positioning methods provided by the SECC. The electric vehicle and the EVSE inform each other about their positioning status during the process until they are correctly aligned.

- **Target setting and energy transfer scheduling**: This use case covers the charging process scheduling, the reverse power transfer, and bidirectional power transfer. The charging schedule can be made according to different factors that can be chosen by the user and the SECC: maximum power that can be drawn from the charging point, local needs, grid needs (e.g. drawn more power when produced with renewable sources), user e-mobility needs (e.g. the user must leave by the 5 P.M. and needs the car fully charged).

- **Energy transfer controlling and re-scheduling**: This use case covers all the information that needs to be exchanged between the EV and the electric vehicle supply equipment during the charging loop. Also, meter reading information can be exchanged between the actors involved. The energy transfer loop can be interrupted by the EVCC (or the user) or by the SECC.

- **Value-added services**: Different services can be added to the charging process, for example, "reservation of a public charging site, spots availability along the journey, required energy for next usage", as listed in [12].

- **End of energy transfer period**: Closing down the charging process must be done in the safest way possible: a shutdown sequence of messages is defined by the use case.

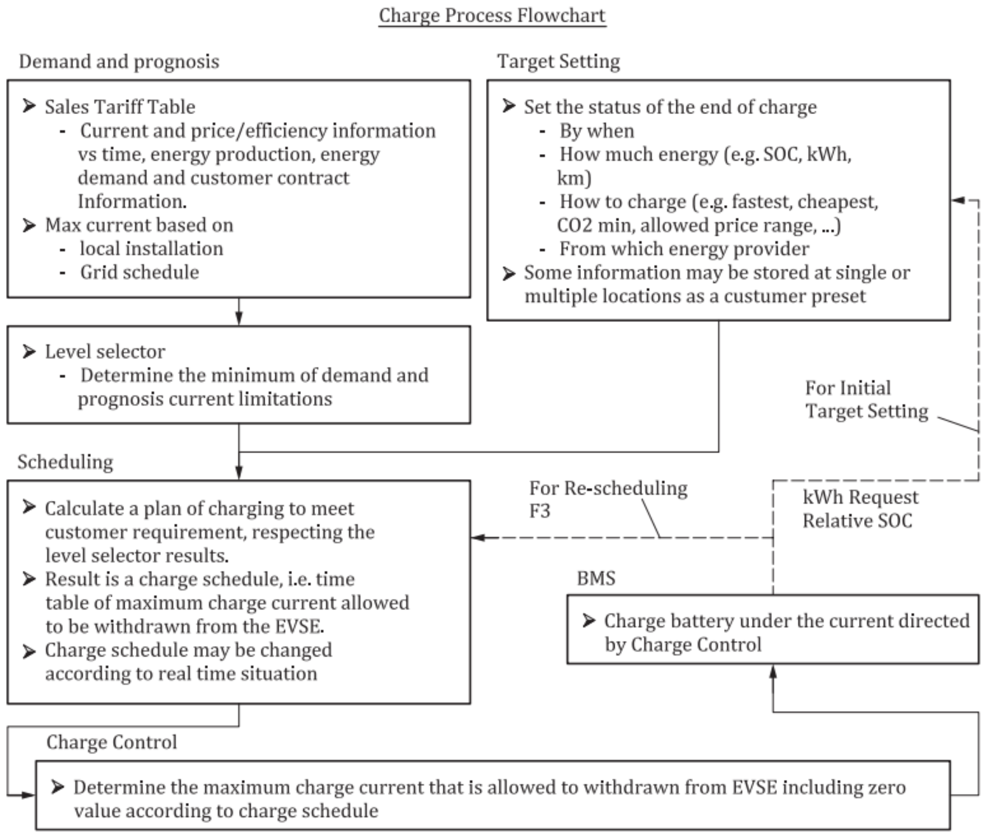

Charge Process Flowchart
Demand and prognosis
Sales Tariff Table
- Current and price/efficiency information vs time, energy production, energy demand and customer contract Information.
Max current based on
- local installation
- Grid schedule
Target Setting
Set the status of the end of charge
- By when
- How much energy (e.g. SOC, kWh, km)
- How to charge (e.g. fastest, cheapest, CO2 min, allowed price range, ...)
- From which energy provider
Some information may be stored at single or multiple locations as a custumer preset
Level selector
- Determine the minimum of demand and prognosis current limitations
For Initial Target Setting
Scheduling
Calculate a plan of charging to meet customer requirement, respecting the level selector results.
Result is a charge schedule, i.e. time table of maximum charge current allowed to be withdrawn from the EVSE.
Charge schedule may be changed according to real time situation
For Re-scheduling F3
kWh Request Relative SOC
BMS
Charge battery under the current directed by Charge Control
Charge Control
Determine the maximum charge current that is allowed to withdrawn from EVSE including zero value according to charge schedule

Figure 2.3: Charge process flowchart [12]

As shown in the figure 2.3 and described in [12], the charging process can be divided in into four different elements:

- **Target Setting** It covers all kinds of user-demand related information such as:
  - Type of control mode selected (scheduled or dynamic);
  - The e-mobility needs;
  - When the charging process is finished;
  - How much energy is needed for the charging;
  - Charging preferences like fast charging, cheapest charging, least CO2 charging, etc..
- **Demand and Prognosis** It covers all the information needed to create a charging schedule that meets the needs of every actor involved. The sales

tariff table contains the information related to the energy production and the price.

  - **Scheduling** It is the charging schedule based on the "Demand and Prognosis" that respect the user e-mobility needs. It can be changed during the transfer loop if requested by the SECC or the EVCC.
  - **Charging Control** It covers the control of the charging process according to the "scheduling". Renegotiation is executed if environmental conditions are changed during the charging loop.

- **ISO 15118-2 "Network and application protocol requirements"**: This document is the heart of the ISO 15118 family since all the application layer messages exchanged between the EV and the EVSE are defined here. The focus of the thesis project is on this part of the document series, for this reason, an in-depth explanation of this document is given in the next paragraph.

- **ISO 15118-20 "2nd generation network and application protocol requirements"**: This document is an updated version of ISO 15118-2, with some important additional features:

  - **Bidirectional Power Transfer (BPT)**: All the messages related to the V2G technology are now defined. This standard gives the means to let the EV feed back energy to the grid when connected to charging infrastructure at home, at workplace or even in the street. With this standard information, V2G and V2H scenarios are possible to be implemented.
  - **Wireless Power Transfer (WPT)**: New messages with respect to ISO 15118-2 are implemented to further improve the wireless charging process. Thanks to these new messages, the smoothest communication between the EV and the EVSE is achieved.
  - **Automated Connection Device (ACD)**: Conducting devices such as a pantograph can be used to automatically connect the vehicle to the charging infrastructure. Obviously, this is a challenging feature, with several engineering problems that need to be solved. This document specifies the messages exchanged between the EV and the EVSE to start and complete the charging process. This technology can be used for exampled for buses that will automatically to the pantograph.

  - **Dynamic Mode**: With the ISO 15118-2 standard, the user can choose between different charging schedules. With ISO 15118-20, the dynamic mode is added: no negotiation between the EV owner and the charging point, but a dynamic mode that change the energy provided with respect to the grid needs is accomplished. The EV yields control to the charging station.
  - **Stronger data security**: Longer keys for the certificates creation and a new version of TLS is used.
  - **Easier multi-contract handling**: The EV can now identificate itself with more than just a contract. This can be useful, for example, to have different certificates for different use cases: free charging, workplace charging, home charging and so on. Obviously, all of this is transparent to the user, and the OEM (Original Equipment Manufacturer) can decide how many certificates to install on the EV.

  ISO 15118-20 is a very recent standard: it has been published in May 2022, so EVs do not implement this standard. Some of the actual electric vehicles do implement the ISO 15118-2 standard. Therefore, a retro-compatibility problem raises: if both the EV and the EVSE support ISO 15118-2 and ISO 15118-20, the latest one will be used. If one of the parts doesn't implement ISO 15118-20 but both the EV and the EVSE support ISO 15118-2, this standard will be chosen. If there is no common standard supported by the parts (e.g. EVSE only supports ISO 15118-20, and EV only support ISO 15118-2), the only way to complete the charging process is to use analogue communication, based on PWM.

  As already said, the ISO 15118-20 standard has been published in May, but the thesis work started in march. Plus, open protocol stack solutions have been released a month after, and, being a new technology, I preferred to relate to the ISO 15118-2 standard and its long-year tested protocol stack solutions. This ensures a more reliable PoC, but for the future, the ISO 15118-20 features will be integrated into the demo.

- **ISO 15118-3 "Physical and data link layer requirements"**: As described in [11], this document specifies the communication at the two lowest OSI layers: data link layer and physical layer. The basic signalling and HLC general requirements are defined in standard ISO 15118-3. The EV-EVSE communication is composed of basic signalling plus HLC: HLC is used in addition to basic signalling in order to enable a bidirectional communication. The standard also specifies all the control pilot requirements both from EVSE side and EV side. The overall Plus, the interaction with standard IEC 61851 is covered. All the timings and constants of the EV-EVSE communication are illustrated in this document, together with all requirements for the HomePlug Green PHY Technology on control pilot line. The protocol SLAC (Signal Level Attenuation Characterisation) is specified in ISO 15118-3.

- **ISO 15118-4 "Network and application protocol conformance test"**: This standard, as the name may suggest, specifies the test architecture for the ISO 15118-2 standard. This part of the ISO 15118 series is related to the conformance test for the ISO 15118-2 part: a series of rules are specified and must be checked if conformance with the standard. The conformance test, as specified in [13] and in [11], consists of an Abstract Test Suite (ATS) for a System Under Test (SUT) implementing an EVCC or SECC according to ISO 15118-2. The layer 3 and all the above are being tested with this document. With this standard, the behaviour of the EV-EVSE communication at application and network layer is checked if it is conformed with the standard ISO 15118-2.

- **ISO 15118-5 "Physical layer and data link layer conformance test"**: As for the ISO 15118-4, the scope of this document is to provide the requirements to test the ISO 15118-3 standard. Therefore, all the messages related to layer 2 and layer 1 of the OSI model are checked.

- **ISO 15118-8 ”Physical layer and data link layer requirements for wireless communication”**: This part of the ISO 15118 series specifies the technical requirements for wireless charging communication on the physical and data link layers, while for all the above OSI layers, the communication is already described in ISO 15118-2. It covers the overall information exchange between all actors involved in the electrical energy exchange, and only the EVSEs compliant with the standard IEC 61851 is covered by this document.

- **ISO 15118-9 ”Physical and data link layer conformance test for wireless communication”**: This document provides the requirements to test the part 8 of this standard series. This part of of the ISO 15118 family is the only one still under development.

2.3 ISO 15118-2

The ISO 15118-2 standard specifies the necessary messages in all the seven OSI model layers: as stated in [11], the communication between the EV and the EVSE is based on the OSI model, therefore it is divided into seven different layers, each with different functions and messages exchanged. When a message needs to be sent from the EVCC to the SECC or vice-versa, it is processed from the application layer all the way down to the bottom layer (physical layer). Once the message arrives here, it is transmitted to the SECC through the physical connection. It follows a brief explanation of each layer function:

- **Application layer**: At this level, two different messages exist: SECC Discovery Protocol (SDP) messages and V2G messages. SDP messages are used at the beginning of the communication between the EV and the EVSE to communicate their IP address, in order to set up the communication. The V2G messages are used to exchange all the information related to the charging process. The V2G message is divided into a body (the information of the message), a header that contains the ID of the session, and eventually, the signing if the message information is reserved.
- **Presentation layer**: The message created at the application layer is sent to the presentation layer, where it is converted into a format that both sides can understand. In this communication, the format chosen is Efficient XML Interchange (EXI).
- **Session layer**: On this layer, the e Vehicle-to-Grid Transfer Protocol (V2GTP) is used to understand if the message is an SDP or a V2G one.
- **Transport layer**: At this level, all SDP messages are transmitted using the User Datagram Protocol (UDP), which focuses on the velocity of the message more than its security. For the V2G messages, instead, Transmission Control

Protocol (TCP) or Transport Layer Security (TLS) protocols are used. TLS must be used when the messages need to be secured by encryption, therefore every time Plug and Charge is selected.

- **Network layer**: On layer three, unique Internet Protocol (IP) addresses are given to both the EVCC and the SECC. In addition, Stateless Address Autoconfiguration (SLAAC) and Dynamic Host Configuration Protocol (DHCP) protocols are applied in this layer.

- **Data link layer and Physical layer**: This two layers are better defined in ISO 15118-3 and it is where are defined the specifics about the electric signals used to transmit the messages.

The focus of this document is about the application layer messages used by the EVCC and SECC to communicate. The communication is based on a client/server structure where the EVCC always acts as a client and the SECC acts as a server during the whole charging process. Indeed, it is always the EVCC to ask for a service through a request message, and the SECC responds with a response message.

One of the first steps of the thesis project has been to understand every message that the EVCC and the SECC send to each other, therefore a detailed explanation of the messages is provided. The charging session can be divided in AC or DC and, in turn, in EIM or PnC, resulting in four different message sessions: AC-EIM, AC-PnC, DC-EIM, DC-PnC. Some messages are common for all the communications, others are just for some of them.

The four main cases are represented by the following sequence diagrams.

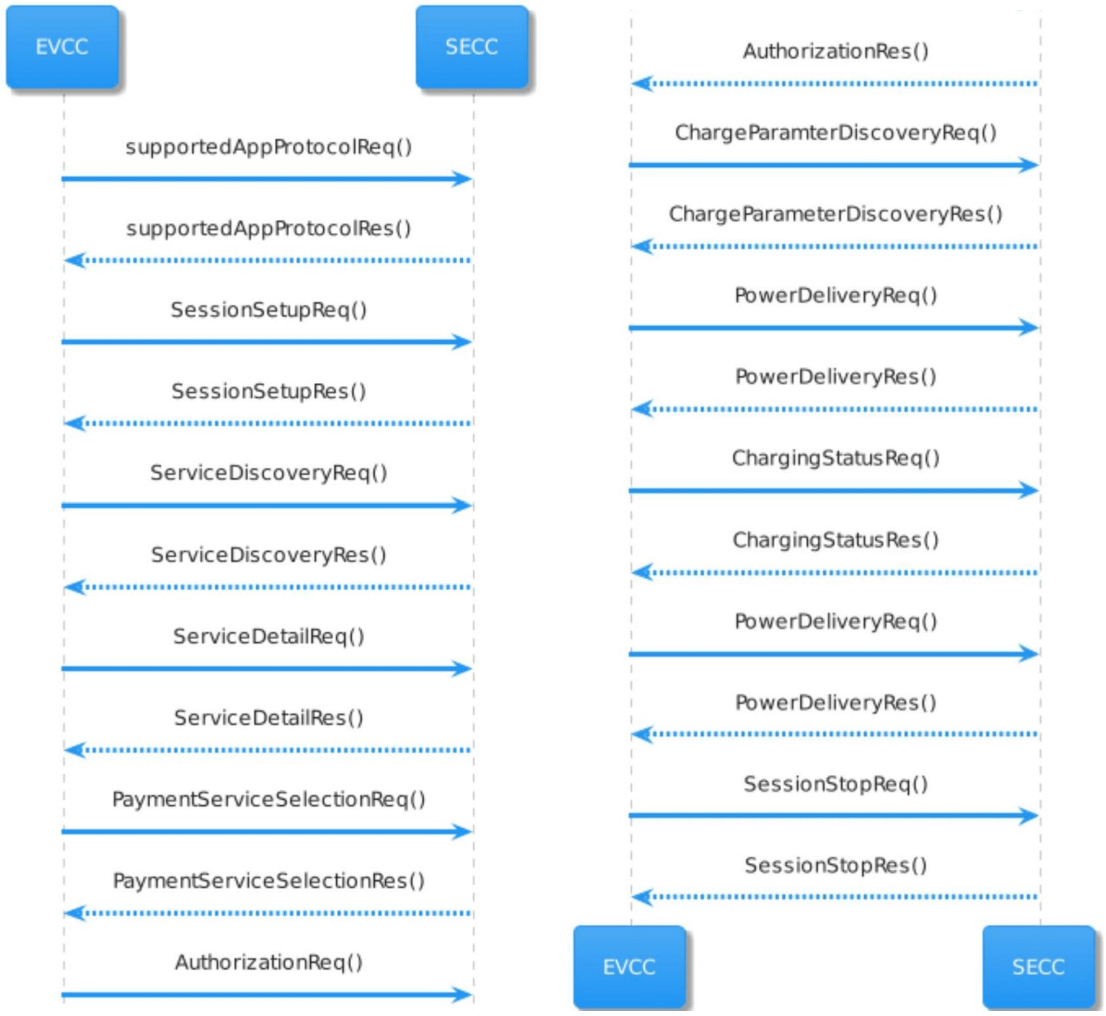


Figure 2.4: AC sequence messages with EIM [14]

