Privacy-Preserving Architecture for EV Charging and Billing

Master thesis by Dustin Kern
Date of submission: January 10, 2021

1. Review: Prof. Dr. Michael Waidner
2. Review: Prof. Dr. Christoph Krauß
3. Review: Maria Zhdanova
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department

IT Security, M.Sc.

Privacy-Preserving Architecture for EV Charging and Billing

Master thesis by Dustin Kern

1. Review: Prof. Dr. Michael Waidner
2. Review: Prof. Dr. Christoph Krauß
3. Review: Maria Zhdanova

Date of submission: January 10, 2021

Darmstadt

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Dustin Kern, die vorliegende Masterarbeit gemäß §22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 10. Januar 2021

D. Kern

Zusammenfassung

Die Entwicklung und der Einsatz von Technologien im Zusammenhang mit Elektrofahrzeugen bekommen ein großes Interesse von sowohl Wissenschaftlern als auch Industrievertretern und politischen Entscheidungsträgern. Infolgedessen haben sich Elektrofahrzeugtechnologien in den letzten Jahren erheblich weiterentwickelt und die weltweite Verbreitung von Elektrofahrzeugen nimmt stetig zu.

Die bisherigen Entwicklungen im Elektrofahrzeugsektor waren jedoch meist von ökologischen und/oder finanziellen Zielen getrieben und haben die wichtigen Themen Sicherheit und Datenschutz weitgehend vernachlässigt. Die fehlende Berücksichtigung dieser Themen wird besonders bei den Prozessen des Ladens und der Abrechnung von Elektrofahrzeugen deutlich. Aufgrund der hochgradig sicherheits- und datenschutzkritischen Natur dieser Prozesse führt diese Situation zu einem inakzeptablen Risiko für die Nutzer von Elektrofahrzeugen und ist wohl nicht mit dem aktuellen Datenschutzrecht, d.h. mit der Datenschutz-Grundverordnung (DSGVO), konform.

In dieser Arbeit werden gängige, quelloffene Elektrofahrzeug-Ladeprotokolle untersucht und die damit verbundenen personenbezogenen Daten identifiziert. Darüber hinaus wird eine detaillierte Analyse der Sicherheits- und Datenschutzbedrohungen auf Basis der STRIDE (für Sicherheit) und LINDDUN (für Datenschutz) Methoden durchgeführt, die z.B. das hohe Risiko aufzeigt, dass ein Angreifer in der Lage ist, Bewegungsprofile von Elektrofahrzeugnutzern zu erstellen. Um den identifizierten Bedrohungen zu begegnen, wird in dieser Arbeit eine datenschutzfreundliche Architektur für das Laden und Abrechnen von Elektrofahrzeugen entworfen. Die vorgeschlagene Architektur zielt darauf ab, die Sicherheit der Zahlungsdaten eines Elektrofahrzeugnutzers auf der Basis von Trusted-Computing-Methoden zu schützen sowie die Privatsphäre der Benutzer auf der Basis eines Konzepts für unverknüpfbare Ladeautorisierungen zu schützen. Die Architektur ist so konzipiert, dass sie ihren Schutz auch unter Berücksichtigung von mächtigen Angreifern mit physischem Zugang zu Steuergeräten sowie neugierigen Betreibern gewährleistet und gleichzeitig mit den bestehenden Definitionen von Rollen und Prozessen beim Laden von Elektrofahrzeugen weitestgehend kompatibel ist.

Die Architektur wird als Proof-of-Concept implementiert, um ihre Machbarkeit zu zeigen, und wird im Hinblick auf die identifizierten Bedrohungen evaluiert. Die Evaluierung zeigt die Angemessenheit der Lösung für den Anwendungsfall, den hohen Grad an Kompatibilität zu den aktuellen Elektrofahrzeug-Ladeprotokollen und das hohe Maß an Sicherheit und Datenschutz, das die Lösung bieten kann. Die vorgeschlagene Architektur wird, insbesondere unter Berücksichtigung der strengen Bestimmungen der DSGVO, als ideale Lösung für den Schutz des Ladens und der Abrechnung von Elektrofahrzeugen bewertet.

Abstract

The development and deployment of Electric Vehicle (EV) technologies is receiving a great deal of attention from the scientific community, industry representatives, and policy-makers alike. As a result, EV technologies have advanced considerably over the past years and the global adoption rate of EVs is steadily increasing.

The past developments in the EV sector, however, were mostly driven by environmental and/or financial goals and have largely neglected the important topics of security and privacy. The lacking consideration of these topics is especially obvious in the processes of EV charging and billing. Due to the highly security- and privacy sensitive nature of these processes, this situation results in an unacceptable level of risk to EV users and is arguably not compliant to contemporary data protection law, i.e., the General Data Protection Regulation (GDPR).

This thesis assesses popular, open source EV charging protocols and identifies the involved personal data. Furthermore, a detailed security- and privacy threat analysis is conducted based on the STRIDE (for security) and LINDDUN (for privacy) methodologies showing, for instance, the high risk of an adversary being able to build movement profiles of EV users. In order to address the identified threats, this thesis propose a privacy-preserving architecture for the charging and billing of EVs. The proposed architecture aims to protect the security of an EV user's payment credentials based on trusted computing methods as well as protect the privacy of users based on a concept for unlinkable charge authorizations. The architecture is designed to provide its protections even under consideration of powerful physical-access adversaries and curious operators while being compatible with the existing definitions of roles and processes in EV charging to the fullest extent possible.

The architecture is implemented as a proof-of-concept to show its feasibility and evaluated with respect to the identified threats. The evaluation shows the appropriateness of the solution for the use case, its high degree of compatibility to the current EV charging protocols, and the high level of security- and privacy protections it can provide. The proposed architecture is argued to be an ideal candidate for protecting the charging and billing of EVs especially under consideration of the GDPR's strict provisions.