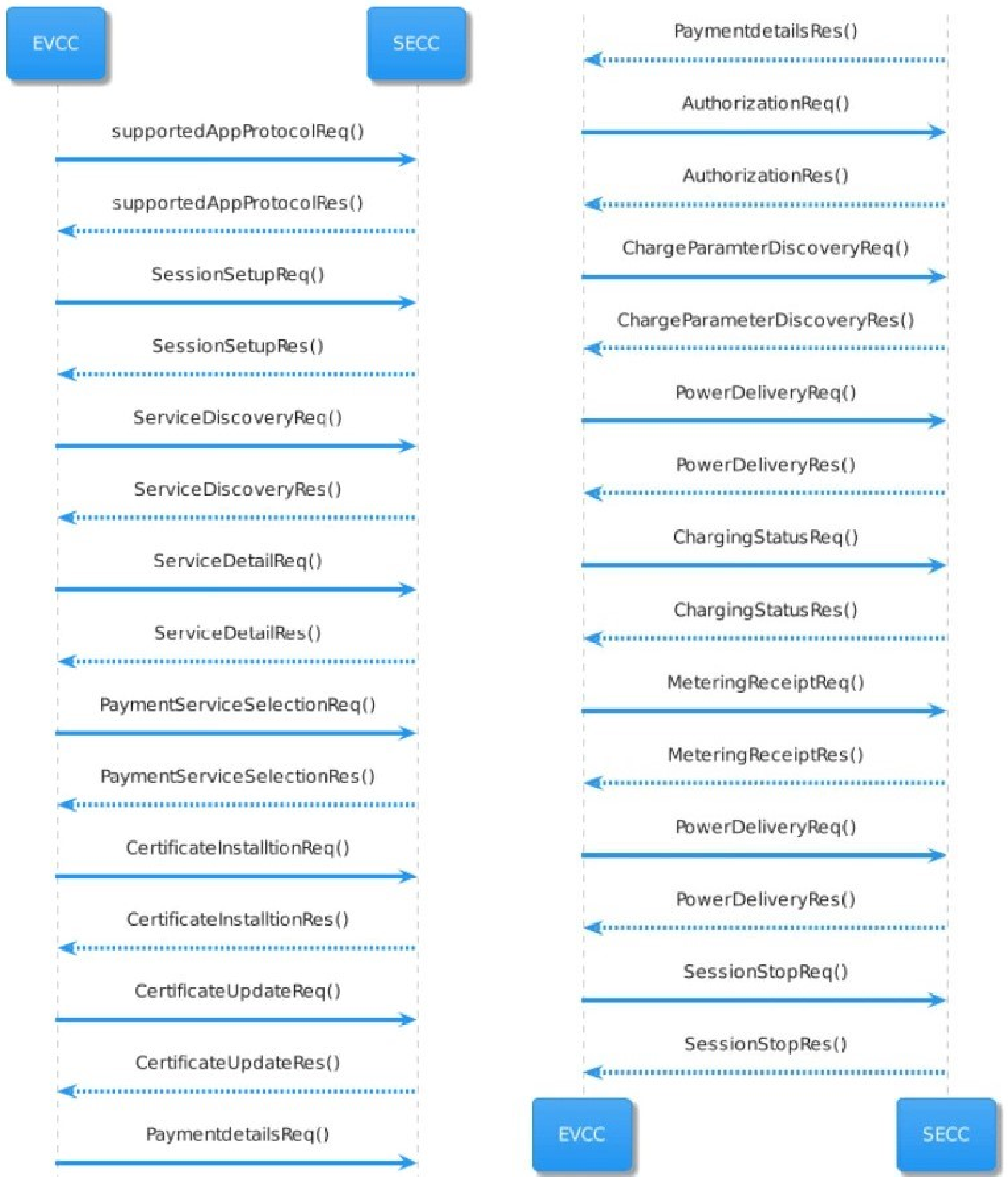


Figure 2.5: AC sequence messages with PnC [14]

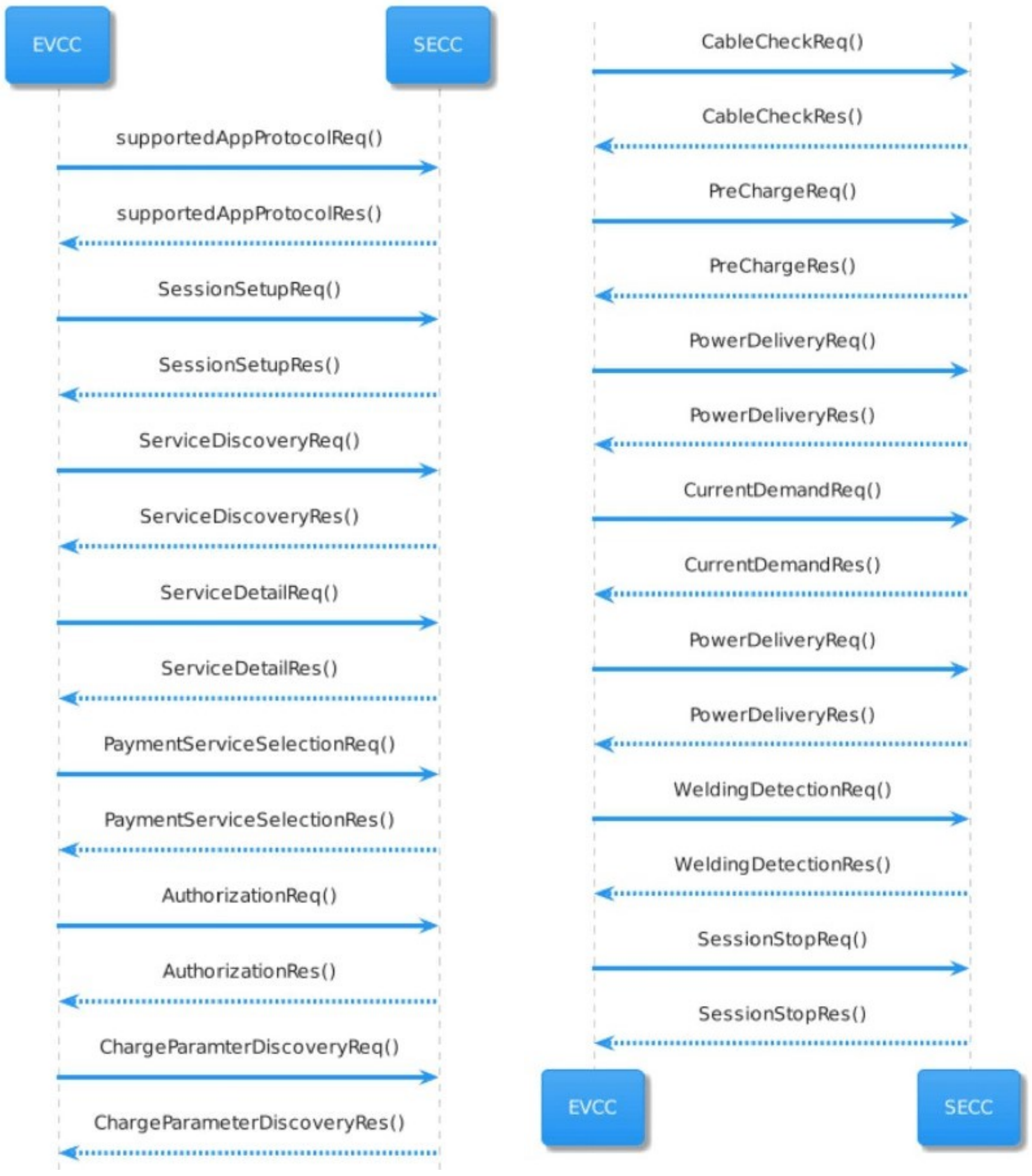


Figure 2.6: DC sequence messages with EIM [14]

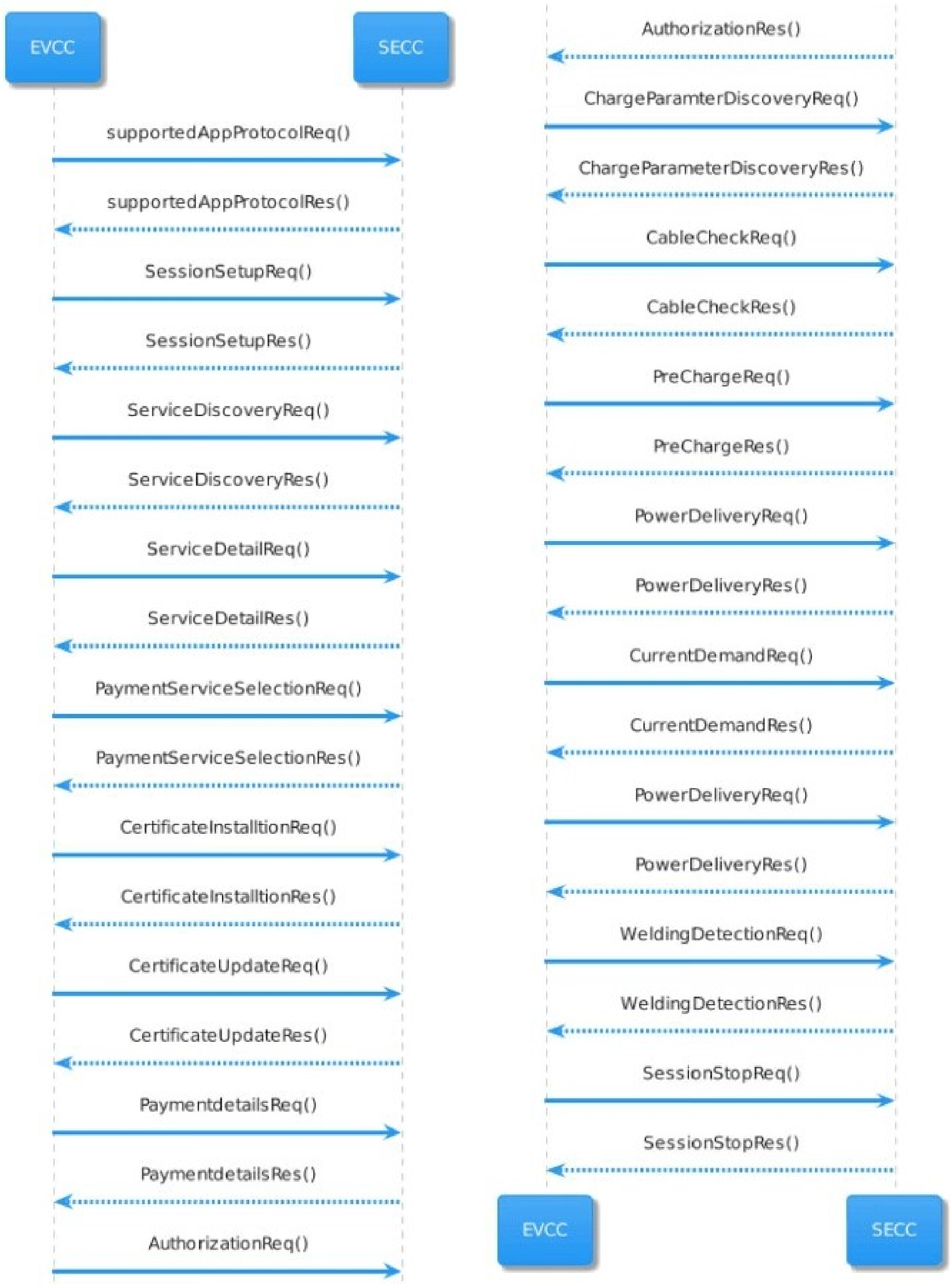


Figure 2.7: DC sequence messages with PnC [14]

- **supportedAppProtocolReq/Res**: First of all, the EVCC always sends a supportedAppProtocolReq message in order to start the communication setup. With this message, the EVCC can share with the SECC up to twenty protocols supported, each of these with its priority of preference. At this point, the SECC respond with a supportedAppProtocolRes message communicating the protocol that will be used during the charging process. To choose the protocol, the SECC selects from its list of supported protocols, the one with the highest priority indicated by the EVCC. If there is no protocol in common between the parts, a fail message is sent to the EVCC.

- **sessionSetupReq/Res**: To complete the communication setup, the EVCC sends a sessionSetupReq message and the SECC responds with a session-SetupRes message. After this pair of messages, a new charging session is created.

- **ServiceDiscoveryReq/Res**: After that, a sequence of messages about the identification, authentication and authorization can start. Every messages session start this part of the communication with a ServiceDiscoveryReq message sent by the EVCC. With this message, the EVCC requests to the SECC all the services provided by the charging point. The EVCC, with the body of the ServiceDiscoveryReq can filter the services by its scope or category. With the ServiceDiscoveryRes, the SECC sends to the EVCC the services available, together with the payment options supported by the charging point (EIM or Plug and Charge).

- **ServiceDetailReq/Res**: Then, the EVCC sends the ServiceDetailReq message to request the CP additional details about the services provided in the previous message. With the ServiceDetailRes message, the SECC sends the additional information requested.

- **PaymentServiceSelectionReq/Res**: Now, the EVCC sends the PaymentServiceSelectionReq, indicating the chosen payment methods, for the services that the charging point is going to provide. The SECC responds with a PaymentServiceSelectionReq message, confirming or not the chosen payment methods.

If the Plug & Charge method is chosen, three pair of messages can be exchanged between the parts: **CertificateInstallationReq/Res**, **CertificateUpdateReq/Res** and **PaymentDetailsReq/Res**.

- **CertificateInstallationReq/Res**: The CertificateInstallationReq message is sent by the EVCC if its contract certificate is expired or if it does not have one. Within this request, the EVCC sends the OEM provisioning certificate, already installed in the EV by its car manufacturer. This certificate is needed to authenticate the EV the first time and request the certificate from the SA. The SECC responds by sending the certificate requested by the EVCC, within the encrypted private key required to sign the certificate and the certificate chain.

- **CertificateUpdateReq/Res**: If the EV certificate is still valid, but it will expire soon, a CertificateUpdateReq message is sent by the EVCC. Within this message, the EVCC shall send the Contract Signature Certificate Chain, which is fundamental to verify the validity of the certificate. At this point, the SECC connects with the SA and asks for the certificate requested by the EVCC. With the CertificateUpdateRes message, the SECC provides the new contract signature certificate chain, that will be used as the private key for the new-installed certificate.

- **PaymentDetailsReq/Res**: With the PaymentDetailsReq/Res pair of message, the EVCC and the SECC check if the payment details provided by the EVCC are correct. Plus, the SECC sends a challenge to the EVCC, which has to be correctly signed by the EVCC, in order to confirm the validity of the certificate.

- **AuthorizationReq/Res**: After the three above messages pair, all the communication sequences continue with the AuthorizationReq/Res message pair. If the payment method chosen is PnC, the EVCC responds to the challenge previously sent by the SECC. Otherwise, the AuthorizationReq message is empty. With the AuthorizationRes message, the SECC checks if the challenge has been signed correctly. After this message, the EVCC is allowed to charge at that specific charging point.

- **ChargeParameterDiscoveryReq/Res**: By sending the ChargeParameterDiscoveryReq message, the EVCC starts the negotiation of the charging parameters with the SECC. EVCC's charging needs are sent within this message, such as the energy transfer mode (AC/DC), when the user will need to use the EV, and other parameters like maximum voltage, current, and power of the charging process. Receiving this information, the SECC shall ensure to satisfy the user needs, concurrently with the smart grid needs. Then it sends the ChargeParameterDiscoveryRes message indicating the maximum voltage, current, and power that the charging point can provide, together with a sales tariff table indicating the costs of the charging during the charging time.

Subsequently, if the AC transfer mode is selected, the message sequence continues with the **PowerDeliveryReq/Res** message pair. Instead, if it is the DC transfer mode case, before to send PowerDeliveryReq/Res, two other message pairs are eschanged: **CableCheckReq/Res** and **PreChargeReq/Res**.