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

AES Advanced Encryption Standard
AIK Attestation Identity Key
CA Certificate Authority
CCH Contract Clearing House
CDR Charge Detail Record
CFB Cipher Feedback Mode
CS Charge Station
CSO Charge Station Operator
CPS Certificate Provisioning Service
CSR Certificate Signing Request
DAA Direct Anonymous Attestation
DER Distributed Energy Resources
DFD Data Flow Diagram
DHCPv6 Dynamic Host Control Protocol version 6
DPIA Data Protection Impact Assessment
DSO Distribution System Operator
ECC Elliptic Curve Cryptography
ECDH Elliptic Curve Diffie Hellman
ECDSA Elliptic Curve Digital Signature Algorithm
ECU Electronic Control Unit
EIM External Identification Means
EK Endorsement Key
eMAID e-Mobility Account Identifier
eMIP eMobility Interoperation Protocol
EMS Energy Management System
eMSP e-Mobility Service Provider
EV Electric Vehicle
EVCC Electric Vehicle Communication Controller
EVSE Electric Vehicle Supply Equipment
EXI Efficient XML Interchange
FTP File Transfer Protocol
GCM Galois/Counter Mode
GDPR General Data Protection Regulation
HMAC Hash-based Message Authentication Code
HTTP Hypertext Transfer Protocol
ICMPv6 Internet Control Message Protocol version 6
IMSI International Mobile Subscriber Identity
IOI Item Of Interest
IoT Internet of Things
IP Internet Protocol
IV Initialization Vector
JSON JavaScript Object Notation
KDF Key Derivation Function
LAN Local Area Network

LC Local Controller
LP Local Proxy
MAC Media Access Control
MitM Man-in-the-Middle
ND Neighbor Discovery
NV Non-Volatile
OEM Original Equipment Manufacturer
OCA Open Charge Alliance
OCHP Open Clearing House Protocol
OCPI Open Charge Point Interface
OCPP Open Charge Point Protocol
OCSP Online Certificate Status Protocol
OICP Open InterCharge Protocol
OID Object Identifier
PCR Platform Configuration Register
PCID Provisioning Certificate Identifier
PET Privacy-Enhancing Technology
PII Personally Identifiable Information
PKI Public Key Infrastructure
PLC Power Line Communication
PnC Plug-and-Charge
RFID Radio Frequency Identification
RSA Rivest-Shamir-Adleman
SDL Secure Development Lifecycle
SDP SECC Discovery Protocol
SECC Supply Equipment Communication Controller
SHA Secure Hash Algorithm
SIA Subject Information Access
SIM Subscriber Identification Module
SLAAC Stateless Auto Address Configuration
SOAP Simple Object Access Protocol
TCG Trusted Computing Group
TCP Transmission Control Protocol
TLS Transport Layer Security
TPM Trusted Platform Module
UDP User Datagram Protocol
UID Unique Identifier
URL Uniform Resource Locator
V2G Vehicle to Grid
V2GTP V2G Transfer Protocol
VAS Value-Added Services
VPN Virtual Private Network
VM Virtual Machine
WAN Wide Area Network

1. Introduction

The adoption of Electric Vehicles (EVs) is steadily increasing worldwide. EVs provide many societal benefits over traditional combustion engine vehicles, such as reducing the urban air pollution, combating the threats of climate change, and decreasing the dependency on fossil fuels [189]. With these benefits, EVs are a driving force in the development of a cleaner urban transportation sector and can aid in the process of achieving environmental objectives [165]. For these reasons, EVs receive a great deal of support from the scientific community, industry representatives, and policy-makers alike, pushing the technological advance of EVs as well as the public interest in EVs [94]. In fact, the global number of EVs had already reached 5.1 million in 2018 [91], grew to 7.2 million in 2019 [92], and is expected to increase to a total of 140 to 245 million by the year 2030 [92].

1.1. Context

EVs are “refueled” by charging them at a Charge Station (CS). However, this process also involves a multitude of other actors besides the EV and CS (cf. Fig 1.1). For one, the EV user typically has a contract with an e-Mobility Service Provider (eMSP). The eMSP provides their customers with services related to EV operation, e.g., the authorization and billing of a charging process. Second, a Charge Station Operator (CSO) (who may also act as an eMSP) typically operates the CSs. In order for an EV user to charge their EV at a CS of this user’s eMSP/CSO, authorization and billing does not involve any further actors. However, in order to also enable charging at other CSs, eMSPs can have roaming contracts with different CSOs. In the roaming case, charge authorization and billing is usually handled via another actor, the Contract Clearing House (CCH), as intermediary between a multitude of CSOs and eMSPs.

A prerequisite for charging is the credibility of the EV user. For contract-based charging, this is achieved by identifying the user as the customer of an eMSP. This identification serves as basis for the authorization of the charging process and the billing of the user for their consumed energy. Currently, the process for charge

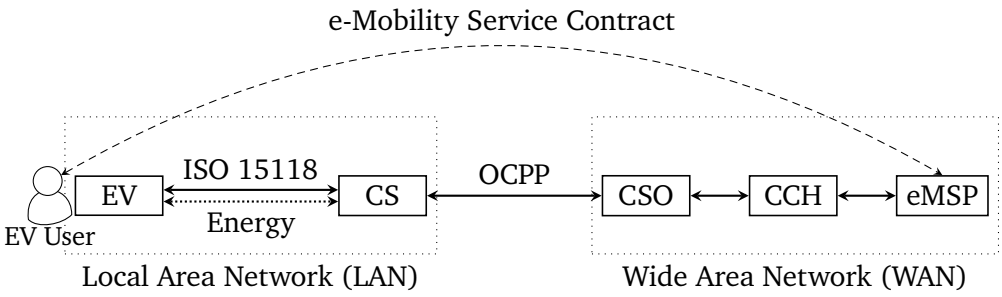


Figure 1.1.: Overview of the EV Charging Infrastructure

authorization and billing is mostly realized with proprietary and incompatible solutions (e.g., with a Radio Frequency Identification (RFID) card at the CS or via a smartphone app over the eMSP), possibly leading to an inconvenient user experience [190]. To alleviate this problem, many standardization efforts have been made. For instance, ISO 15118 [96, 97] defines a Vehicle to Grid (V2G) communication interface between an EV and a CS that supports the automatic authentication and charge authorization based on the user's e-Mobility Service Contract without requiring user interaction. For this automatic charge authorization, called Plug-and-Charge (PnC), the EV is provided with contract credentials (an asymmetric key pair and a public key certificate) from the eMSP that are used at a CS to authenticate the corresponding EV user as a customer of the eMSP. Additionally, Open Charge Point Protocol (OCPP) [145, 148] is the de facto standard protocol for communication between CS and CSO, and several competing protocols exist for the communication between CSOs, CCHs, and eMSPs [50].

1.2. Motivation

As outlined by the multitude of actors and communication protocols (cf. Fig 1.1), EV charging does not only involve the transfer of electric energy, but also the transfer of a variety of data a lot of which is (directly or indirectly) privacy sensitive [119]. For example, as EV users need to be individually billed, even when charging on third party sites (i.e., the roaming case), identifiers need to be transferred together with the amount of consumed energy. Additionally, existing communication protocols also identify the charge location, -duration, and further user preferences. This data could be used to infer consumer habits. Moreover, the transferred data is shared between a variety of different actors and across legal boundaries [162]. Combined with the very frequent charging of EVs (possibly more than 3 times a day [176]) and the desirable high density of CSs in public places [183], this situation creates huge privacy risks for EV users as all backend actors could use this data to build detailed movement profiles as well as habitual patters.