- **CableCheckReq/Res**: With the CableCheckReq message, the EVCC informs the SECC if the cable is locked and if the EV is ready to charge. Plus, the EVCC also communicate the state of charge of the EV battery. The SECC responds with a CableCheckRes message informing the EVCC if it is ready to start the charge.
- **PreChargeReq/Res**: By sending the PreChargeReq message, the EV indicates to the SECC the target voltage and the target current. The SECC responds indicating the EVSE status and the present voltage.

Then, both AC and DC charging modes continue the communication with the **PowerDeliveryReq/Res** message pair.

- **PowerDeliveryReq/Res**: By sending a PowerDeliveryReq message, the EVCC finally requests the SECC to start the charging process. At this point in time, the energy starts to flow from the EVSE towards the EV. The SECC responds with a PowerDeliveryRes message, confimring the charging profile chosen by the EVCC and sending information about the EVSE status.

During the charging process, if we are in AC charging mode, the **ChargingStatusReq/Res** message pair is continuosly exchanged between the parts. If it is the

DC case, the **CurrentDemandReq/Res** message pair is exchanged during the whole charging process.

- **ChargingStatusReq/Res**: During the AC charging process, the EVCC continuosly sends an empty ChargingStatusReq message to the SECC, that responds with a ChargingStatusRes message. Within this message, the SECC gives information about the maximum current provided by the EVSE, the EVSE status and the current meter readings. Plus, the SECC tells the EVCC if it has to request a receipt for billing purposes: if this is the case, the EVCC must send a **MeteringReceiptReq** message, confirming to have received the metering data from the SECC. The SECC responds with a **MeteringReceiptRes** message, accepting or not the receipt previously sent by the EVCC.

- **CurrentDemandReq/Res**: During the charging process, the EVCC continuosly sends a CurrentDemandReq message containing different information such as the target current, the target voltage, the ramining time to fully charge the battery pack, the maximum power, current and voltage allowed by the EV. The SECC responds with a CurrentDemandRes message containing the EVSE status, the present voltage, the present current, the maximum power, current and voltage the EVSE can provide, the metering info and if a receipt is required.

To terminate the charging session, the EVCC shall send a PowerDeliveryReq message, informing the SECC of its intention to stop the charging. During the charging process, the user or the EVCC can also request a renegotiation, always with a PowerDeliveryReq message, in order to renegotiate the charging profile and continue with the charging process (for example, the user must leave before the already communicated departure time). The charging session could also be paused, letting it resume after a certain amount of time with the same charging profile.

Next, in case of DC charging mode, the EVCC sends a **WeldingDetectionReq** message requesting the EVSE to check the welding at its side. The SECC responds sending a **WeldingDetectionRes** message, containing the information requested and also the EVSE status.

Finally, the communication terminates with the **SessionStopReq/Res** message pair, both for AC and DC charging modes. The EVCC simply sends a Session-StopReq message requesting the SECC to terminate the charging process. The SECC responds with a SessionStopRes message, indicating if the termination was successful.

2.4 Public Key Infrastructure

Security is one of the main ISO 15118 goals. It is absolutely required, since a lot of information are exchanged between the parts during the charging process.
The three pillars of ISO 15118 standard are the following:

- **Confidentiality**: This means that the content of the message will be read only by the intended receivers.
- **Data integrity**: This service assures to avoid or at least detect unauthorized change of the sent message.
- **Authentication**: This means that the parties involved are able to identify themselves.

To provide these services, the **Elliptic Curve Cryptography (ECC)** method is implemented: it is an asymmetric crypto system where the parts involved have a pair of key, a public one and a private one. The public key is known to everyone, the private is only known by the component who owns it. If someone wants to send a message to another component, it must encrypt it with the public key of the receiver, which will be able to decrypt it with the private key. The private key and the public key are related one to each other based on a mathematical principle. The use of public and private keys requires a **Public Key Infrastructure**.

Creating, storing, distributing, and revoking digital certificates used to prove that a specific public key belongs to a specific person or thing are all done through public key infrastructure. According to [15], the use of PKI needs a clear definition of the ecosystem and the roles that every element must accomplish.
The most important role in a PKI is the Certificate Authority (CA). It is the party that every PKI element can trust to release certificates. Using its private key, the CA signs the certificates. With the public key released by the CA, every party can verify that a specific contract has been signed with the CA's private key (therefore

by the CA itself). The PKI structure defined in the standard ISO 15118-2 is based on a hierarchical structure, where the CA lies at the top realizing the so-called **root certificates**. Below the CA, the subordinated Certificate Authority (SubCA) can be found, realizing intermediate certificates. They can be useful to group the certificates, and sign them with their public key to issue them to the layer below. In fact, after the intermediate certificates, there are the **leaf certificates**, that belong to a private entity that only uses certificates provided by the layers above, therefore the private key of leaf certificates is not used to sign other certificates.

When the certificate validity needs to be checked (e.g. to start the charging process with PnC), three steps must be followed:

- Control that the certificate is valid at that precise moment therefore checks if the current time is between the certificate released time and the certificate expiring time.

- Verify that the certificate has been issued by a trusted party, checking the chain of the certificates, i.e. all the signatures from the roof (CA's and subCA's signature).

- Check if the certificate has been revoked. In fact, if a CA can no longer be trusted, every certificate issued by this CA must be revoked globally. For this reason, an online Certificate Revocation List (CRL) is available for the PKI parties that need to check the validity of the certificate.

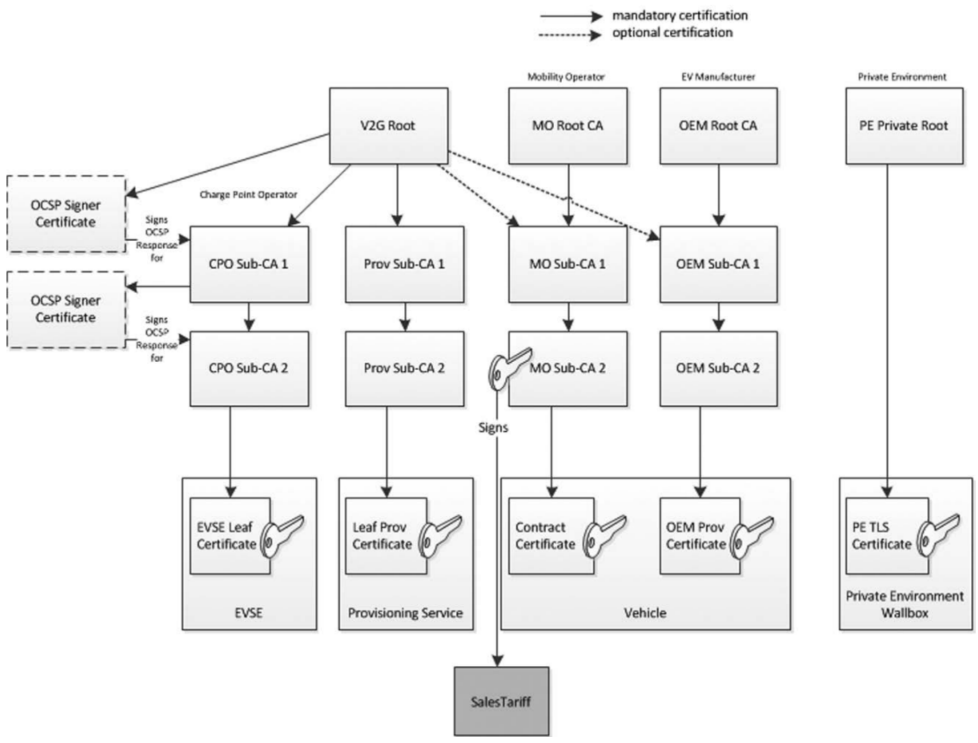


Figure 2.8: ISO 15118 certificate hierarchies [12]

As shown in 2.8, four different chains are possible in the ISO 15118 ecosystem. Here it follows a list of the parties that need to coexist in the ISO 15118 PKI:

- **V2G root operator**: is the highest trusted anchor in the ISO 15118 hierarchical structure and it issues certificates to all the ISO 15118 stakeholders.
- **Charge Point Operator (CPO)**: this role is accomplished by the companies that manage the charging stations and their related IT system to handle all the several payments (e.g. payment of offered services).
- **Mobility Operator (MO)**: companies that provide several charging related services such as handling the payment (RFID cards) or finding the charging point nearest to the user. Often referred as E-Mobility Service Provider (EMSP), they need to conclude a contract with the user in order to provide different services.
- **Original Equipment Manufacturer (OEM)**: it simply refers to vehicle manufacturers.

When the EV is connected for the first time to a charging point, the only EV available certificate is the one provided by the car manufacturer: the OEM provisioning certificate. Therefore, the OEM chain shall be known by the charging station in order to be able to authenticate the validity of the OEM provisioning certificate.
A contract certificate is needed by the user to access the EMSP services, therefore to communicate with it. The EV can not communicate directly with the MO, but it has to pass through the charging station and the Charging Station Operator (CSO). The Plug&Charge feature requires the presence and the validity of the contract certificate in the EV. This certificate is created and issued by the EMSP, for this purpose a different certificate chain is created. When the EV wants to install or update the contract certificate, it can do it before the starting of the charging process, by requesting it to the charging station. The CS forward this request to the CSO that forwards it to the EMSP. To install the certificate issued by the EMSP, the EV needs to use the OEM provisioning certificate public key already installed.

Plus, the charging stations need a charging station leaf's certificate in order to establish the TLS connection with the EV.

In the PKI there are several actors and customers that need to coexist, and since certificate handling is a critical and difficult task, the PKI can result very complex. As described in [15], ISO 15118 defines different proposals to reduce the complexity:

- **Certificate Provisioning Service (CPS)**: As already said, the EV needs to communicate with the EMSP. To do so, the EMSP identifies itself with a private key that can be derived either by the V2G root operator (the same as the EV) or by the EMSP itself that acts as CA. This means that the EV must store the CA certificate of every existing MO. To resolve this issue, the messages sent by the EMSP are not signed by the EMSP itself, but by the CPS that is derived from the V2G root operator.

- **Derive certificates from the same root CA**: If every CS has a certificate derived from a different CA, the EV shall store all these certificate chains. Therefore, all the CS certificates must be derived from the same root CA.

- **Limit the number of V2G root CAs in an electric vehicle**: Since the car manufacturer wants to limit the number of certificates to be installed on the EV, the V2G root certificates are limited by the ISO 15118 standard.

- **Limit the number of layers for (SUB)CAs to two**: Again, the EV manufacturers want to limit the certificate installed on the EV, but every EMSP and CSO would like to sign their own certificates and therefore act as a SUBCA. ISO 15118 defines the compromise at two layers maximum.
- **Limit the number of contract certificates in an electric vehicle to one**: The EV will need to have just a contract certificate to access all the services offered by the ISO 15118 ecosystem.

Chapter 3

RiseV2G

3.1 Protocol stack solutions

After understanding how an electric vehicle works, how it can be charged, and what the ISO 15118 standard is able to define, the following step of the thesis has been to find an open source protocol stack implementing the ISO 15118-2 requirements. Several messages and functions are required to create a charging process communication compliant with the ISO 15118 standard requirements. For this reason, two main open source protocol stack has been developed: OpenV2G and RiseV2G.
The first one focuses more on the PKI world, therefore simulating the communication between all the PKI parties and the certificate handling. All the hierarchical structure elements are simulated, such as the EV, the CS, the CSO, the EMSP, etc..
Since this thesis is focused on the application layer messages defined by the ISO 15118 standard, the RiseV2G has been the best possible option to implement. It is the only fully-featured ISO 15118-2 implementation, it is open-source and it is implemented in JAVA. In the next paragraph, a more-in-deep explanation is given. During the thesis project, the new document ISO 15118-20 has been released, together with a new open-source protocol stack called Josev. The scope of Josev is to stand as the reference of the ISO 15118-20 standard, which adds new features to the ISO 15118-2 standard. Anyway, the market implementation of ISO 15118-2 standard is a lot closer than the implementation of ISO 15118-20, which seems still

distant to be practically implemented in the public market. Also, RiseV2G has been published in 2015, therefore it has already been used worldwide and during these years it is been updated in order to solve different issues. For these reasons, I decided to work on the most asserted RiseV2G protocol stack, and therefore ISO 15118-2 standard.

3.2 The structure of RiseV2G

RISE V2G stands for "Reference Implementation Supporting the Evolution of the Vehicle-to-Grid" communication interface ISO 15118. In fact, all the messages defined in the ISO 15118 standard are implemented by this open-source software. It has been released on GitHub and used worldwide since then by companies that want to enter the ISO 15118 ecosystem. It is been published under an MIT license. Its scope is to convert the ISO 15118 communication requirements into a runnable program that can be used by the companies to build up their own market-ready software, both for the EV and the charging station. Indeed, the scope of the thesis project is to study in-depth RiseV2G, modify it according to the user choices and implement it to create a communication channel between the EV and the EVSE.

As already covered, the digital communication is between the EVCC and the SECC, which are the communication controllers of the electric vehicle and the charging station.
It is completely implemented in Java and five different folders are present:

- **RISE-V2G-SECC**: In this folder, there is the code regarding the charging station communication controller.
- **RISE-V2G-EVCC**: As per the SECC folder, RISE-V2G-EVCC holds the code for the EV communication controller.
- **RISE-V2G-Shared**: In this folder, everything that is used by both the EVCC and the SECC is present, in order to avoid error and code repetition. In particular, in this folder is present the codec used to encode and decode the messages using the EXI codec. As already discussed, the messages are encoded to result smaller in size and faster to process. Plus, some functions about security are contained in this folder.

- **RISE-V2G-PARENT**: This folder mainly contains the *pom.xml* configuration file, where Project Object Model (POM) is an XML representation of a Maven project.
  In this file, different information are contained, such as:
  - **Project name**
  - **Project version**
  - **Information about the project creator**
  - **Other build files explaining how to create a runnable project**

  RiseV2G makes use of a Maven project. Maven is used to build the project, automating the process and creating, for example, a JAR file for the EV and another one for the charging point. Other than automating and simplifying the build process, Maven provides other useful features, such as: taking care of all the project dependencies, perfectly integrating with the IDE Eclipse, providing quality information about the project, and some useful guidelines to follow for best practice development. In the RiseV2G case, the Maven project generates four different projects, one for each folders (except for the certificates one): RISE-V2G-SECC, RISE-V2G-EVCC, RISE-V2G-Shared, RISE-V2G-PARENT. Maven also allows us to see the dependencies of each project, e.g. the RISE-V2G-SECC and the RISE-V2G-EVCC depend on RISE-V2G-Shared. The source code for each project is located at the directory *src/main/java*, while in the *target* folder the created executable file (JAR file) is kept.

- **RISE-V2G-Certificates**: This folder, as the name may suggest, contains the code regarding the generation of certificates. As a demo, it is not our scope to integrate the project into the PKI, therefore static certificates created locally can be used. In this folder, several shell scripts are present to create the digital certificates. The certificates are created with OpenSSL, a complex and open cryptography library. Some variables are present to modify the certificate created, such as the period of validity.

Both the RISE-V2G-SECC and the RISE-V2G-EVCC contain a *.properties file*: this file extension is generally used in java applications to set the properties

of the java project. In RiseV2G, two property files exist, one for the SECC (SECCConfig.properties) and one for the EVCC (EVCCConfig.priorities). These two files are very important for our application since most of the important parameters related to the charging communication are set here.
First of all, in both the priorities files is set the network interface on which the EVCC-SECC communication can be established. It can be a wifi port, the ethernet cable or it is even possible to create a communication loop, running both the EVCC and the SECC on the same computer. For RiseV2G to run, network interfaces have to provide an IP version 6 address. Another priority set in the SECCConfig.priorities file is the one about the transfer mode supported by the charging stations, picked from the following list defined in [14]:

- AC three phase core
- AC single phase core
- DC core
- DC extended
- DC combo core

In SECCConfig.priorities can also be set if the charging process is a free service, if it is located in a private or public environment, and a lot of information about the debugging.
Plus, the authentication modes supported by the charging sessions can be set: it can be Contract for Plug&Charge, ExternalPayment for EIM (RFID card or QR code), or can be both.