These risks are to be taken very seriously, especially in consideration of contemporary data protection law [112, 201], specifically the EU's General Data Protection Regulation (GDPR) [58] effective since 25 May 2018. For instance, an EV user's identifier (and any information relating to it) is considered *personal data* under the GDPR [58] Art. 4(1). As such, this data is subject to the GDPR data protection principles (e.g., "*lawfulness, fairness and transparency*," "*purpose limitation*," and "*data minimisation*") [58] Art. 5(1). Furthermore, this data may only be processed if a condition from [58] Art. 6(1) applies (e.g., the data is required in performance of a contact like the e-Mobility Service Contract; cf. Fig 1.1), and it requires appropriate (i.e., in regard to the state of the art, cost, and risk) protection measures by design and by default [58] Art. 25.

However, existing communication protocols that are involved in EV charging are not designed with adequate privacy preserving measures. In the current state, respective protocols often simply transfer a static identifier of the EV user's e-Mobility Service Contract (which serves as pseudonym of the user) along with the timestamps, location, and consumed energy of the charging session to actors that do not require this data for the performance of their services. The pseudonym of the user can be used to link multiple charge sessions of the same user and thus enables anyone in possession of this data to build a movement profile of the user. The possibility to build these movement profiles is a serious privacy concern as movement profiles are often shown to uniquely identify an individual [39] and can enable the inference of sensitive data like an individual's political views or state of health [48]. Furthermore, this information could be used in order to carry out real world attacks as, for instance, robbing an individual's home in their absence [65]. This shows the existing high privacy risks, the inadequate consideration of data protection principles, and the lacking protection measures that exist in

the current EV charging infrastructure. These circumstances are arguably not GDPR compliant and certainly not in line with the objectives of the GDPR.

1.3. Contribution

The goal for this thesis is the development of privacy preserving measures for the EV charging infrastructure (including EV, CS, and backend), building on existing definitions of open source protocols and roles. Our focus lies on the processes for contract-based charge authorization and billing as these processes are the most critical with regard to user privacy. These processes must be adjusted in order to protect the user's privacy by providing anonymity of the EV driver whenever possible and thus providing the unlinkability of charging sessions to the highest possible extent (e.g., regarding CS, CSO, and CCH). Additionally, since we are concerned with a payment/billing use case, the involved processes should offer a high level of security and the payment credentials of EV users should receive a high level of protection throughout their entire life-cycle. Furthermore, any security-/privacy solution should stay compatible to the existing definitions of actors, processes, and protocols to the highest possible extend since the respective charging infrastructure is already being deployed and not all of its components are easily upgradeable.

The solution is designed under the presence of an adversary with not only access to the communication channel (i.e., the Dolev-Yao adversary model [47]) but also with physical access to the relevant hardware components (e.g., the Electronic Control Units (ECUs) of EV and CS) as these components are left unattended in public places for long periods. For example, EVs typically do not employ any technical safeguards to protect their locally stored data [162]; hence, it would be easy for an adversary with physical access to extract privacy sensitive and/or security relevant data from an EV's ECUs. Furthermore, the potential misuse of an EV user's personal data by the different backend actors must be considered due to the privacy-sensitive nature of this data and the previously described opaque involvement of a variety of actors.

The contributions of this thesis can be summarized as follows:

- An evaluation of the relevant protocols, roles, and processes in the current EV charging infrastructure is provided. This evaluation highlights the relevant scenarios and identifies the personal data that should be considered by a possible solution.
- An extensive security- and privacy threat analysis of the current EV charging protocols is performed. The analysis is based on the widely used STRIDE (for security) and LINDDUN (for privacy) methodologies and considers the previously described diverse group of possible adversaries in order to provide the most accurate reflection of possible threats.
- An architecture is presented that secures the processes of charge authorization and billing. The architecture protects the EV user's location privacy and it provides for the unlinkability of a user's charge session data. Furthermore, the architecture offers support for all relevant processes of the current EV charging infrastructure. More precisely, it the proposed architecture provides a privacy-preserving authorization mechanism that supports the existing concepts of charge authorization, including local authorization via RFID cads or using the PnC mechanism as well as remote authorization (e.g., via a smartphone app) via the user's eMSP and remote CS reservations. Additionally, the proposed architecture supports the existing methods of charge authorization, i.e., online authorization (the CS verifies the user's charge authorization status via a request to the backend) and offline authorization (the CS verifies the user's charge authorization status via a locally available whitelist). Furthermore, the integration of a Trusted

Platform Module (TPM)-based Direct Anonymous Attestation (DAA) scheme into EV charging protocols as a privacy-preserving replacement for the existing PnC authentication is presented.

- The proposed architecture is integrated into existing EV charging protocols as a proof-of-concept implementation, showing that an integration is possible with minimal changes to message definitions and message handling, while maintaining the general data flow and role definitions. The proof-of-concept implementation is used to evaluate the overhead of the proposed architecture, showing that the integration of the DAA scheme incurs only minimal overhead. Similarly, the overhead of the privacy-preserving charge authorization mechanism in the scope of online authorization is shown to be minimal. Furthermore, while the overhead that is required to enable the privacy-preserving offline authorization (the distribution of authorization whitelists) is shown to be high, it is argued to be manageable by in the scope of backend systems.
- The security and privacy properties of the architecture are evaluated under consideration of the previously described diverse group of adversaries. The proposed architecture is argued to adequately address the identified security and privacy threats of EV charging protocols.

1.4. Structure of the Thesis

This thesis is structured as follows: Section 2 presents the relevant background. Section 3 provides an overview of related work. Section 4 presents the considered system model. In Section 5, the considered adversary model is discussed and the STRIDE-/LINDDUN-based threat analysis is conducted. Section 6 discusses the security-, privacy-, and functional requirements that a solution must provide in order to address the analyzed threats. Section 7 presents the privacy-preserving charge authorization concept. Section 8 describes the proof-of-concept implementation and the respective overhead measurements. In Section 9, the presented concept is evaluated with regard to the security-, privacy-, and functional requirements. Section 10 provides concluding remarks.

2. Background

The following sections provide the relevant background for this thesis. Section 2.1 provides a brief background on privacy focusing on Personally Identifiable Information (PII) and the risk of re-identification attacks on de-identified data. Since the strict provisions of the GDPR are one of the main motivations for the protection of an EV user's personal data a high-level overview of the GDPR and its application to the EV charging domain are presented in Section 2.2. Section 2.3 describes the threat modeling methodologies that are used to analyze the existing security and privacy threats in the EV charging infrastructure. Afterwards, Section 2.4 provides details on the existing definitions of actors, processes, and protocols in the current EV charging infrastructure. Finally, Section 2.5 offers an overview of the functionalities of a TPM that are used in the proposed architecture in order to secure the processes of EV charging and billing, including DAA and existing methods for the integration of a TPM into EV charging.

2.1. Privacy

The concept of privacy is extensive with several different interpretations depending on its context [181]. Moreover, it has been significantly affected by technological advances, which introduce new potential privacy harms [180]. As a result, several different definitions of privacy have been formulated over the years.

Warren and Brandeis, who are often cited as the most important figures in the history of the legal right to privacy [117, 198], defined privacy as "the right to be let alone" [203]. Westin notably defined privacy as "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [206]. Westin's definition of privacy with a focus on an individual's control over their information is also the basis for most modern data privacy principles and laws [37]. Nissenbaum further introduced the notion of "contextual integrity" in order to address the increasing problem of public surveillance resulting from technological developments in the areas of information gathering, analysis, and dissemination [141]. For the protection of privacy, contextual integrity requires information gathering to be appropriate for the context (e.g., sharing ones physical condition with a physician or financial information with a creditor but not sharing religious affiliation with ones employer) and information distribution to respect contextual norms (e.g., of discretion, confidentiality, need, entitlement, and obligation).

2.1.1. Personally Identifiable Information

PII (also referred to as personal data) is one of the most central concept in the field of privacy as it is used to define the scope and boundary of most privacy regulations [170]. Non-PII are left unregulated based on the basic assumption that without PII there are no privacy risks [170]. While several different definitions of PII exist, PII is commonly considered to not only incorporate data that directly identifies an individual but also

indirect identifiers, i.e., data that can reasonably likely be used to identify an individual, taking account for the possibility of deductive disclosure [136]. Examples of PII include an individual's names, pseudonyms of an individual or of a small, well-defined group of individuals (e.g., an Internet Protocol (IP)- or Media Access Control (MAC) address), personal characteristics (e.g., photographic images or biometric data), information that identifies personally owned property (e.g., a vehicle identification number), and information about an individual that is linked or linkable to one of the previously mentioned PII [129]. The "Safe Harbor" method [199] of the U.S. Health Insurance Portability and Account-ability Act [200] provides a similar list of PII including names, various kinds of pseudonyms, device identifiers, geographic subdivisions smaller than a state, as well as dates that are directly related to an individual.

2.1.2. De-Identification

A commonly used method for addressing privacy risks when working with personal information is data de-identification [49, 166]. By de-identifying data, i.e., by removing all identifying information from it in order to prevent linkability to specific individuals, the privacy risk of data processing can be reduced at the cost of reduced data utility [72]. In [72], an overview of de-identification approaches is provided, including the following:

- Removal of direct identifiers such as names, email addresses, and telephone numbers.
- De-identification of quasi-identifiers (or indirect identifiers), i.e., of data that when taken together and linked with other information can be used to identify a specific individual. One of the most prominent example of quasi-identifiers is the combination of ZIP code, birth date, and gender, which was found by Sweeney to uniquely identify 87.1% of the U.S. population [187].[1] Quasi-identifiers can be de-identified by, e.g., suppression, generalization, or perturbation.
- Pseudonymization, i.e., the systematic replacement of direct identifiers with pseudonyms. The use of pseudonyms can allow the linking of an individuals personal information across data records and may be reversed if the link between pseudonym and direct identifier is preserved. Additionally, the use of a unique pseudonym over an extended period of time results in an increasing risk to privacy as it enables the linkability of an increasing amount of information.

  In [161], Pfitzmann and Hanse propose a distinction between different kinds of pseudonyms based on the context of their usage into the following classes:

  Person pseudonym: A pseudonym is used to replace an individual's real name. Actions of the same user can always be linked.

  Role pseudonym: A pseudonym is assigned for a specific role, e.g., a customer pseudonym. Actions under the same role pseudonym can be linked.

  Relationship pseudonym: A separate pseudonym is used for each communication partner. Actions concerning the same communication partner can be linked.

  Role-relationship pseudonym: A separate pseudonym is used for each role and each communication partner. Actions concerning the same combination of role and communication partner can be linked.

[1] A more recent study, while generally in agreement with the findings of Sweeney, places the number individuals in the U.S. population that are identified by a combination of ZIP code, birth date, and gender at 63.3% [76].

Transaction pseudonym: A separate pseudonym is used for each transaction. Only actions concerning the same transaction can be linked.

The concept of anonymization is closely linked to de-identification. One of the most commonly used definitions of anonymity is provided by Pfitzmann and Hansen who define the anonymity of a subject as "the subject is not identifiable within a set of subjects, the anonymity set" or from the perspective of an adversary as "the attacker cannot sufficiently identify the subject within a set of subjects, the anonymity set" [161]. In the context of de-identification, anonymization is often refereed to as the de-identification of PII until the resulting data cannot be linked back to the corresponding individual [139].

2.1.3. Re-Identification

While de-identification is generally considered to reduce privacy risks it cannot completely eliminate them, i.e., anonymize the data, without an unacceptable reduction in data utility [72]. This limitation to the effectiveness of de-identification is shown by many successful re-identification attempts on released data sets that was believed to be de-identified [170, 136, 79]. Some of the most prominent examples of re-identification are:

- In the late 1990s the Group Insurance Commission, a government agency in Massachusetts, released medical records of state employees to researchers [14]. While the medical records were stripped of all direct identifiers, they still contained nearly one hundred attributes per record including sensitive medical data (diagnosis, medication, etc.) along with the respective patients ZIP code, birth date, and gender [186]. Sweeney demonstrated the possibility of re-identifying specific individuals in these medical records by identifying the records of William Weld who was the governor of Massachusetts at that time [186]. Re-identification was possible by linking the medical records to the voter registration list for Cambridge Massachusetts, which contains the names of individuals, based on the tuple of ZIP code, birth date, and gender [186].
- In 2006, America Online publicly released twenty million search queries from users of its search engine [153]. The data was de-identified by replacing direct identifiers like usernames and IP addresses with a pseudonym that allows linking different queries of the same user [153]. However, only a few days after the data had been released, reporters of the *New York Times* were able to re-identify one of the pseudonymized individuals purely based on the contents of their search queries [10].
- Later in 2006, Netflix released a data set of over one hundred million records that revealed how their customers had rated different movies [140, 16]. The data was released as part of their "Netflix Prize" data mining competition and had been de-identified by replacing direct identifiers with a pseudonym that allows linking different ratings of the same customer [140, 16]. In [137], however, Narayanan and Shmatikov demonstrate a re-identification attack on the Netflix data set. Narayanan and Shmatikov start by presenting a robust re-identification attack against high-dimensional micro-data – i.e., information about specific individuals that is high-dimensional (each record contains many attributes) and sparse (the average record is unique in its attributes) – such as individual preferences, recommendations, and transaction records. Narayanan and Shmatikov use their developed technique to show that an adversary with only a small amount of background knowledge about an individual can use this knowledge to re-identify this individual in the Netflix data set. For instance, they find that 68% of the records can be uniquely identified if an adversary knows only two ratings of an individual, along with the corresponding date (with a 3-day error). Additionally, Narayanan and Shmatikov demonstrate the practicability of their re-identification method by using public Internet Movie Database[2] ratings (often created under an