In the EVCCConfig.priorities, instead, it can be set to use TLS communication or just TCP. TLS communication is required for the Plug&Charge feature but optional if the payment method chosen is the EIM.
Another important parameter is the requirement of the certificate update from the EV: it is expressed in the number of days, so if the value is 20, it means that twenty days before the certificate is supposed to expire, the EV requires the EVSE a certificate update.
The energy transfer mode requested by the EV is defined in this file, and it must be contained in the SECCConfig.priorities transfer mode supported list.

These properties files are crucial since many fundamental parameters are set here. They are frequently opened by the main to take the parameters values. For this reason, I decided to insert here all the configurable parameters of the charging session, in order to modify it from my own program. This is better explained in the next chapter, where an in-depth explanation of the structure of the demo is given.

3.2.1 RISE-V2G-SECC

The RISE-V2G-SECC project is mainly composed by the following parts:

- **Main**: In the *Main* folder of the RISE-V2G-SECC project, just the *SartSECC* java project is present. This file allows us to start the SECC. As already said many times, ISO 15118 is a client/server communication where the SECC acts as the server. For this reason, the SECC must already be active when the EVCC program is launched. First of all, the values stored in the SECCConfig.priorities file are loaded into the program, so that they can be accessed at any time during the process.
  Then, a UDP, a TCP, and a TLS servers will be launched, waiting for the EVCC to connect. The UDP server is used at the beginning of the communication setup, where the EVCC sends messages requests to the SECC asking for the network interface address and the port of the SECC. Plus, the UDP server is used by the EVCC to communicate if a TLS or TCP server-based communication will be used. All these messages are exchanged according to the SECC Discovery Protocol (SDP) specified by the ISO 15118-2 standard. Before to launch the transport layer threads, the SECC session handler is initialized. This is to avoid possible race problems, where the EVCC asks to start a communication setup through the UDP server, but the communication session is still not initialized.

- **Communication session**: It is composed of two different java files:

  - **V2GCommunicationSessionHandlerSECC**: This project creates and manage the communication sessions, called *V2GCommunicationSession SECC*. If the EV wants to start a charging session, a new connection handler object is created. Instead, if the charging session was paused, and the EV simply wants to restore it, the connection handler is resumed because it was been kept alive, but the TCP/TLS communication channel needs to be opened again. In this file, if the EV requests a charging pause, the communication channel is closed but the charging session data is memorized.

  - **V2GCommunicationSessionSECC**: This is the java project that handles and manages the communication. Its scope is to define which controller implementation to use (if the AC or the DC) and to create the back-end interface. It has to handle the notifications arriving from the communication and act correspondingly. The payment option defined in the properties file is stored here so that the EV wants to know which payment option is supported. This java file is crucial because it processes the incoming message from the EV, and it sends the V2G message from the SECC to the EVCC. The functions defined in this file are used by all the states and controllers that will be explained later. Indeed, it is used by the states as a shared file where important settings defined during the charging process can be found, such as the energy transfer mode selected, the status of the EVSE, the contract certificate chain, the status of the charging process, and so on. In conclusion, this communication handler is where the session is initialized and its crucial properties are stored.

- **Backend interface**: This file is crucial for the certificate's use. Normally, an OCPP communication would be used between the EVSE and the EMSP. In RiseV2G, this is simulated by the *BackendInterface.java* project. Given the OEM provisioning certificate, this project simulates the certificate chain that the SECC has to check to authenticate the EV. This file interacts with the certificates and keys generated with the shell files contained in the RISE-V2G-Certificates folder. Simulating the interaction of the charging station with the PKI, also the communication with the secondary actor (SA) is simulated here. A list of the schedules provided by the SA is created and provided to the EV. Based on the departure time provided by the EVSE during the charging communication, this java project creates a sales tariff, indicating the energy price against the time. Three different schedules are provided to the EV that needs to choose the preferred one. This Sale tariff needs to be signed by the SA if Plug&Charge has been selected. During the charging time, the sale tariff can be composed of a maximum of three different price levels, and it can be expressed in absolute value or percentage to the maximum price. In RiseV2G, this sale tariff is generated once, therefore it is always the same. In my demo, as explained later in this document, I generate the sale tariff based on the values of the time that the EV is connected to the charging session: there are times during the day when the demand peak is high, so the price is increased, meanwhile other periods during the days where the price is lower because there is no high energy request. All the different sales tariffs are grouped in a list that is sent to the EVCC, so the user is able to choose which one he/she prefers.

- **AC/DC controller**: In the AC and DC controllers file are defined the charging parameters and the functions that need to be called by other project to access these values.
  In the ACEVSE controller, the parameters defined are the following:
  - EVSE nominal voltage;
  - EVSE maximum current;
  - EVSE status;
  - Metering information.

  In the DCEVSE controller, the parameters defined are the following:
  - EVSE notification: parameter used by the EVSE to send a notification to the EV, for example to inform the user about the changing of the charging schedule.
  - EVSE maximum voltage;
  - EVSE minimum voltage;
  - EVSE maximum current;
  - EVSE minimum current;
  - EVSE maximum power;
  - Metering information;
  - EVSE peak current ripple;

  - Other parameters defining if the power/voltage/current limit has been reached.

  All the parameters set in these two controller files are then called and processed by the corresponding states and sent to the EVCC in the correct response messages, as defined in ISO 15118-2.

- **States**: During the communication with the EVCC, the SECC constantly waits for a request message and then responds with a response message. For this purpose, for every request message exists a *WaitForMessageReq* SECC state. The *ServerState* java project contained in this folder manage the state transmission of the SECC, analyzing the header and the body of the V2G messages and deciding which state has to come next. Also, error messages are handled for debug purposes.

  The SECC's states are the following:

  - **WaitForSupportedAppProtocolReq**: In this state, the SECC reacts to the first request message coming from the EVCC: SupportedAppProtocolReq. It checks if the protocols listed by the EVCC are supported by the SECC and finds the one with the highest priority. Then, send the SupportedAppProtocolRes message indicating which protocol has been chosen and if the negotiation has been successful.
  - **WaitForSessionSetupReq**: It waits for the SessionSetupReq message and response sending the EVSE ID, useful to set up the communication. Sending the SessionSetupRes, the communication setup is concluded and the authentication can begin.
  - **WaitForServiceDiscoveryReq**: By processing this request message, the SECC has to respond by indicating which services can be provided to the user. It first controls if the request message has filters regarding the category or the scope of services required. A ServiceDiscoveryRes message is created to send the list of services provided. Plus, in this state, the energy transfer modes supported by the EVSE are set, together with the name and the scope of the related service. All the value-added services provided by the EVSE are listed here, such as internet access or the service enabling the exchange of information related to the EVSE.

For each service, the name and the category are specified, together with the serviceID: this is a crucial element regarding the services and strict rules are applied, as defined by ISO 15118-2. Indeed, ServiceID must be a unsignedshort variable type: the 0 ID, together with the IDs from 5 to 60000 are reserved by ISO/IEC. Service ID 1 is for the charging services, Service ID 2 is for the services related to the installation or update of the certificates, Service ID 3 is for internet access and Service ID 4 is for the exchange of EVSE-specific information. The Service IDs from 60001 to 65535 can be used for implementation-specific use.

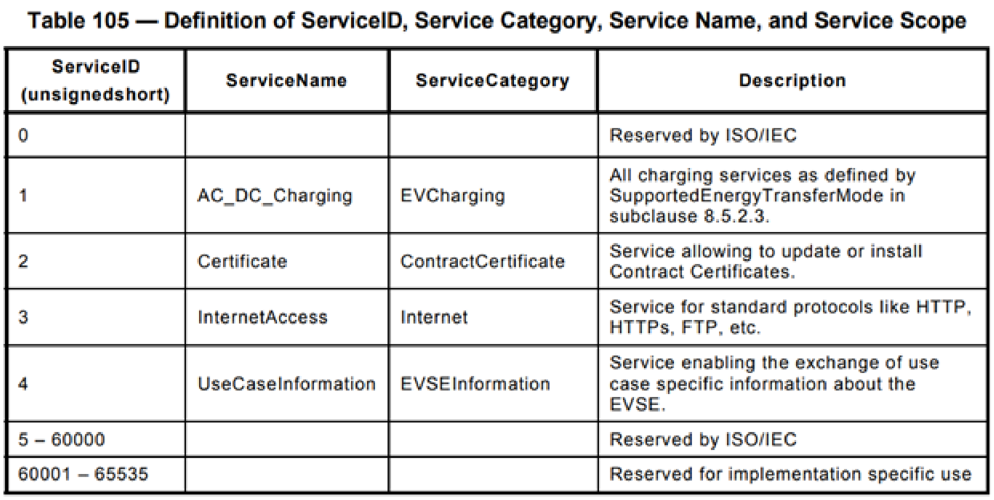


Table 105 — Definition of ServiceID, Service Category, Service Name, and Service Scope

ServiceID (unsignedshort)	ServiceName	ServiceCategory	Description
0			Reserved by ISO/IEC
1	AC_DC_Charging	EVCharging	All charging services as defined by SupportedEnergyTransferMode in subclause 8.5.2.3.
2	Certificate	ContractCertificate	Service allowing to update or install Contract Certificates.
3	InternetAccess	Internet	Service for standard protocols like HTTP, HTTPs, FTP, etc.
4	UseCaseInformation	EVSEInformation	Service enabling the exchange of use case specific information about the EVSE.
5 – 60000			Reserved by ISO/IEC
60001 – 65535			Reserved for implementation specific use

Figure 3.1: Definition of Services [14]

- **WaitForServiceDetailReq**: If a ServiceDetailReq has been sent by the EVCC, the SECC responds by indicating detailed parameters.
- **WaitForPaymentServiceSelectionReq**: In this state, the payment method selected by the EVCC is analyzed and processed. If the payment method selected is Plug&Charge, the check of the validity of the certificates is started. If the certificate is expired or is about to expire, is indicated to the EVCC to send a CertificateInstallationReq or CertificateUpdateReq message. Plus, in this state is again checked if the payment option selected is supported by the EVSE and if the selected service can be provided.
- **WaitForCertificateInstallationReq**: If the EV's certificate is expired, a new contract certificate is requested by the EVCC. In this state, the

SECC encrypts the private key and set the new certificate chain signature, defining the ID, the encrypted private key, and the EMAID. Also, the XML reference elements are set in this state.

- **WaitForCertificateUpdateReq**: If the EV's certificate is about to expire and if the remaining time is lower than the threshold chosen, a CertificateUpdateReq is sent by the EVCC. The SECC, in this java project, generates the new certificate and updates it. Also, a check of the contract certificate chain is executed and eventual errors are handled.
- **WaitForPaymentDetailsReq**: If the certificate payment method has been chosen, the EVCC sends a PaymentDetailsReq message. The SECC processes it in this state and prepares to send the PaymentDetailsRes message. The EV contract certificate chain is validated and saved. The validity period of the certificate is checked again, and if it is lower than the threshold, in my case twenty-one days, the certificate update must be executed.
- **WaitForAuthorizationReq**: In this state, the SECC decides if the authorization has been concluded with a positive result by reading the signature inserted in the header of the AuthorizationReq message sent by the EVCC. If the TLS connection has been chosen together with the contract payment method, the signature is verified.
- **WaitForChargeParameterDiscoveryReq**: When the ChargeParameterDiscoveryReq is sent by the EVCC, the SECC creates a new schedule if the charging process is not already started. Based on the tariff table defined in the backend java file, the schedules are created. The sales tariff is signed here, to simulate the real backend functionality, using the *security utils* class contained in the security folder. The energy transfer mode is checked again in this java file. At this stage, the parameters given by the EV regarding the charging process are checked: if a mandatory parameter has not been sent, an error must be shown to the user; also if one of the parameters exceed the EVSE limits, an error must be prompted. This is done for both the DC and the AC charging process.
- **WaitForCableCheckReq**: The SECC enters this state only if the DC charging mode has been selected. A CableCheckRes message is sent

indicating the EVSE controller status taken from the DCEVSE controller.

- **WaitForPreChargeReq**: It responds to the EVCC indicating the target voltage and target current from the DCEVSE controller. Also the EVSE status and the present voltage are taken from the DCEVSE class type and sent to the EVCC.
- **WaitForPowerDeliveryReq**: In this state the SECC reacts to the PowerDeliveryReq incoming message: if a start of the charge process has been requested the SECC start the charging process, instead, if a stop of the charge process is requested, the SECC stop the charging process. If the EVCC asks for a renegotiation, the SECC checks if there is a charging process ongoing; if this is the case, the SECC responds with a PowerDeliveryRes message indicating the possibility to renegotiate the charging schedule. In this state, the charging profile is checked, and the response message is prepared in order to send the EVSE status for the AC or DC controller, depending on te charging mode selected. The PowerDeliveryReq message coming from the EVCC is crucial to start, stop or renegotiate the charging process.
- **WaitForCurrentDemandReq**: In DC charging, during the charging loop, the EVCC continuosly sends a CurrentDemandReq message requesting the SECC to send the charging parameters status. Indeed, in this state all the requested parameters are set and send to the EVCC through a CurrentDemandRes message. Plus, the SECC indicates the EVCC to send a MeteringReceiptReq message if Plug&Charge feature has been selected. In case of EIM, the metering receipt is not involved in the charging communication.
- **WaitForChargingStatusReq**: In AC charging, during the charging loop, the EVCC continuously requests the charging status to the SECC. In this state, the EVSE state is set and sent to the EVCC.
- **WaitForWeldingDetectionReq**: A WeldingDetectionRes is created and sent containing the EVSE status and the present voltage.
- **WaitForMeteringReceiptReq**: In the case of PnC, the EVCC must send a MeteringReceiptReq message, sending the metering values. If these values match the ones sent by the SECC, the charging process can end, if

the values do not match, an error is generated.

  - **WaitForSessionStopReq**: if the EV has sent a request to terminate the charging session, the communication session is terminated and the last SessionStopReq message is sent. Instead, if the EV has asked for a pause of the charging session, the communication is set to start again from the SupportedAppProtocolReq message.

- **Transport layer**: Inside this folder are defined the servers used during the communication: TLS, TCP and UDP. For each server a java file is created. These java files are called by the SECC's main to initialize the communication sockets. The local address of the servers is taken from the properties file, and the port number is generated randomly.

3.2.2 RISE-V2G-EVCC

The RISE-V2G-EVCC project is mainly composed of the following parts:

- **Main**: This Java class is simpler than the main class for the charging station. Only two actions are taken here: setting of the correct properties file and initiation of a ”session handler” for the EV communication controller. A session handler is a program that takes care of the message flow between the EVCC and the SECC. It makes sure that the EVCC is sending request messages in the right order and processes the incoming responses accordingly.

- **Communication session**: As for the charging station controller, inside the communication folder two different java files are present:

  - **V2GCommunicationSessionHandlerEVCC**: This project creates and manage the communication sessions, called *V2GCommunicationSession SECC*. Firsts, the TCP, TLS, and UDP clients are initialized if the seccDiscoveryRes has been successful. A function telling the TCP/TLS client to create a notification when a new V2G message arrives or when a timeout has occurred while waiting for the SECC response message to be created. In case of terminating or pausing the charging session, a function stopping the TLS client is created. All the supported protocols are listed in this state.

  - **V2GCommunicationSessionEVCC**: This is the java project that handles and manages the communication. In this file, V2G communication session is initialized and all the still unknown values are set to default. The incoming messages are processed and the correct state of the EVCC is called. The session ID of the communication session, together with the authentication mode and the energy transfer mode requested are memorized here. Plus, the timers to measure the time that the EVCC is waiting for the SECC response are initialized here. Every EVCC state has to call the V2GCommunicationSessionEVCC methods to access the communication properties: energy transfer mode requested, EVSE ID, selected payment option, if to request a renegotiation, charging profile, supported protocols, chosen protocol, if an old session has been chosen, SA schedule chosen, selected services, offered services, service details to be requested, certificate status, challenge sent by the SECC and the charging session.

- **EV controller**: This java project is crucial for the EV operations: all the actions taken by the EVCC are defined here and called from the states files during the charging process.
  One of these functions allows the EVCC to get the payment methods supported by the SECC in any state, so that can be shown to the user through a human-machine interface.
  The Energy transfer mode of the EV is defined here.
  Plus, the following parameters for the AC charging mode are defined:

  - **Departure time**: The number of seconds the user expects to leave the car at the charging station.
  - **Energy amount**: Energy amount needed for the charging of the battery pack, expressed in Wh.
  - **EV maximum voltage**: Maximum voltage the EV can accept from the EVSE.
  - **EV maximum current**: Maximum current the EV can accept from the EVSE.
  - **EV minimum current**: Minimum current the EV can accept from the EVSE.