[2] `https://www.imdb.com/` (visited on 2020-11-24)

individuals real name) as a source of background knowledge to re-identify users in the Netflix data set. The authors claim that they were able to infer particularly sensitive information about the re-identified individuals based on their non-public Netflix viewing history that cannot be deduced based on their public Internet Movie Database ratings. This sensitive information includes indicators of the individuals' political positions, religious views, and sexual orientation.

- In [39] de Montjoye et al., analyze 15 months of mobility data from one and a half million individuals. The data was collected based on mobile phone cellular connections, i.e., the spacial resolution depends on the density of antennas, has a temporal resolution of one hour, and was de-identified by pseudonymization (allowing the linking of an individual's records to build mobility traces). The authors find that 95% of individuals can be re-identified based on their mobility traces by an adversary with background knowledge of only four random spatio-temporal points, whereas two random points are already enough to identify more than 50% of individuals. Additionally, the authors point out that even lowering the spatial and temporal resolution of the mobility data is not enough to protect privacy. For example, with a temporal resolution of 5 hours and clustering antennas into groups of five, four random points are enough to still identify more than 50% of individuals. The authors conclude that mobility traces are highly unique and can be linked to an individual based on little outside information (e.g., using any public information such as geo-localized tweets or -pictures).
- In [40] de Montjoye et al., analyze 3 months of credit card transactions for 1.1 million individuals. The transaction records were de-identified by pseudonymization (allowing the linking of an individuals records) and include a timestamp with a resolution of 1 day, the associated shop, as well as the price. The authors demonstrate that 90% of individuals can be re-identified based on their financial traces by an adversary with background knowledge of only four random spatio-temporal points, i.e., the day and location of four purchases. Additionally, they show that if an adversary knows the approximate price of transactions (with a precision of 0.5), the probability of re-identification increases by 22%. Furthermore, the authors demonstrate that even a generalization of financial data is not enough to protect privacy. For instance, at a timestamp resolution of 15 days, with shops aggregated into clusters of size 350, and with knowledge of approximate prices – while the probability of re-identification with four known spatio-temporal points is less than 15% – the probability of re-identification with 10 known points is still more than 80%. The authors conclude that any high-dimensional sparse data that are generated as an effect of human interaction (e.g., mobile phone-, web browsing-, and transportation data sets) are likely to be re-identifiable as long as only simple de-identification methods are used.

While the identifiability of individuals based on location data, as demonstrated by de Montjoye et al. in [39], relies on the linkability of an individual's records based on pseudonyms, linkability can also be achieved with completely anonymous records if sufficient contextual information is available [184, 185]. For instance, regarding the EV charging use case, in [184] Stegelmann and Kesdogan demonstrate an algorithm to reduce the anonymity set size of anonymous charging events (knowing only location and time) solely based on vehicle travel times. In [185], the same authors present a possibility of enhancing the approach from [184], potentially enabling the linkability of anonymous charging events, based on additional knowledge about charging parameters – i.e., the vehicle's state-of-charge, the required amount of energy, and the user's intended departure time – as this information is commonly available to backend actors for load balancing. If enough charging events can be linked to the same EV user, this information likely serves as a quasi identifier of the user as shown by de Montjoye et al. in [39], as well as many other papers (e.g., [69, 28, 41, 63]), and this information may be used to re-identify the user by correlation with publicly available data [39, 28, 77, 208].

The consequences of a failure to protect the individuals location privacy can be substantial and range from unsolicited marketing over intrusive inference possibilities (e.g., inferring an individual's political views, state

of health, or personal preferences) all the way to threats to personal well-being and safety [48]. Specifically, in the EV charging context, a historic profile of EV charging parameters in combination with the associated charging location and an EV (user) ID is argued to allow the inference of a variety of additional personal data, including indications of personal wealth, the times of leaving/returning home, the distance to places of travel, the location of an individual's residence, the identities of friends, the doctors/hospitals an individual visits, the individual's shopping preferences, as well as the locations of businesses an individual visits [74]. The potential consequences of a severe privacy breach are further demonstrated by ensuing underground business models – including cyberstalking, cybercasing (i.e., using the information to carry out real world attacks like burglaries), enabling subsequent attacks (e.g., phishing or social engineering), economic profiling, and targeted espionage – and highlight the importance of comprehensive privacy protection [64].

2.1.4. Privacy-Enhancing Technologies

In order to protect an individual's privacy, a large amount of Privacy-Enhancing Technologies (PETs) has been proposed over the years [171]. The authors of [171] present an overview of possible PET functions ranging from providing (at varying degrees) anonymous or pseudonymous communication, over providing identity management with minimum identity disclosure, over enabling data mining while preventing intrusions to the privacy of affected individuals, to verifying compliance with privacy policies. In [80], Heurix et al. propose a taxonomy of the classification of PETs. The taxonomy defines the primary dimensions of PETs as follows:

Scenario: The scenario dimension describes the involved actors and categorizes them based on how trustworthy they are. Heurix et al. consider an untrusted client, an untrusted server, a scenario of mutual distrust between the different actors, and a scenario with an external adversary.

Aspect: The aspect dimension describes, which aspect of privacy the PET is trying to address. Heurix et al. consider the privacy aspects of concealing the identities of involved actors, hiding the processed or created data content, and hiding the behavior of actors.

Aim: The aim dimension describes a PET's aim for or means of privacy protection. Heurix et al. consider the possible aims/means of indistinguishability, unlinkability, deniability, and confidentiality.

Foundation: The foundation dimension describes the security model (computational or information-theoretical) and cryptographic foundation of a PET. Heurix et al. list the types of cryptographic foundation as symmetric, asymmetric, unkeyed and non-cryptographic.

Data: The data dimension describes, which types of data a PET addresses. Heurix et al. distinguish between stored, transmitted, and processed data.

Trusted Third Party: The trusted third party dimension describes a PET's requirement for and involvement of trusted third parties. Heurix et al. list the criteria as frequency of involvement, phase of involvement, and task of the trusted third party.