Also for the DC charging mode, the parameters are defined as follows:

- **Departure time**;
- **Energy amount**;
- **EV maximum voltage**;
- **EV maximum current**.

The charging profile is defined in this java file. From the secondary actor services provided by the SECC, the wanted schedule is selected. Initially, always the same schedule was selected. As explained in the next chapter, in the demo I let the user decide which schedule to choose.
The EV controller project also defines the EV status: if it is ready to charge and the state of charge.
Other fundamental parameters such as the target voltage and the target current are defined here and then taken by the different states and sent to the SECC, according to ISO 15118-2 requirements.
Plus, if the charging is complete has to be defined here, together with the remaining time to fully charge the battery, or to charge 80% of the battery.
The last crucial parameters defined in this file are the following: the maximum power, the maximum voltage, and the maximum current limits for the EV.

Lastly, a very important function called *isCharginLoopActive* is defined here. With this function, a static charging flow is created: the charging process just lasts one hundred cycles, with a renegotiation at the fiftieth cycle. It is obviously set in this way just for testing purposes. I changed it, transforming the test into a simulated communication where the charging process is stopped only when the user asks to terminate the charging or when the EV battery is fully charged. All my changes applied to RiseV2G are explained in the next chapter, paragraph 4.6.

- **States**: During the communication with the SECC, the EVCC constantly sends a request message and then waits for a response message. For this purpose, for every response message coming from the SECC, exists a *Wait-ForMessageRes* EVCC state. The *ClientState* java project contained in this folder manage the state transmission of the EVCC, analyzing the header and the body of the V2G messages and deciding which state has to come next. Also, error messages are handled for debug purposes. Through this file, the states projects can access the properties defined in the controller project.

The EVCC's states are the following:

- **WaitForSupportedAppProtocolRes**: This state is entered by the EVCC just after sending the first communication V2G message: *SupportedAppProtocolReq*. The session ID is taken from the *EVCC.priorities* file and it is checked if a previous charging session has been paused or if a new charging session has to be started. Moreover, the protocols supported by the SECC are analyzed and it is checked if the communication setup can continue.
- **WaitForSessionSetupRes**: If the SECC has sent a *new session established* message, this means that no older charging sessions were started, therefore the EVCC assigns a new session ID to the just created charging process. Instead, if an old session is requested to be joined, the EVCC checks if the ID sent by the SECC belongs to an old session previously stopped. If the check gives positive results, the SelectedPaymentOption and the RequestedEnergyTransferMode are restored, as defined in ISO15118-2. Otherwise, an error is generated, indicating that there is no old session requested by the SECC.
  Then, the services are restricted by scope and category and requested to the SECC through a ServiceDiscoveryReq message.
- **WaitForServiceDiscoveryRes**: A list, containing all the services that need a *ServiceDeatilsReq* message, is created in this state java file. In this list, all the other services other than the charging one are inserted.
  Then, the energy transfer mode is checked to be supported by the SECC: if not, a message error is shown. Also, it checks that the TLS is selected if the Plug&Charge payment method has been selected.
  The need for a certificate installation or update is executed here, together with eventually value-added services selected by the user: if one of these checks is positive, the EVCC will send a ServiceDetailReq. Otherwise, the PaymentServiceSelectionReq message is sent.
- **WaitForServiceDetailRes**: This state is entered if a value-added service

or if a certificate installation/update has been requested by the EVCC in the previous state. Therefore, the details related to the requested service are analyzed. Then a PaymentServiceSelectionReq message is sent.

- **WaitForPaymentServiceSelectionRes**: If the payment method chosen is Plug&Charge, the certificate status is checked again: if an installation or update is needed, the EVCC sends a CertificateInstallationReq or a CertificateUpdateReq message. If the certificate is valid and it is not about to expire, a PaymentDetailsReq message is sent. The XML reference elements, together with the signing private key are defined here. Instead, if EIM was selected, the next message sent by the EVCC is AuthorizationReq.
- **WaitForCertificateInstallationRes**: This state is executed if a certificate installation was needed. The certificate provisioning service chain is completely checked. If everything is correct, the new certificate is stored, otherwise, an error is shown.
- **WaitForCertificateUpdateRes**: This state is executed if the certificate installation was about to expire. The certificate provisioning service chain is completely checked. If everything is correct, the new certificate is stored, otherwise, an error is shown.
- **WaitForPaymentDetailsRes**: This state is reached only if PnC has been selected. The challenge sent by the SECC is saved and processed. An AuthorizationReq message, containing the response to the challenge, is sent.
- **WaitForAuthorizationRes**: the AuthorizationRes message sent by the SECC is processed. The SECC must have sent the EVSE processing state: if this is absent, the charging communication is terminated and an error is prompted. if the EVSE processing status is set to finish, the EVCC gets ready to send a ChargeParameterDiscoveryReq message. Otherwise, if the EVSE is still processing the previous message, the EVCC waits for the SECC to send another AuthorizationReq message indicating it has finished processing. In the ChargeParameterDiscoveryReq message, the energy transfer mode has to be sent, together with all the DC parameters (maximum current, maximum power, maximum voltage,

departure time, state of charge, EV status) or the AC parameters (energy amount, maximum voltage, maximum current, and minimum current), depending on the charging mode selected.

- **WaitForChargeParameterDiscoveryRes**: First of all, it is checked if the EVSE is still processing the ChargeParameterDisoveryReq message or if it has finished (it is indicated in the body of the ChargeParameterDiscoveryRes message). If it is set to finish, the AC/DC charge parameters are stored. If the EVSE requested to stop the charging process, the EVCC terminates it.
  After that, the sales tariff is analyzed and stored, only if the TLS connection has been enabled. In the case of PnC, and if a SalesTariff is used by the secondary actor, the secondary actor shall sign the field SalesTariff of type SalesTariffType. In the case of EIM, the secondary actor MAY sign this field. If the EVCC treats the SalesTariff as invalid, it shall ignore the SalesTariff, i.e. the behavior of the EVCC shall be the same as if no SalesTariff was received. Furthermore, the EVCC MAY close the connection. It then may reopen the connection again. The SA schedules are then saved to be used later.
  If we are in DC charging, the next message is the CableCheckReq, and the related timer to verify that the SECC responds before a defined time is started. If we are in the AC case, the next message is PowerDeliveryReq.
- **WaitForCableCheckRes**: The CableCheckres message is processed, storing the EVSE status together with the target voltage and the target current. Next, the PreChargeReq message is sent.
- **WaitForPreChargeRes**: After the PreChargeRes message is received from the SECC, the present voltage and the EVSE status are updated inside the EVCC memory. If the present voltage is equal to the target voltage, the charging session can start and the EVCC sends a PowerDeliveryReq message. Otherwise, the EVCC keeps sending the PreChargeReq message until the present voltage is equal to the target voltage.
- **WaitForPowerDeliveryRes**: The SECC sends a PowerDeliveryRes indicating the EVSE status and accepting the parameters sent by the EVCC in the PowerDeliveryReq message. This message is sent to start the charging session to terminate it or renegotiate it. If we are in DC and

the charging process is set to START, the CurrentDemandReq message is sent. If the renegotiation is requested, always in DC, the EVCC has to go back to the WaitForCableCheckRes state. In DC, if the EVSE asks to terminate the charging session, the next state will be the WaitForWeldingDetectionRes. In AC, after the start response from the SECC, the charging loop can begin with a continuous ChargingStatusReq/res messages exchange. If the EVSE responds to terminate the charging session, the EVCC sends a MeteringreceiptReq message.

- **WaitForCurrentDemandRes**: In DC charging, during the charging loop, the EVCC continuously sends a CurrentDemandReq message requesting the SECC to send the status of the charging parameters. Indeed, in this state, the parameters are processed. Then, if the SECC has required a metering receipt, the EVCC proceeds to send a MeteringReceiptReq message. The Charging session status sent by the SECC is processed: it can be *renegotiate* or *stop*.
- **WaitForChargingStatusRes**: In AC charging, during the charging loop, the EVCC continuously sends a ChargingStatusReq message requesting the SECC to send the status of the charging parameters. Indeed, in this state, the parameters are processed. Then, if the SECC has required a metering receipt, the EVCC proceeds to send a MeteringReceiptReq message. The Charging session status sent by the SECC is processed: it can be *renegotiate* or *stop*. After this state, when the charging process terminates, the next message is again a PowerDeliveryReq.
- **WaitForWeldingDetectionRes**: In this state, the EVCC reacts to the WeldingDetectionRes containing the EVSE status and the present voltage.
- **WaitForMeteringReceiptRes**: In the case of PnC, the EVCC previously sent a MeteringReceiptReq message and the SECC responds with a MeteringReceiptRes message communicating if the metering information is valid or not.
- **WaitForSessionStopRes**: This is the last EVCC state, where the communication is closed and the EVCC client is closed. A message showing if the communication has ended successfully is shown.

• **Transport layer**: Inside this folder are defined the servers used during the communication: TLS, TCP, and UDP. For each server, a java file is created. The header of the V2G message coming from the SECC is analyzed here. If the payload of the header is too high, an error is prompted. Also if the SECC message arrives after the timeout, an error is shown. If an error occurred in the run method, the TCP client will be stopped by closing all streams and the socket and interrupting the thread. The local address of the servers is taken from the properties file, and the port number is generated randomly.

3.2.3 RISE-V2G-Shared

The shared project is where global values used by every project are defined. The RISE-V2G-Shared is mainly composed of the following parts:

- **Shared enumerations**: In this folder, all the V2G request/response messages are defined. Different PKI parts (OEM, MO, CPS, CPO) are defined for security purposes. The four different message sets are created (AC_CHARGING_EIM, AC_CHARGING_PnC, DC_CHARGING_EIM, DC_CHARGING_PnC) so that the projects can recognize which path they have to follow.

- **Shared EXI codec**: This folder is used to encode and decode the V2G messages using the *EXI codec* to reduce the size of the messages and speed up the process.

- **Message handling**: In this project, the status change is handled for both the EVCC and the SECC. Based on the information received from the states, the following state is launched. Also, the message is created, adding the header and the body to send. All these functions handling the received message or the to-be-sent message are called by the states or the controllers, to perform precise actions. The pause and termination of the charging session functions are also defined here.

- **Shared utils**: In this folder two crucial projects are present:

  - **Security utils**: In this project, all the functions related to security for both the EVCC and the SECC are defined and then called by other programs. The contract certificate status object is defined. A method

returning the certificate chain and the password is created and can be called by EVCC or SECC. These data are taken from a truststore file that is kept outside the JAR file: this is because the JAR file is read-only, while the certificates and the keys need to be modified.
A *verifyValidityPeriod* method is defined where the certificate validity is check with regards to date and time.
The *verifyDomainComponent* method is called to verify the PKI components defined in the certificate chain (OEM, CPS, CPO).
The *getValidityPeriod* returns how many days the certificate is still valid. If the certificate is not valid any more, a negative number will be returned according to the number of days the certificate is already expired.
The *verifyCertificateChain* method verifies the signature for each certificate present in the given certificate chain (leaf, SubCA2, SubCA1, root).
The *verifySignature* verifies if the certificate has been signed with the correct private key associated to the given public key.
Several methods handling the PKI structure are defined (e.g. getting the leaf certificate, getting the SubCA certificate, returning the public key of every certificate, etc.)
The *saveContractCertificateChain* is called by the WaitForCertificateInstallationRes and the WaitForCertificateInstallationRes states to store the new certificates provided by the SECC. Also the functions defining if the installation or the update of the certificate is needed are defined in this file.
One of the most important methods defined here is the *getCertificateChain* which is called by several states and is able to return the whole certificate chain, starting form the leaf certificates until the root certificate.
All the algorithms encrypting and decrypting the keys are defined here and can be called by every program present in RiseV2G.
The *getEMAID* returns the e-mobility account identifier from the contract certificate as part of the contract certificate chain.
The *verifySignature* verifies the signature given in the received header of an EVCC or SECC message.

– **Misc utils**: In this file, a method that determines the link-local IPv6 address, which is configured on the network interface provided in the

properties file, is defined.
The *getPropertyValue* method is called by the EVCC and SECC controllers to access the values specified in the properties files.

- **Messages definition**: In this folder, the instances of every request/response message are created. These are then called in the *Shared enumerations* folder.

Chapter 4

Proof of concept focused on ISO15118

An in-depth study of the ISO15118 standard and its reference implementation (RiseV2G) has been crucial for the realization of the Proof of Concept (PoC).
The demo is focused on the V1G (unidirectional power flow) communication between the electric vehicle and the charging station according to the ISO15118 requirements. With the use of two single-board computers, the charging station and the EV behaviors are simulated. The PoC allows the user to interact with the EV and with the charging station, in order to check and personalize the charging process. All the features introduced by the ISO15118-2 standard and implemented by the PoC are the following:

- **Plug&Charge**: The user can simply plug the cable into the EV and the charging process starts automatically. If the user still want to use an RFID card, is possible by scanning the QR code at the charging station.

- **Smart charging together with energy flexibility**: Different charging profiles are provided to the user, containing information about the price per hour, the percentage of energy coming from renewable sources and the CO_2 emission. The user can select the charging profile he prefers through the electric vehicle HMI.

- **Strong data security**: TLS protocol is used to exchange data between the electric vehicle and the charging station. It ensures authenticity, confidentiality and data-integrity.

The scope of the PoC is to demonstrate the future electric vehicle charging experience, with the main objective to improve the quality of the user journey during charging operations.

4.1 The structure of the demo

As previously explained, the demo is composed by two different simulated parts: the Electric Vehicle onboard charger and the Electric Vehicle Supply Equipment (EVSE).

The EVCC and the SECC are respectively part of the EV onboard charger and of the EVSE. Their scope is to handle the communication between the EV and the EVSE. They are implemented on two different single-board computers, the *Raspberry Pi 4 Model B*. Together with the two communication controllers, also the **Graphical User Interfaces (GUI)** are realized through the *Raspberry Pi Touch Display*.
The protocol stack solution, **RiseV2G**, is implemented to handle the communication between the two parts. Several changes have been made to the original RiseV2G in order to personalize the charging process to the user needs (handled with the GUI) and all the information related to the battery state of charge and the charging station energy available. For this purpose, two different **Power Flow Simulator (PFS)** have been created, one for the electric vehicle that simulates the battery state of charge before, during and after the charging process, and one for the charging station that simulates the energy available from the grid and the energy provided to the EV during the charging process.
All these programs (RiseV2G, GUI, PFS) are managed and processed by the **Business Logic (BL)** program. The BL is the only program to communicate with the others and its scope is to synchronize the operations. As already explained, RiseV2G is implemented in Java, while the programs that I created (GUI, PFS and BL) are implemented in Python.

In figure 4.1, a schematic representation of the demo is outlined. As shown, to be compliant with the ISO15118 standard, the EV and the charging station can only communicate through RiseV2G.

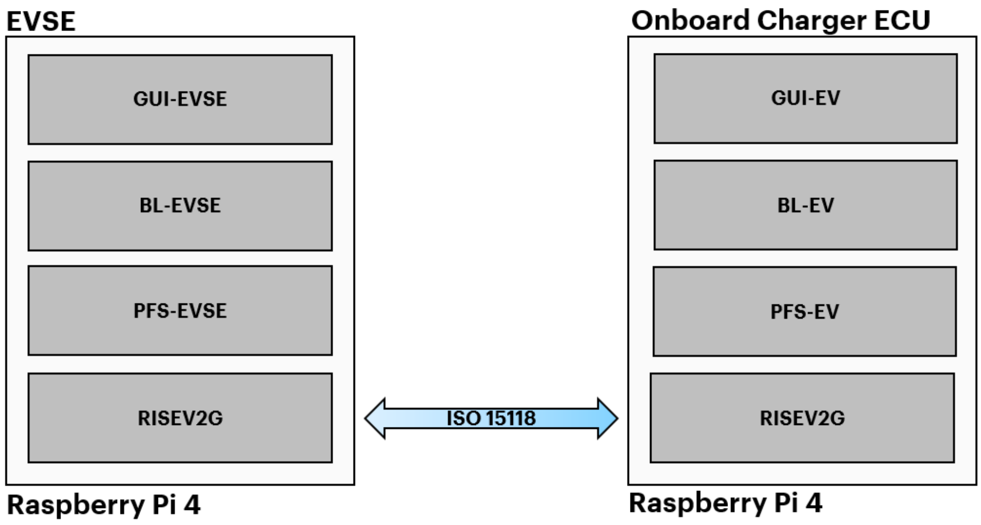


Figure 4.1: The structure of the PoC

A detailed overview of every element consisting the demo is provided in the following paragraphs.

4.2 Hardware components

Since the scope of the demo is to simulate the behaviors of the EVSE and the EV onboard charging, two communicating boards have been required. The boards must support both Java and Python programs because RiseV2G is implemented in Java and the other parts of the demo are realized in Python. The main difference between a Python project and a Java project is that the Java project requires a JVM (Java Virtual Machine).
In fact, in order to develop Java projects, three fundamental elements are required: JVM, Java Runtime Environment (JRE), and Java Development Kit (JDK). Without these three parts, Java doesn't work.
The JVM is necessary for both JDK and JRE. It is platform-dependent and its main scope is to convert byte-code to machine-specific code. It is composed of three parts: Class Loader Subsystem, Runtime Data Areas, and Execution Engine.
The JRE comprehends different software tools with the scope to develop and execute the Java application. JRE is composed of several libraries that run the Java program

The JDK is platform-specific and it is used to create Java programs: it converts the source code into a form that the JVM and the JRE can execute. It comprehends different tools required to compile, debug and run the source code. Since these parts are necessary to implement a Java program and make it platform-independent, the selected boards will have to support it as well. Implementing a JVM for a micro-controller is very difficult, due to its low memory capacity and limited computing capacity. Another factor that has been taken into account is the need of a display and a camera. For this reason, it is necessary to use two micro-computers that run a proper operating system with the JVM already integrated.

After a research about the micro-computers available and their characteristics, two *Raspberry Pi 4 Model B* have been chosen. The recommended operating system for the Raspberry Pi 4 Model B is the *Raspberry Pi OS*, which can be easily installed on a Micro-SD through the *Raspberry Pi Imager* program. It is Debian-based and it provides a desktop environment, similar to macOS and Microsoft Windows. APT (Advanced Packaging Tool) is used to install, upgrade and manage software. On the Raspberry, a SD card support is present to insert the SD card for loading the operating system and data storage. The processor used in the Raspberry Pi 4 Model B is the *BCM2711*, a quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz broadcome chip. The RAM implemented is a 8GB LPDDR4-2400 SDRAM, fully supported by the CPU. The Raspberry Pi 4 comes with forty general-purpose I/O pins. Two USB 3.0 and two USB 2.0 ports are present, together with a Gigabit Ethernet. In addition, 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE are supported by the Raspberry Pi 4. The Raspberry Pi 4 Model B supports the communication with the Raspberry Pi Touch Display and the Raspberry Pi Camera Module 2.
Two 7" touchscreen display for Raspberry Pi are used to create the graphical user-interface for the EVSE and the EV. The 800 x 480 RGB LCD display connects to Raspberry Pi 4 Model B via an adapter board that handles power and signal conversion.

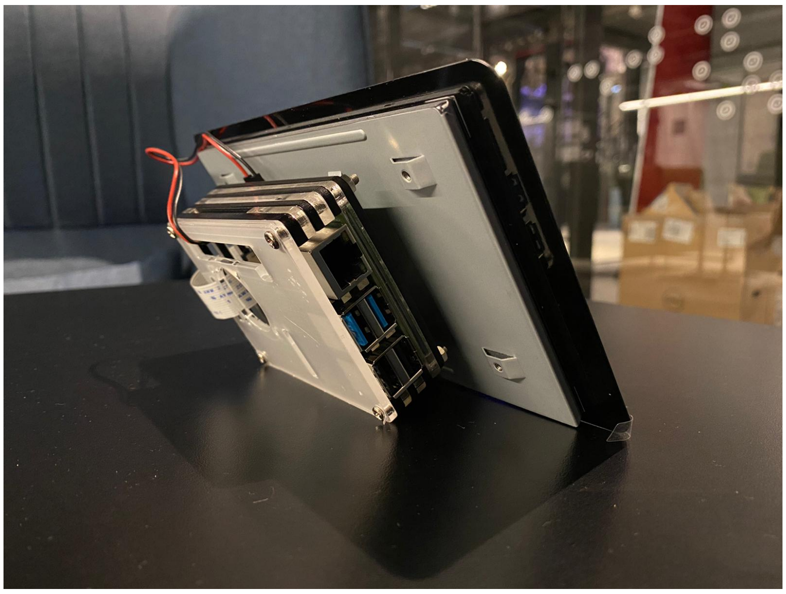


Figure 4.2: Hardware used for the demo

The 8MP camera is capable of taking high-resolution photographs, along with full HD 1080p video, and can be fully controlled through the various software tools. The camera integrate a Sony IMX219 sensor, that assures image quality, colour fidelity, and low-light performance. The camera is used to scan the QR code used by the user to start the electric vehicle charging process. In this demo, only Plug&Charge and QR code method payments are implemented, but the procedures to use an RFID card or a credit card are the same as the last mentioned.

4.3 Integration of RiseV2G

Since RiseV2G is realized to test an EVSE-EV communication during the charging process, every message contains static information. All the parameters that the user should modify, or that can change during the day, are initialized to standard values in the properties files (one for the EVCC, one for the SECC). Therefore, all these parameters (e.g if the charging mode is AC or DC) can be selected and modified by opening the Java program. Obviously, the scope of the demo is to create a user-friendly charging session, for this reason, the parameters of RiseV2G are modified by the user (GUI) or by the PFS, always through the BL synchronization. Moreover, RiseV2G simulates the charging loop exchange of messages one-hundred

times: this is just to test that the messages are exchanged correctly. Since the proof of concept objective is to simulate a real charging session, the communication ends when the battery is fully charged (battery state of charge is simulated by the PFS program of the EVCC).
For these reasons, RiseV2G has to be managed by the BL, that pause the EV-EVSE communication to wait for the user interaction. Moreover, it is the BL that stops the charging session when it receives the information from the PFS that the battery is fully charged, or when the user selects it on the EV tablet. For this purpose, a socket communication between RiseV2G and the BL is created for both the EVCC and SECC parts. Any time RiseV2G changes its state, it has to communicate it to the BL that can stop RiseV2G, in order to wait for the user interaction or just to check the parameters coming from the EVCC-PFS or the EV-PFS. Since the socket has to be called by both the EVCC and the SECC, it has been implemented in the RISE-V2G-Shared folder, in the *MiscUtils* file.
The socket handle the RiseV2G-BL communication in both ways: the RiseV2G starts the socket communication in order to communicate to the BL its current state and all the parameters coming from the other part (i.e. at the EVCC-RiseV2G, the parameters coming from the SECC can only be transmitted through ISO15118, therefore, these parameters must be passed to the BL that will handle and pass them to the EVCC-PFS and to the EVCC-GUI). The BL starts the socket anytime it wants to communicate something to the other part (e.g. the EV user has selected the departure time that has to be sent to the SECC through RiseV2G) using ISO15118 communication standard.
How the BL synchronize and handle the EV and the EVSE processes is better explained in the dedicated paragraph of this chapter.

It follows a list containing all the parameters modified at the EV part, by the PFS, or by the user via the GUI:

- **Payment option**: It is selected by the user through the graphical user interface of the electric vehicle (EV-GUI). It is sent within the *PaymentServiceSelectionReq* message in all four different charging sessions (AC-EIM, AC-PnC, DC-EIM, DC-PnC).

- **Energy transfer-mode requested**: It arrives from the EV power flow simulator and it is sent within the *ChargeParameterDiscoveryReq* message in all the possible charging sessions. It can be AC three-phase core, AC single-phase core, DC core, DC extended, or DC combo core.

- **Departure time**: It is selected by the user via the EV graphical user interface. The user can communicate the departure time in hours and minutes, but the ISO15118 requires it in seconds, therefore conversion is necessary. It is sent within the *ChargeParameterDiscoveryReq* message in all four possible charging modes.
- **Energy amount**: It is the expected requested energy to fulfill the user-configured charging goal for the current charging session. It is calculated by the EV-PFS and sent for all the available charging modes.
- **EV maximum voltage**: It is the maximum voltage the EV can accept. It is defined in the EV-PFS and sent within the *ChargeParameterDiscoveryReq* message in all the different charging modes.
- **EV maximum current**: It is the maximum current supported by the EV. It is defined in the EV-PFS and sent within the *ChargeParameterDiscoveryReq* message in all the different charging modes.
- **EV minimum current**: It defines the minimum value of current under which the charging is no more energy/cost efficient. It is sent only in AC charging modes, in the *ChargeParameterDiscoveryReq* message.
- **EV ready**: If it is set to TRUE, the EV is ready to charge. It is defined only for DC charging modes, in the EV-PFS.
- **EV state of charge**: It defines the state of charge of the EV (0-100). It is defined only for DC charging modes, in the EV-PFS. Sent within the *CableCheckReq* and *CurrentDemandReq* messages, together with the "EV ready" property.
- **Chosen SA schedule ID**: The user can choose between three different charging profiles proposed by the EVSE. Every charging profile is defined in the "Sales Tariff" property, defined later, with a different ID. It is sent within the *PowerDeliveryReq* message in all the charging modes (AC-EIM, AC-PnC, DC-EIM, DC-PnC).

- **Target Voltage**: It is the voltage required by the EV to start the DC charging mode. It is defined in the EV-PFS and sent within the *CurrentDemandReq* message.
- **Target Current**: It is the current required by the EV to start the DC charging mode. It is defined in the EV-PFS and sent within the *CurrentDemandReq* message.
- **Maximum voltage limit**: Maximum voltage accepted by the EV in DC charging mode. This parameter is defined in the EV-PFS and sent within the *CurrentDemandReq* message.
- **Maximum current limit**: Maximum current accepted by the EV in DC charging mode. This parameter is defined in the EV-PFS and sent within the *CurrentDemandReq* message.
- **Maximum power limit**: Maximum power accepted by the EV in DC charging mode. This parameter is defined in the EV-PFS and sent within the *CurrentDemandReq* message.
- **Bulk charging completed**: If set to TRUE, the EV indicates that 80% of battery capacity has been reached. It is calculated in the EV-PFS and sent within the *CurrentDemandReq* message.
- **Charging completed**: If set to TRUE, the EV indicates that the EV charging is completed. It is calculated in the EV-PFS and sent within the *CurrentDemandReq* message.
- **Remaining time to bulk SoC**: Calculated in the EV-PFS and sent within the *CurrentDemandReq* message.
- **Remaining time to full SoC**: Calculated in the EV-PFS and sent within the *CurrentDemandReq* message.
- **Charging loop status**: It is set to *TRUE* if the charging process is ongoing, to *FALSE* if the charging session is terminated, to *RENEGOTIATION* if the user has requested a renegotiation of the charging profile previously selected, to *PAUSE* if the user has requested a pause of the charging session.

The SECC parameters modified by the PFS or by the user via the GUI are the following:

- **EVSE nominal voltage**: It is the line voltage supported by the EVSE. It is defined in the EVSE-PFS and sent within the *ChargeParameterDiscoveryRes* message in AC charging modes.
- **EVSE maximum current**: It is the maximum current supported by the EVSE. It is defined in the EVSE-PFS and sent within the *ChargeParameter-DiscoveryRes* message in AC charging modes.
- **EVSE maximum voltage limit**: It is the maximum voltage the EVSE can deliver. It is defined in the EVSE-PFS and sent within the *ChargeParameter-DiscoveryRes* message in DC charging modes.
- **EVSE minimum voltage limit**: It is the minimum voltage the EVSE can deliver with the expected accuracy. It is defined in the EVSE-PFS and sent within the *ChargeParameterDiscoveryRes* message in DC charging modes.
- **EVSE maximum current limit**: It is the maximum current the EVSE can deliver. It is defined in the EVSE-PFS and sent within the *ChargeParameter-DiscoveryRes* message in DC charging modes.
- **EVSE minimum current limit**: It is the minimum current the EVSE can deliver with the expected accuracy. It is defined in the EVSE-PFS and sent within the *ChargeParameterDiscoveryRes* message in DC charging modes.
- **EVSE maximum power limit**: It is the maximum power the EVSE can deliver. It is defined in the EVSE-PFS and sent within the *ChargeParameter-DiscoveryRes* message in DC charging modes.
- **Peak current ripple**: It is the peak-to-peak magnitude of the current ripple of the EVSE. It is defined in the EVSE-PFS and sent within the *ChargeParameterDiscoveryRes* message in DC charging modes.
- **Meter reading**: It includes the meter-Info record containing the latest meter reading and other meter relevant data. It is defined in the EVSE-PFS and sent within the *ChargingStatusRes* message in DC charging modes.

- **Sales Tariff**: This is the most important and complex parameter modified. In the *ChargeParameterDiscoveryRes*, the *SAScheduleList* containing up to three *SAScheduleTuples* is sent. Every SAScheduleTuple is a different charging profile that the user can choose: there are up to three different charging profiles, each of them with several information about the maximum power delivered from the EVSE, the price against the time, the percentage of energy coming from renewable sources and the emission of carbon dioxide.
  For this reason, every SAScheduleTuple is composed of:
  - an **ID**, used as unique identifier during the charging communication;
  - a **PMaxSchedule** containing up to 1024 **PMaxScheduleEntry** encapsulating elements describing all relevant details about the power delivered;
  - one **Sales Tariff** containing information about the energy provided during the charging time.

  Every **PmaxScheduleEntry** contains the *RelativeTimeInterval* and the *PMax* elements describing the trend of the power delivered by the EVSE from the current time until the departure time. This means that for each of the three possible charging profiles, a graph describing the power trend along the charging time is defined, with a maximum of 1024 segments (PmaxScheduleEntry).

  For every charging profile, another graph is defined and it is the **Sales Tariff** graph. Every graph can contains up to 1024 **Sales Tariff Entry** (segments) containing, for each segment time, the following information:

  - **EPriceLevel**: for each segment an EPriceLevel is assigned, defining the cost of the charging during that time segment. High EPriceLevels mean expensive charging, lower EPriceLevels mean cheaper expensive charging segments.
  - **Price percentage**: for each segment a price-percentage is assigned, defining the percentage of the maximum cost of the charging, during that time segment.
  - **Renewable energy percentage**: it defines, for every time segment, the percentage of energy coming from renewable sources. This allows the user to prefer a charging profile with an elevate amount of energy obtained from renewable sources.

– **Carbon dioxide emission**: it defines, for every time segment, the Carbon Dioxide emissions, in grams per kWh.

It follows an example of the Sales Tariff graph of the charging profile. In addition to this graph, a parallel graph describing the maximum power delivered by the EVSE is provided in the demo.

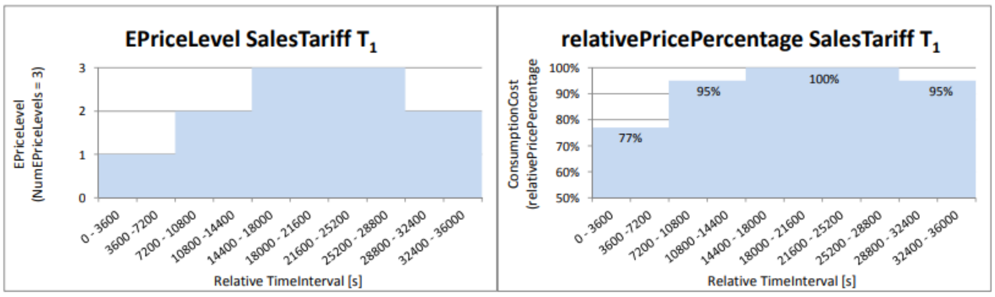


Figure 4.3: Example of a Sales Tariff, [14]

To define the above-described graphs, for each charging profile the following parameters are sent to RiseV2G from the EVSE-PFS:

– **Times vector**: Vector defining the time division, from 0 until the departure time selected by the user.

– **Eprice levels vector**: Vector defining the EPrice level for each time segment.

– **Percentages prices vector**: Vector defining the percentage of the total price for each time segment.

– **Renewable percentages vector**: Vector defining the percentage of energy obtained from renewable sources for each time segment.

– **Carbon emission vector**: Vector defining the carbon Dioxide emissions for each time segment.