Reversibility: The reversibility dimension describes if and how a PET is reversible. Heurix et al. make distinctions by whether or not reversibility requires the cooperation of the data subject and in the possible degree of reversibility (e.g., full or partial).

2.2. General Data Protection Regulation

The EU's GDPR [58] replaces the previous data protection directive 95/46/EC [57]. It came into force on 24 May 2016 and is applicable since the end of a two-year transition period on 25 May 2018. The GDPR provides unified and comprehensive provisions for the effective enforcement of the fundamental right for protection of personal data. In doing so, the GDPR provides privacy related definitions, principles, rights, and obligations, which are relevant for the EV charging use case. This section summarizes the GDPR's provisions with relevance to the thesis.

2.2.1. Personal Data

In the context of the GDPR, personal data is defined as "any information relating to an identified or identifiable natural person ('data subject')" whereby identifiable also covers the indirect identification via, inter alia, an identification number (cf. [58] Art. 4(1)). With regard to personal data, the GDPR defines the roles of "controller" and "processor." The "controller" is an actor that "determines the purposes and means of the processing of personal data" (cf. [58] Art. 4(7)) and the "processor" is an actor that "processes personal data on behalf of the controller" (cf. [58] Art. 4(8)).

Due to its broad definition of personal data, the GDPR covers a large number of use cases. For instance, even pseudonymized data constitutes personal data and is protected by the GDPR (cf. [58] Recital 26), which, furthermore, can also be true in the case where only a third party has access to the additional knowledge that is required for re-identification (cf. the Breyer case[3] [104]). Moreover, based on the opinion of the Article 29 Data Protection Working Party,[4] even the risk of a security breach should be taken into consideration when assessing *identifiability* of a natural person [6], as a part of "all the means reasonably likely to be used, such as singling out, either by the controller or by another person to identify the natural person directly or indirectly" as defined in [58] Recital 26. Additionally, the risk of re-identification attacks (inter alia, the ones described in Section 2.1.3) should be considered [5].

Additionally, for an application to the EV charging use case, the concept of "relating to" needs to be defined more precisely. The opinion of the Article 29 Data Protection Working Party[4] provides a detailed insight into the possible interpretation of the "relating to" concept (cf. [6]). Based on this, information could be considered to relate to a natural person based on a "content" element (i.e., the information is *about* that person, regardless of the purpose of the controller or the impact on the person), or a "purpose" element (i.e., the information is (likely) used with the *purpose* to "evaluate, treat in a certain way or influence the status or behaviour of" that person), or a "result" element (i.e., the information is used in a way that has an *impact* on the rights and interests of that person). As pointed out in [164], the Working Parties interpretation encourages

[3] *Patrick Breyer v Bundesrepublik Deutschland*, wherein the judgment of the Court of Justice of the European Union states that the dynamic IP address of a client, stored by a host when a person accesses their website, constitutes personal data under [57] Art. 2(a). This judgment was based on the reasonable likelihood of re-identification of the client, resulting from the legal channels provided to the host allowing them to contact a competent authority, which, in the context of criminal proceedings, can request the required data for re-identification from an internet service provider. While the judgment relates to directive 95/46/EC, it's definition of personal data ("'personal data' shall mean any information relating to an identified or identifiable natural person ('data subject')"; cf. [57] Art. 2(a)) is effectively the same as the GDPR's and, hence, the interpretation can still be considered valid.

[4] The Working Party was an independent advisory body set up under [57] Art. 29 and is comparable to the new European Data Protection Board set up under [58] Art. 68. As such, their opinion can be considered highly relevant even though it is not legally binding. While the opinion of the Working Party relates to directive 95/46/EC, it's definition of personal data is effectively the same as the GDPR's and, hence, the interpretation can still be considered relevant.

the classification of any information generated by (observing) people (e.g., online behavior) as well as any information generated by objects that the people interact with (e.g., their vehicles) as personal data. This interpretation is also in line with research indicating that any high-dimensional and sparse information about individuals can likely be used for re-identification (cf. [137, 40] as mentioned in Section 2.1.3).

A broad view of personal data is also adopted by the European Data Protection Board in their current version for guidelines relating to the processing of personal data in the context of connected vehicles and mobility related applications [19]. The Board argues that, inter alia, details of the journeys made, vehicle usage data (e.g., relating to the distance covered), technical data of the vehicle (e.g., relating to movement or condition), and even metadata (e.g., maintenance status) relate to an indirectly identifiable person and as such, are considered personal data and protected by the GDPR. Furthermore, the Board identified categories of vehicle data, which due to their sensitivity and/or potential impact warrant special attention:

Location data: Data about the geolocation of a vehicle could potentially reveal the place of work, the residence, the religion, or sexual orientation of the user, based on the places that they visited. For this reason, it should only be collected if it is required for the use case, should not be collected more frequently than necessary (even with the user's consent), and, if processing is based on consent, that consent should be separate from the general conditions of use.

Biometric data: This category of data may, for example, be used to authenticate the user. It should only be stored and processed locally and with appropriate security measures and appropriate non-biometric alternatives should be provided.

Data that could reveal offenses or traffic violations: An example for this kind of data is the vehicle's speed combined with its current location as it could reveal that the driver is breaking the speed limit. Offense-related data may only be processed under the control of official authority and thus any data that is potentially offense-related (e.g., the vehicle's speed, which could become offense-related if it is combined with other data) should only be processed locally and with strong security measures.

2.2.2. Data Protection Principles

The GDPR's core provisions relating to the handling of personal data are outlined in its data protection principles. These principles must be adhered to whenever personal data is processed and failure to comply may result in a fine of up to 20,000,000 EUR or of up to 4% of an organization's worldwide annual turnover ([58] Art. 83(5)). Note that the GDPR adopts a very broad view of the term "processing" as any operation on personal data including, inter alia, collecting, storing, altering, using, and disclosing (cf. [58] Art. 4(2)). The GDPR's data protection principles are as follows (cf. [58] Art. 5(1)):

Lawfulness, fairness and transparency: All personal data must be processed lawfully, fairly and transparent to the data subject.

The principle of *lawfulness* is further defined in [58] Art. 6 and can be achieved with multiple conditions, inter alia, if the data subject consented to the processing of their personal data, if processing is required for performance of a contract with the data subject, or if a legitimate interest of the data controller requires this processing.[5]

[5] Note that the European Data Protection Board [19] argues that a connected vehicle constitutes "terminal equipment" under directive 2008/63/CE [34] Art. 1(a). Thus, it is affected by provisions of Art. 5(3) from the ePrivacy directive [56], stipulating that prior consent from the user in accordance with the GDPR is required before storing or assessing any information on the terminal unless this information is required for communication or for an explicitly requested information society service. Any further processing

The conditions for *consent* as a basis for *lawfulness* are further defined in [58] Art. 7 and include: (i) a controller must be able to demonstrate that the data subject has consented, (ii) the request for consent bust be clearly distinguishable from other matters and in plain language, (iii) the data subject has the right to withdraw their consent at any time, and (iv) consent should not be valid if the performance of a contract is conditioned on the consent to the processing of personal data even though the respective personal data is not necessary for the performance of this contract (cf. [58] Art. 7(4) and [58] Recital 43).