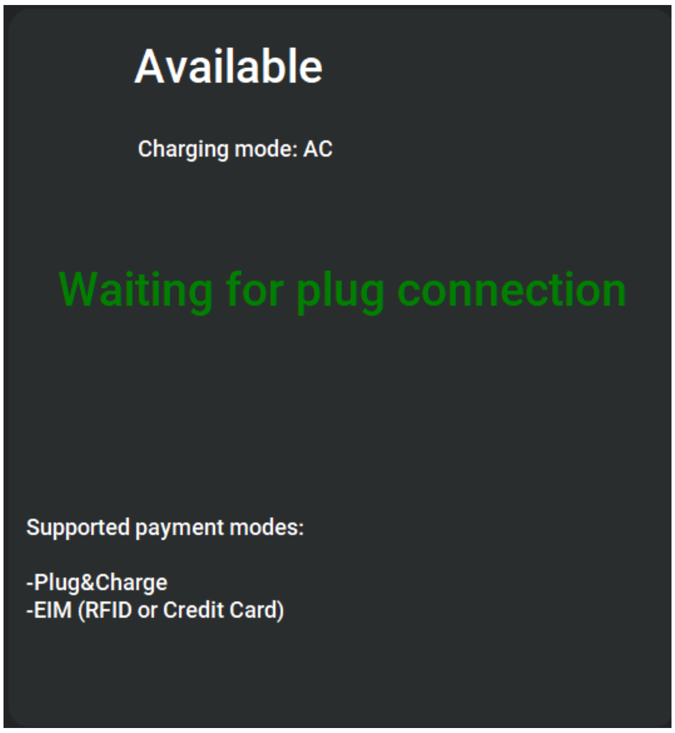
4.4 EV/EVSE Graphical User Interface

The user interaction with the electric vehicle is essential. Two graphical user interfaces are realized: one for the charge point (**EVSE-GUI**), and one for the

electric vehicle (**EV-GUI**).
Both the GUIs exchange information with the relative BL program about the user choices and the charging process characteristics.
The graphical user interfaces are programmed in Python with the *CustomTKinter* library. It is a UI library based on the classic *TKinter*, which allows the programmer to create a modern graphical user interface.
The scope of the **EVSE-GUI** is to show the user how to start the charging process and then display the characteristics of the real-time EV charging.
The following pages are created and exposed to the user:

- **Startup screen**: The first page shown to the user indicates the starting of the charging station, together with the Accenture logo.

- **Standby screen**: This page is displayed every time no electric vehicles are connected to the charging station. The available charging modes supported by the charge point are indicated (AC/DC). Also, the different charging payment options are shown to the user (RFID simulated by a QR code and/or Plug&Charge).


Available

Charging mode: AC

Waiting for plug connection

Supported payment modes:

-Plug&Charge
-EIM (RFID or Credit Card)

Figure 4.4: Standby Screen

- **Identification screen**: When the EV is plugged in, the user is requested to specify the payment option through the EV-GUI. This page simply shows a message confirming the correct connection of the EV and indicates the user to choose the payment option on the EV tablet.
- **EIM screen**: If the EIM payment method is chosen, a page requesting to show the QR code to the camera is prompted. The user has to scan the QR code to authenticate himself and let the charging process start.
- **Certificates screen**: If the user chooses Plug&Charge as the payment method, the EVSE and the EV exchange information validating the certificates. During this messages sequence, a page that displays the ongoing certificate processing is launched. Also, the conclusions of the certificate validation are displayed to the user:
  - The certificate has been uploaded;
  - A new certificate has been installed;

  - The current certificate is valid;

- **Waiting screen**: Meanwhile the user chooses the departure time and the charging profile, and the charge point display a page requesting the user to make his choices.

- **Charging preparation screen**: When all the charging properties have been defined, the EVSE proceeds to prepare the charging session. At this moment, a page indicating the charging preparation is displayed.

- **EV Charging in progress screen**: During the EV charging, the EVSE shows some information about the current process: real-time voltage, real-time current, real-time power, energy delivered, charging time remaining, and charging mode (AC/DC). Plus, three buttons are present:

  - **Pause**: The user can indicate to pause the charging session, for example, if the battery charging level is almost reached but the departure time is still distant.
  - **Terminate**: The user can terminate the charging session at any moment, via the EVSE or the EV tablet.
  - **Renegotiation**: If the user wants to change the charging profile previously agreed, he can activate the charging renegotiation through this button.

  If any of the previous buttons are selected by the user, the information is sent to the EVSE business logic that will handle the user request.

- **Charging completed screen**: When the charging process is terminated, this page is displayed showing some charging information: initial and final charging time, total energy delivered, time taken to complete the EV charging, payment option utilized, and carbon dioxide emissions.

The objective of the **EV-GUI** is to communicate with the user and allow him to express his preferences about the vehicle charging.
The following pages are realized and displayed to the user:

- **Startup screen**: The first page shown to the user indicates the starting of the electric vehicle, together with the Accenture logo.

- **Connect-the-vehicle screen**: A page requesting the user to connect the vehicle to the charging point is displayed.
- **Identification screen**: In this page, the user can select the payment option between the ones supported by the EVSE. Depending on the choice, a new page is displayed.
- **EIM screen**: If the user has chosen the EIM method, the user is requested to scan the QR code at the charging station. The validity of the QR code is then shown to the user, informing him about the correct authentication.
- **Certificates screen**: As per the EVSE-GUI, if the user chooses Plug&Charge as the payment method, the EVSE and the EV exchange information validating the certificates. During this messages sequence, a page that displays the ongoing certificate processing is launched. Also, the conclusions of the certificate validation are displayed to the user:
  - The certificate has been uploaded;
  - A new certificate has been installed;
  - The current certificate is valid;
- **Departure time screen**: The user is asked to insert the expected departure time. This information is needed by the EVSE to calculate the possible charging profiles.
- **Charging profile screen**: The charging profiles created by the EVSE-PFS are displayed in this page. The user can analyze them, choosing the most suitable profile.
- **Charging preparation screen**: When all the charging properties have been defined, the EVSE proceeds to prepare the charging session. At this moment, a page indicating the charging preparation is displayed.
- **EV Charging in progress screen**: As per the EVSE-GUI, during the EV charging, the EV-GUI shows some information about the current process: real-time voltage, real-time current, real-time power, energy delivered, battery state of health, battery state of charge, charging time remaining, and charging mode (AC/DC). Plus, three buttons are present:

  - **Pause**: The user can indicate to pause the charging session, for example, if the battery charging level is almost reached but the departure time is still distant.
  - **Terminate**: The user can terminate the charging session at any moment, via the EVSE or the EV tablet.
  - **Renegotiation**: If the user wants to change the charging profile previously agreed, he can activate the charging renegotiation through this button.

  If any of the previous buttons are selected by the user, the information is sent to the EV business logic that will handle the user request.

- **Charging completed screen**: As per the EVSE-GUI, when the charging process is terminated, this page is displayed showing some charging information: initial and final charging time, total energy delivered, time taken to complete the EV charging, payment option utilized, and carbon dioxide emissions.

4.5 EV/EVSE Power Flow Simulator

Since the focus of the PoC is on the communication between the EV and the EVSE during the charging process, the power part is simulated by the two programs **EV-PFS** and **EVSE-PFS**.

The main objective of the EV-PFS is to define the EV charging properties. For example, the energy transfer mode required by the EV is defined in this program, and it can be AC three-phase core, AC single-phase core, DC core, DC extended, or DC combo core. Based on this parameter, the charging session proceeds following the AC or DC scheme.
All the EV properties can be chosen before the test phase, or they can be generated randomly, with different weights, in order to recreate the most possible situation. The Python functions used to generate the parameters are two: the **random.choices**, that allows the programmer to assign to each element a different probability to be generated, and the **random.gauss** that allows the programmer to obtain a random number based on the gaussian distribution defined.
In case of AC charging mode, the following properties are generated randomly based on the standard EV values:

- **EV maximum voltage**
- **EV maximum current**
- **EV minimum current**

In case of DC charging mode, the following properties are generated randomly based on the standard EV values:

- **EV maximum voltage**
- **EV maximum current**
- **EV maximum power**
- **EV target voltage**
- **EV target current**

All these properties are defined before the testing phase and they can not be changed during the charging simulation.
The following values, instead, change during the charging process:

- **Energy amount**: Based on the EV state of charge previously (randomly) generated, the EV-PFS calculates the amount of energy needed to reach the user goal, that can be fully charged (100%) or bulk charged (80%). Of course, this value depends on the EV battery capacity, a value declared in the EV-PFS but never sent to the EVSE.
- **EV state of charge**: The EV state of charge is a number between zero and one-hundred that defines how much the battery pack is charged. It is initialized randomly at a value in the range (0-70) and during the charging process it is changed based on the power delivered by the EVSE.
- **Bulk charging completed**: It is set to *TRUE* if the EV state of charged has reached the 80% of the maximum battery capacity. This could lead to the charging session termination, if the user has requested a bulk charging.
- **Charging completed**: It is set to *TRUE* if the EV state of charged has reached the 100% of the maximum battery capacity. This lead to the charging session termination, since the EV can not be charged more.

- **Remaining time to bulk SoC**: Based on the power that the EVSE is delivering, the PFS calculates the needed time to reach the 80% battery charge goal.

- **Remaining time to full SoC**: Based on the power that the EVSE is delivering, the PFS calculates the needed time to reach the 100% battery charge goal.

The main objective of the EVSE-PFS is to define the EVSE properties regarding the power, the current, and the voltage that can be delivered. As per the EV-PFS, all the EVSE properties can be chosen before the test phase, or they can be generated randomly, with different weights or with a gaussian distribution, in order to recreate the most possible situation.
The following properties are defined in this project file:

- **EVSE nominal voltage**
- **EVSE maximum current**
- **EVSE maximum voltage limit**
- **EVSE minimum voltage limit**
- **EVSE maximum current limit**
- **EVSE minimum current limit**
- **EVSE maximum power limit**
- **Peak current ripple**
- **Meter reading**: This parameter is sent by the EVSE during the charging session loop, and it is defined as the current energy delivered by the EVSE during the EV charging, therefore expressed in Watt-hour.
- **Sales Tariff**: Three different charging profiles are created. Each of them has different values of power deliverable, different percentages of renewable energy, carbon emission, and price per time.

4.6 EV/EVSE Business Logic

All the demo parts need to collaborate to create the final product. The BL programs' scope is to organize and synchronize the programs' operations. Every program (PFS, GUI, and the implementation of RiseV2G) continuously exchange data with the BLs that manage their operations. This is valid for the EV and for the EVSE. Since different programs need to run at the same moment, different threads are created and organized with the *threading* Python function, which allows for the creation of a multi-threading program.
The **EV-BL** and **EVSE-BL** are similar in structure, but they differ in the operations managed, since the EV and the EVSE functions during the charging process are quite different.
Since all the parameters need to share different global variables (such as the charging mode, the payment option selected by the user, etc..), two files containing all these global properties are created: EVCC_Properties for the EV part, and SECC_Properties for the EVSE part. In these files, all the data that the programs need to exchange with each other are defined and can be modified or read by every project. Therefore, when the charging profile is selected by the user through the EV-GUI, a parameter defined inside the EVCC_Properties is modified and the new value can now be read by the EV-BL. The **EV-BL** methods are the following:

- **___init___ method**: This method is called when the only class of the EV-BL program is called. At the raspberry simulating the EV, the first program launched is the EV-BL, therefore its main is called. The only code line of the main is the recall of the BL class, which the ___init___ method starts the different threads. The main thread recall the *BL_loop* method, which continuously check for the RiseV2G state to understand the next step to be completed. Another crucial thread launched in the initial method is the *socket* one: it is called every time the BL needs to communicate with the RiseV2G implementation. The last thread launched is the one about the GUI: the first page showing the *startup screen* is displayed for about six seconds. After the startup screen, the others page are managed by the BL_loop method.

- **BL_loop**: This loop follows the message sequence defined in the ISO15118-2 standard, managing RiseV2G to send/receive the correct messages and to stop it, allowing eventual calculations or user interactions. Instead of a RiseV2G free-running, the BL needs to frequently stop it, request the current RiseV2G state via the socket method, and, based on the RiseV2G state, a specific action is executed.

After the startup screen, the *connect-the-vehicle* screen is displayed. The user has to connect the vehicle to the charging point. When this happens, the communication between the EV and the EVSE can start through RiseV2G. For this purpose, the BL opens the socket and lets RiseV2G start with the first communication setup messages. The EV-PFS properties are sent to RiseV2G at this moment.

When RiseV2G has to send a *PaymentServiceSelectionReq* message, the BL stops it and launches the *Identification screen* on the tablet, asking the user to select the payment option, among those supported by the EVSE. This information is sent by the BL to the RiseV2G through the socket method.

If the Plug&Charge method is chosen, the EV-EVSE exchange information checking the certificate validity and if an update/installation is needed. During these messages, the *Certificates screen* is displayed, informing the user to wait for the processing of the certificate.

If EIM is chosen, the EV-BL launches the *EIM screen*, requesting the user to scan the QR code to authenticate himself. After that, the user has to indicate the expected departure time, through the *Departure time screen.* The EV-BL converts this time in seconds and sends it to RiseV2G.

At this point, when the RiseV2G message to send is *PowerDeliveryReq*, the user has to choose the charging profile shown in the *Charging profile screen*, among the ones proposed by the EVSE. The charging session can start after a certain time during which the EVSE prepares itself to deliver the power requested. If the charging mode is DC, the target voltage and current must be equal to the present voltage and current at the charge point.

During the charging loop message sequence, the *EV Charging in progress screen* is shown to the user. During this time, the user can request a pause of the charging session: the EV-BL communicates this intention to EVCC-RiseV2G which will send this information to the SECC part. The user, now, can restart the charging session with the same charging profile previously established.

Also, the user can request a renegotiation: if EVCC plans to perform a renegotiation, it shall start by sending a *PowerDeliveryReq* with the parameter *ChargeProgress* set to *Renegotiate*, followed by an exchange of *ChargeParameterDiscoveryReq/Res* and *PowerDeviveryReq/Res* message-pairs and then re-enter the charging loop.
The user can also ask to terminate the charging session. The charging can be stopped also automatically by the EVCC when the EV battery, simulated in the EV-PFS, is fully charged. At this point, the final *Charging completed screen* is displayed, sharing some information about the just-concluded charging session.

- **PFS-start method**: This method is called by the BL-loop to start the EV-PFS program that will randomly generate all the properties related to the battery. After that, the EV-PFS will continue to communicate with the BL, sending information about the current battery state of charge, and if the charging is completed.
- **Socket method**: This method starts a socket-based communication with the Java program RiseV2G. The BL acts as the client, asking for example RiseV2G to communicate its current state or some information passed from the EVSE, that can only communicate with RiseV2G to be compliant with ISO15118. The socket is also called by the BL to modify the parameters set in the *EVCCConfig.priorities* file.
- **Open_page method**: This method is called anytime the BL wants to open a new page on the tablet. It recalls a different EV-GUI class method, which will open the selected page.

The **EVSE-BL** is similar to the EV_BL concerning the structure and methods, but the BL_loop is quite different. The methods are the following:

- **___init___ method**: This method is called when the only class of the EVSE-BL program is called. Also at the raspberry simulating the EVSE, the first program to be launched is the EVSE-BL, therefore its main is called. The only code line of the main is the recall of the BL class, in which the ___init___ method starts the different threads. The main thread recalls the *BL_loop* method, which continuously checks for the EVSE-RiseV2G state to understand the next step to be completed. Another crucial thread launched in the initial method is the *socket* one: it is called every time the BL needs to communicate with the RiseV2G implementation. The last thread launched is the one about the GUI: the first page showing the *startup screen* is displayed for about six seconds. After the startup screen, the others page are managed by the BL_loop method.

- **BL_loop**: This loop follows the message sequence defined in the ISO15118-2 standard, managing RiseV2G to send/receive the correct messages and to stop it, allowing eventual calculations or user interactions. Instead of a RiseV2G free-running, the BL needs to frequently stop it, request the current RiseV2G state via the socket method, and, based on the RiseV2G state, a specific action is executed.
  After the startup screen, the *Standby screen* is displayed, waiting for the user to connect the EV to the charge point. When the EV is plugged in, the EVSE-PFS is launched to generate the charging parameters to be sent to SECC-RiseV2G via the socket method.
  The EVSE has to wait for the user to choose the payment method, therefore the *Identification screen* is shown. The payment option is received by the SECC-RiseV2G through the *PaymentServiceSelectionReq* message sent by the EVCC-RiseV2G.
  If the payment method chosen is EIM, the *EIM screen* is displayed indicating the user to bring the QR code closer to the camera. The *QR code method* is called in order to scan the QR code exposed by the user's smartphone.
  If the PnC method has been chosen, the certificate checking messages need to be exchanged, therefore the *Certificates screen* informing the user about the EV certificate status is shown.
  At this point, the user has to choose the departure time and the charging profile previously computed by the EVSE-PFS: therefore, during this time, a *Waiting screen* is displayed on the charge point tablet. If the charging mode requested by the EV is DC, to start the charging process the EVSE must provide the exact voltage and current requested by the EV. Therefore, a charging preparation procedure has to be accomplished. During this time the *Charging preparation screen* is displayed.
  Now the charging process can start: the *EV Charging in progress screen* is

shown during the message sequence loop. All the properties handled by the EVSE-PFS are continuously updated and sent to the EVSE-BL to simulate the power values. As per the EV-GUI, the user can choose to stop, pause or renegotiate the charging session.
If the pause is requested, the EVSE-BL communicates this intention to SECC-RiseV2G which will send this information to the EVCC-RiseV2G. The user, then, can restart the charging session with the same charging profile previously established.
If the renegotiation is requested, the EVSE-BL tells the SECC-RiseV2G to send a *ChargingStatusRes* message with EVSENotification set to ReNegotiation to ask EVCC to proceed to a renegotiation. Then, the EVCC shall start another round of negotiation by sending PowerDeliveryReq followed by an exchange of *ChargeParameterDiscoveryReq/Res* and *PowerDeviveryReq/Res* message-pairs.
If the termination of the charging session has been requested by the user, or by the EV-BL because the charging is completed, the final *Charging completed screen* is displayed, sharing some information about the just-concluded charging session.

- **QR code method**: This method is called by the BL_loop method in case the user has chosen the EIM as the payment method. The RFID is substituted with a QR code, that can be scanned with the *Raspberry camera*. This is done with the *image* and *zbarlight* Python libraries.

- **PFS-start method**: This method is called by the BL_loop to start the EVSE-PFS program that will randomly generate all the properties related to the charge point capacities. After that, the EVSE-PFS will continue to communicate with the BL, sending information about the *meter reading* and the crucial *sales tariff*, indicating the charging profiles.

- **Socket method**: This method starts a socket-based communication with the Java program RiseV2G. The BL acts as the client, asking for example RiseV2G to communicate its current state or some information passed from the EVSE, that can only communicate with RiseV2G to be compliant with ISO15118. The socket is also called by the BL to modify the parameters set in the *SECCConfig.priorities* file. These properties are then sent to the

EV-RiseV2G through the correct messages, as defined in [14].

- **Open_page method**: As per the EV-BL, this method is called anytime the BL wants to open a new page on the tablet. It recalls a different EVSE-GUI class method, which will open the selected page.

Chapter 5

Testing

In this chapter, different charging sessions are simulated, in order to test the demo functioning in all the possible cases. The use cases analyzed are the following: DC charging mode with PnC or QR code (simulation of RFID card), AC charging mode with PnC or QR code. In both simulations, a renegotiation and a pause of the charging session are requested, in order to verify the correct operation of the proof of concept. For all the use cases, the user point of view is analyzed, together with the actual functioning of the demo programs and of the implementation of RiseV2G.

5.1 AC/DC charging mode simulation with "Plug and Charge"

For AC charging mode simulation, the use case tests the charging process of an electric vehicle with the onboard charger and supporting the Plug&Charge feature. Therefore, this test refers to the vehicles with the OEM provisioning certificate installed at the EV production moment. If it is the first time that the EV is connected to the charging point, certificate installation is needed. This is handled correctly by the SECC installing a new certificate created locally. In a real implementation, the certificate can also be issued by other actors present in the PKI. Obviously, when the user is asked to insert the preferred payment option, Plug&Charge is selected. The ongoing process related to the checking of the certificates is executed in the background, meanwhile, the user can see a screen

telling him to wait for the certificate process handling. This operation usually doesn't take longer than a few seconds because the certificates are all stored locally: in a real use case, the certificate can be downloaded by a secondary actor, leading to a longer waiting time.
At the beginning of the demo, all the EV and EVSE power-related values are generated by the EV-PFS and EVSE-PFS programs. These values are generated randomly, but according to real values implemented in the market. All these properties are then used during the charging process to simulate, for example, the battery state of charge during the charging process, or the energy delivered by the charging point during the connection. Some of these values will be uploaded during the EV charging, and some of them will be sent to the other part within the correct message, as defined in the ISO15518-2 document standard.
To start the EV-EVSE communication, the two raspberry need to be connected with the Ethernet cable, simulating the connection of the electric vehicle at the charging point. In this PoC, the power part of the charging process is not considered, since the focus is on the communication between the two sides, based on the requirements of the ISO15118 standard.
After that, the user is requested to insert the departure time: based on this value, the EVSE-PFS generates the three different charging profiles with values close to the real ones. These charging profiles are then passed to the electric vehicle and shown on its screen: now, the user can choose the charging profile that he prefers, having information related to the price, the energy coming from renewable sources, and the emission of carbon dioxide.
Now, the charging process is ready to start: during this time, a page showing information about EV charging is displayed on the EV display. Meanwhile, the battery state of charge, simulated in the EV-PFS program, increases according to the power delivered by the charging point (also this value is simulated, in the EVSE-PFS program). Since the EV battery would take hours to completely charge, the battery charging is sped up, in order to complete the test within a few minutes.
If no action is executed during the charging process, this is completed correctly and a final screen showing the main charging information is displayed: here, the user can see how much time the charging has lasted, the energy delivered, and the carbon dioxide emissions.
Otherwise, the user can ask for a pause of the charging session: in this way the

charging is just paused, meaning that the battery state of charge no longer increases, and the energy delivered is null. The user can restart the charging process whenever he wants, leading to the battery state of charge increasing.
The user can also ask for the renegotiation of the charging profile: the state is changed from *ChargingStatusReq/Res* to *ChargeParamterDiscoveryReq/Res*, in order to ask again the user for the departure time. Three new charging profiles are proposed to the user which can select the most suitable one and let the charging process start again. This operation can be made an infinite number of times during the charging process.
If the DC charging mode is tested to carry out the charging process for an EV with no onboard charger, the functioning of the demo is basically the same, adding some messages regarding the alignment of the voltage requested by the EVCC and the one present at the charging point. From the user's point of view, the charging process gets completed in the same way.
One of the problems that occurred during this testing phase, is the communication within the BL and the RiseV2G implementation. This is because the SECC can not stop the communication, otherwise, an error coming from the timeout expiration occurs. For this reason, the communication between the EVSE-BL and EVSE-RiseV2G with the socket must occur fast, leading to no waiting time. On the EVCC side, instead, since the user has to have time to think about his choices, no timeout limit is present. Therefore, the EVCC is free to pause the communication on how long he needs, but a timeout has been inserted to reset the communication in case a problem occurs and it's not detected (for example the user is not able to insert his choices on the display).

5.2 AC/DC charging mode simulation with QR code

If the Plug&Charge feature is not supported by the charging point or by the electric car, other methods must be present to allow the user to charge the vehicle. Usually, the main methods are the following: credit card, app identification, and QR code scanning. In this PoC, the QR code is used as an alternative way of identification.

The functioning of the demo is the same and the steps followed by the two parts are the same. The only difference occurs when the *PaymentServiceSelectionReq*

message must be sent by the EVCC. Within this message the *SelectedPaymentOption* parameter is sent and the identification method chosen: Pluig&Charge or EIM (QR code).
The EIM method is selected in the following cases:

- The charging point doesn't support the Plug&Charge feature;
- The EV doesn't support the Plug&Charge feature;
- The EV certificate is expired but there is no possibility to install a new one (e.g. the charging point doesn't have the new certificate stored locally, so it would need to download it from a secondary actor but the connection is not possible at the moment);
- The EV certificate is expired but the user doesn't want to install a new one;
- The user chooses to pay with a credit card or the QR code.

When the EIM payment method is chosen, the user is asked to show the QR code at the charging point camera. The QR code is intended to be provided to the user in the following two ways:

- Given by the car manufacturer when the EV is bought. If the car owner chooses to sell the vehicle, the QR code would not be valid anymore: the new owner would need to contact the OEM to request a new personalized QR code;
- When establishing a contract with an EMSP, an app is included in the deal. Within this app, usually, the QR code generation is present and it allows the user to start the charging session.

The QR code is analyzed correctly, containing the following information:

- Name, surname, and general information about the EV owner;
- Details about the payment method: in this demo, the fake credit card information is contained in the QR code (16 numbers code, CVV number, and card expiration date).

The SECC checks if the details provided match the one contained in a stored document: in this demo, these data are contained in the SECC-PFS file, but in a real charging station this procedure would need to be verified with several secondary actors, such as the EMSP and the bank.
If the identification is verified, the user is asked to insert the departure time and the demo goes on as usual. Instead, if the identification fails, the communication is interrupted and a message informing the user is displayed. Now, the user can choose the Plug&Charge payment method if it is supported, or retry to scan the QR code.

Chapter 6

Conclusions

6.1 Achieved results

The purpose of the thesis project was to create a Proof of Concept focused on the current standard implementation for V1G and V2G, also called ISO15118. This project of thesis is based on a specific market need. Even if at the moment few EV and Charge Point Operators (CPO) are already compliant with ISO15118 standard, the future charging experience would rely on this standard, and all the entities in the E-Mobility panorama would have to be compliant. This will add value to the charging, improving the experience of the users and making the electrification more sustainable. The thesis work has been divided into two phases: a first phase in which an in-depth study of the ISO15118 standard family has been carried on, followed by the realization of the demo simulating the charging process based on the standard.
All eight documents of the ISO15118 family have been studied, with a focus on the ISO15118-1 standard, which defines the general information and use-case definition for conductive and wireless high-level communication between the electric vehicle and the charging point, and the ISO15118-2 standard, which defines all the application layer messages exchanged between the electric vehicle and the charging station during the charging process.
Two Raspberry Pi 4 Model B have been used to simulate the charging point and the electric vehicle, both equipped with a touch display and a camera to interact with the user, giving the possibility to insert his preferences about the charging

process.
The open-source protocol stack implementing the ISO15118-2 messages and requirements, called RiseV2G, has been thoroughly studied and then modified, in order to be integrated into the PoC, both on the electric vehicle and the charging station sides. Then, the python programs to handle the behavior of the EV and of the charging point have been realized and integrated with the implementations of RiseV2G.
An accurate phase of testing has been carried out to verify the expected results. At the end of the project, the obtained results are:

- The user is able to let the charging process start by just plugging in the electric vehicle (Plug&Charge feature);
- The EV owner can insert the departure time and choose the preferred charging profile between the ones proposed by the charging point, enabling energy flexibility and being able to choose the most sustainable and/or most economical one;
- Stronger data security is implemented, ensuring confidentiality, data integrity, and authentication, with the usage of public and private keys managed by a Public Key Infrastructure (PKI).

6.2 Future developments

Since the ISO15118 standard is recent, and the ISO15118-20 document that adds some key features to the charging process has been released in May, several future developments and improvements can be carried out in the future, such as:

- Adding the possibility to enable the V2G technology, also known as bidirectional power transfer, which gives the user the possibility to make his electric vehicle available as a power bank of the smart grid, receiving a discount on the charging process;
- Adding the communication messages defining the wireless power transfer and the automated connection device;

- Adding the dynamic charging mode: the user can select this charging mode to allow the charging point to automatically change the energy provided during the charging process in order to react to the grid needs.

Bibliography

[1] Sesha Gopal Selvakumar. «Electric and Hybrid Vehicles – A Comprehensive Overview». In: *2021 IEEE 2nd International Conference On Electrical Power and Energy Systems (ICEPES)*. 2021, pp. 1–6 (cit. on pp. 1, 3, 4, 6).

[2] Xiaoli Sun, Zhengguo Li, Xiaolin Wang, and Chengjiang Li. «Technology development of electric vehicles: A review». In: *Energies* 13.1 (2019), p. 90 (cit. on pp. 5–7, 17, 19, 20).

[3] Murat Yilmaz and Philip T. Krein. «Review of Battery Charger Topologies, Charging Power Levels, and Infrastructure for Plug-In Electric and Hybrid Vehicles». In: *IEEE Transactions on Power Electronics* 28.5 (2013), pp. 2151–2169. DOI: 10.1109/TPEL.2012.2212917 (cit. on p. 6).

[4] Ilhami COLAK, Ramazan BAYINDIR, and Seref SAGIROGLU. «The Effects of the Smart Grid System on the National Grids». In: *2020 8th International Conference on Smart Grid (icSmartGrid)*. 2020, pp. 122–126. DOI: 10.1109/icSmartGrid49881.2020.9144891 (cit. on pp. 8, 9).

[5] Zahra Alavikia and Maryam Shabro. «A comprehensive layered approach for implementing internet of things-enabled smart grid: A survey». In: *Digital Communications and Networks* 8.3 (2022), pp. 388–410. ISSN: 2352-8648. DOI: https://doi.org/10.1016/j.dcan.2022.01.002. URL: https://www.sciencedirect.com/science/article/pii/S2352864822000025 (cit. on p. 10).

[6] Noha Mostafa, Haitham Saad Mohamed Ramadan, and Omar Elfarouk. «Renewable energy management in smart grids by using big data analytics and machine learning». In: *Machine Learning with Applications* 9 (2022), p. 100363. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2022.

100363. URL: https://www.sciencedirect.com/science/article/pii/S2666827022000597 (cit. on pp. 10, 12).

[7] Amrita Ghosal and Mauro Conti. «Key Management Systems for Smart Grid Advanced Metering Infrastructure: A Survey». In: *IEEE Communications Surveys & Tutorials* 21.3 (2019), pp. 2831–2848. DOI: 10.1109/COMST.2019.2907650 (cit. on pp. 10, 11).

[8] Shirazul Islam, Atif Iqbal, Mousa Marzband, Irfan Khan, and Abdullah M.A.B. Al-Wahedi. «State-of-the-art vehicle-to-everything mode of operation of electric vehicles and its future perspectives». In: *Renewable and Sustainable Energy Reviews* 166 (2022), p. 112574. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2022.112574. URL: https://www.sciencedirect.com/science/article/pii/S1364032122004701 (cit. on pp. 12, 13, 15, 16).

[9] Chunhua Liu, K. T. Chau, Diyun Wu, and Shuang Gao. «Opportunities and Challenges of Vehicle-to-Home, Vehicle-to-Vehicle, and Vehicle-to-Grid Technologies». In: *Proceedings of the IEEE* 101.11 (2013), pp. 2409–2427. DOI: 10.1109/JPROC.2013.2271951 (cit. on pp. 15, 16).

[10] Ali Bahrami. «EV Charging Definitions, Modes, Levels, Communication Protocols and Applied Standards». In: (Jan. 2020). DOI: 10.13140/RG.2.2.15844.53123/11 (cit. on pp. 17, 20–22, 24).

[11] Marc Mültin. «ISO 15118 as the Enabler of Vehicle-to-Grid Applications». In: *2018 International Conference of Electrical and Electronic Technologies for Automotive*. 2018, pp. 1–6. DOI: 10.23919/EETA.2018.8493213 (cit. on pp. 26, 28, 35–37).

[12] «BS EN ISO 15118-2:2016: Road vehicles — Vehicle-to-Grid Communication Interface - Part 2: Network and application protocol requirements». In: (cit. on pp. 28, 31–33, 49).

[13] André Kaufung. «CharIN Implementation Guide to Plug and Charge in the context of ISO 15118». In: (Mar. 2022). URL: https://www.charin.global/technology/knowledge-base/ (cit. on p. 36).

[14] «BS EN ISO 15118-1:2019: Road vehicles — Vehicle-to-Grid Communication Interface - Part 1: General information and use-case definition». In: (cit. on pp. 39–42, 55, 60, 83, 96).

[15] «Exploring the public key infrastructure for ISO 15118 in the EV charging ecosystem». In: ElaadNL, Arnhem, The Netherlands, Nov. 2018 (cit. on pp. 47, 50).