Purpose limitation: All personal data may only be processed in a manor that is compatible to the purpose of their initial collection.

Data minimisation: All personal data must be adequate, relevant and limited to what is absolutely necessary for the purpose of processing. The European Data Protection Board emphasizes this point in relation to the location data of a connected vehicle due to the particularly intrusive nature of this data category [19].

Accuracy: All personal data must be kept accurate and up to date. Inaccurate personal data must be deleted or corrected as soon as possible.

Storage limitation: All personal data may only be stored in a form that allows the identification of the data subject for as long as this form is required for the purposes of processing.

Integrity and confidentiality: All personal data must be protected by appropriate security measures in order to prevent their unauthorized processing as well as accidental loss.

In [58] Art. 32(1) it is further clarified that data controllers and processors must implement risk-appropriate security measures, such as pseudonymization and encryption of personal data as well as measures to ensure confidentiality, integrity, availability and resilience of processing. Furthermore, [58] Art. 32(2) stipulates that even the privacy risks from a potential security breach should be considered when assessing the appropriateness of security measures.

2.2.3. Data Protection by Design and by Default

A further pillar of the GDPR is defined by its notion of data protection by design and by default (cf. [58] Art. 25). It requires a data controller to implement appropriate measures ensuring the adherence to the data protection principles of [58] Art. 5; whereby appropriateness depends on the state of the art, the cost of implementation, and the involved privacy risks. Furthermore the controller needs to ensure that these measures are effective by default and are considered during the systems design (when determining the means of data processing) as well as during the systems operation (when processing the data). Failure to comply with the requirement of data protection by design and by default may result in a fine of up to 10,000,000 EUR or of up to 2% of an organization's worldwide annual turnover ([58] Art. 83(4)).

Further details on the meaning of "data protection by design and by default" are provided by the European Data Protection Board in their current version for guidelines relating to the processing of personal data

still requires an additional legal basis under [58] Art. 6.
Given that the definition of "terminal equipment" from [34] Art. 1(a) includes any equipment indirectly connected to a public network, the Electric Vehicle Communication Controller (EVCC) may also be interpreted as "terminal equipment" (indirectly connected to a public network via the CS). Hence, the provisions of [56] Art. 5(3) also apply to the EV charging use case. However, as EV charging could be seen as an information society service based on the definition from [55] Art. 1(1)(b) (i.e., the eMSP is not present, charge authorization and billing are provided by electronic means, and the service is provided at the individual request of the EV user), arguably only the GDPR's provisions apply here.

in the context of connected vehicles and mobility related applications [19]. First, they recommend the processing of personal data to be exclusively local whenever possible as this presents less privacy- as well as cybersecurity risks and could even fall outside of the scope of the GDPR based on [58] Art. 2(2)(c). Second, only function-relevant data should be processed by default, no data should be transmitted to third parties, and data should only be stored as long as it is needed for the service. Third, if data needs to be transmitted out of the vehicle, it should whenever possible be anonymized in order to minimize the risk and because anonymous data is not covered by the principles of data protection. If anonymization is not possible, pseudonymization is recommended as alternative in order to reduce the risk of data processing.

2.2.4. Data Protection Impact Assessments

The GDPR stipulates that a data controller must conduct a Data Protection Impact Assessment (DPIA) whenever the processing of personal data involves a high privacy risk (cf. [58] Art. 35). The goal of the assessment is to estimate the impact of the processing operations on the protection of personal data. In particular, it involves at least:

- a description of the processing operations and the purposes of processing,
- an assessment of the appropriateness of the operations in relation to the purposes,
- an assessment of the involved privacy risks, and
- a description of measures to address the privacy risks.

For instance, the European Data Protection Board argues in their current version for guidelines relating to the processing of personal data in the context of connected vehicles and mobility related applications [19] that data processing related to connected vehicles is in most cases resulting in a high risk to the individual's fundamental right of protection of personal data. Hence, a DPIA is most likely required in any data processing application related to EVs based on [58] Art. 35. The Board also recommends a DPIA for connected vehicles use cases without high privacy risk as a best practice.

2.3. Threat Modeling

In system security engineering, the threat modeling process is used in order to identify all possible threats to a system in a systematic way (cf. [135]). Threat modeling starts by building a model of the system under analysis that characterizes its components, interconnections, and usage scenarios and identifies possible dependencies as well as assumptions. Afterwards this system model is further analyzed from an adversary's perspective in order to identify assets, possible attack vectors, and finally specific threats to the system. The resulting comprehensive threat model allows an effective elicitation of requirements, while taking the respective risks of individual threats into consideration.

Several methodologies for security and/or privacy threat modeling and requirement elicitation exist (e.g., [169, 130, 82, 105, 43, 127, 173, 163]). This thesis focuses on two methodologies for threat modeling, namely: STRIDE [82], which targets security threats, and LINDDUN [43], which targets privacy threats. STRIDE is chosen as it is generally regarded as the most mature and widely used threat modeling methodology [172, 90]. Additionally, STRIDE has found broad applicability in industry and science [154, 215] and is argued to be one of the most suitable security threat modeling methodologies in the automotive context [109].

LINDDUN is chosen as it is one of the most well-known DPIA frameworks [18] and its ease of use as well as its reliability have been demonstrated in an empirical evaluation [210]. Furthermore, LINDDUN offers a methodological approach along with an extensive privacy knowledge base for the systematic elicitation of privacy threats, whereby it partially shares the same methodological steps as STRIDE [175, 212].

2.3.1. STRIDE and LINDDUN

STRIDE is a methodology for systematic security threat modeling and was presented by Microsoft as part of their Secure Development Lifecycle (SDL) in [82]. At its core, the methodology is based on a data flow model, i.e., a Data Flow Diagram (DFD), of the system under analysis. Based on this DFD, STRIDE guides the analysis by defining a set of possible threat types and the relation of these threat types to the elements of a DFD. STRIDE is an acronym for the possible security threat types that are covered by the methodology, namely: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege.

LINDDUN on the other hand is a methodology for systematic privacy threat modeling and is based on STRIDE [43]. Analogous to STRIDE, it is centered around a DFD of the system under analysis and guides the analysis with a predefined set of possible threat types as well as the mapping between threat types and DFD elements. LINDDUN is an acronym for the possible privacy threat types of: Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, content Unawareness, and policy/consent Non-compliance.

2.3.2. Threat Modeling Process

The STRIDE methodology defines a threat modeling process that starts by establishing a preliminary overview of the system and its environment, continues with DFD building and security threat mapping, and leads to the extrapolation of specific security risks as well as the planning of risk mitigation strategies. LINDDUN extends the STRIDE process by additionally mapping the DFD elements to privacy threats from which privacy specific risks and mitigation strategies can be derived. The steps of the combined STRIDE and LINDDUN threat modeling process are detailed in the following (cf. [82], Chapter 9 and [43]):[6]

Step 1 *Define use scenarios.* In this step, the scope of analysis is defined based on the intended usage scenarios and considering the target audience as well as possible adversaries.

Step 2 *Gather a list of external dependencies.* All external dependencies of the system (e.g., used operating systems or databases) need to be documented.

Step 3 *Define assumptions.* The security and/or privacy of a system often depends on certain assumptions (e.g., on the secure storage of private keys or the security of cryptography primitives). If these assumptions do not or no longer hold, the entire system's security and/or privacy could be compromised. The goal of this step is to explicitly define these assumptions.

Step 4 *Create external notes.* In this step, security/privacy relevant information about the system and the security/privacy implications of its functions are described in the form of "external security/privacy notes." This helps identify the security/privacy boundaries of the system and helps to secure its operation while preserving privacy.

[6] While the authors of the LINDDUN methodology briefly mention that every step of the STRIDE process should be enhanced with its privacy counterpart, the main contribution of LINDDUN is in the extension to the key steps of STRIDE, i.e., *Step 6*, *Step 7*, and *Step 9* [43]. The description of steps is thus mainly based on STRIDE [82] and extended with the relevant LINDDUN/privacy considerations.

Table 2.1.: Description of DFD Elements (Based on [82])

DFD Element Type	Description
External Entity	An external entity interacts with the system (e.g., by providing input) and cannot be controlled by the developer. For example, a user or an external system could be modeled as an external entity.
Process	A process performs a specific task. For example, user authentication could be modeled as a process.
Complex Process	A complex process is a set of processes, i.e., it performs multiple distinct tasks. For example, a user login could be modeled as a complex process consisting of, inter alia, authentication and authorization.
Data Store	A data store provides persistent storage, e.g., via a database or files.
Data Flow	A data flow represents any exchange of data between entities, processes, and/or data stores. For example, it could be used to model network communication or function calls.
Trust Boundary	A trust boundary indicates the separation between trusted and untrusted elements. This could, for instance, be the boundary between low-privilege and high-privilege process or between physically separate machines. Any data entering through the trust boundary should be analyzed for correctness and it should be assured that any data leaving through it does not leak sensitive information.

Step 5 *Create one or more DFDs of the application being modeled.* The system is modeled via a DFD in which its individual components (entities, (complex) processes, and data stores) as well as the trust boundaries and data flows between these components are represented (cf. Table 2.1 for a description of the different elements of a DFD). The DFD can be iteratively refined by splitting up the higher level complex processes into its lower level sub-processes.

Fig. 2.1 shows an example high-level DFD for the EV charging use case for an analysis of the CS' data flows. The CS' functions are modeled as a complex process and it communicates with the EV (user) and the backend (e.g., for charge authorization). Furthermore, the CS has a local data store (e.g., to cache charge authorization responses). The trust boundaries are between the CS and the EV (user) as well as between the CS and the backend. Hence, any data that the CS receives from or sends to one of these entities should be secured (e.g., to ensure confidentiality, integrity, and authenticity). Since the CS's functions are modeled as a complex process, the DFD should be further refined (i.e., by splitting the CS' functions into its sub-processes of charge authorization, load management, etc.) before continuing the threat analysis.

Step 6 *Determine threat types.* The STRIDE acronym provides the relevant security threat types for this step, namely:

(i) Spoofing – an adversary pretends to be someone else,
(ii) Tampering – an adversary is able to maliciously modify data or code,
(iii) Repudiation – an adversary is able to deny having conducted an action,

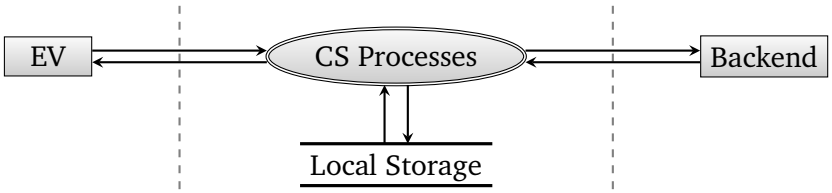


Figure 2.1.: Example of a High-Level EV Charging DFD

(iv) Information disclosure – information is disclosed to someone that should not have access to it,
(v) Denial of service – access to a service by a benign user is not possible or degraded, and
(vi) Elevation of privilege – an adversary is able to increase their capabilities (e.g., gain root rights);

which are the opposites of the common security goals: *(i)* authentication, *(ii)* integrity, *(iii)* non-repudiation, *(iv)* confidentiality, *(v)* availability, and *(vi)* authorization respectively.

Analogously, the LINDDUN acronym provides the relevant privacy threat types for this step, namely:

(i) Linkability – an adversary is able to tell if two Items Of Interest (IOIs)[7] are related,
(ii) Identifiability – an adversary is able to identify the subject associated to an IOI,
(iii) Non-repudiation – a user is not able to deny their past actions or knowledge of something,
(iv) Detectability – an adversary is able to tell if an IOI exists,
(v) Disclosure of information – same as Information disclosure in STRIDE,
(vi) content Unawareness – a user does not know how much data is disclosed to the system, and
(vii) policy/consent Non-compliance – the system is acting outside the given policies or consent;

which are the opposites of the respective privacy goals: *(i)* unlinkability, *(ii)* anonymity/pseudonymity, *(iii)* plausible deniability, *(iv)* undetectability/unobservability, *(v)* confidentiality, *(vi)* content awareness, as well as *(vii)* policy and consent compliance.

While security threats only affect individual DFD elements, some privacy threats affect DFD elements in pairs. For one, linkability refers to a pair of DFD elements of the same type (e.g., linking two data flows). Additionally, identifiability means that a specific entity can be identified based on the respective DFD element, e.g., identifying the sender/receiver of a message (i.e., data flow) or identifying a user within a set of users (i.e., external entities). Lastly, non-repudiation of a DFD element means that an entity cannot deny their involvement in that element, e.g., having sent a message (i.e., non-repudiation of data flow).

Note that a conflict arises when combining STRIDE and LINDDUN: The security goal of non-repudiation is simultaneously a privacy threat. Hence, depending on the specific use case, one has to make a choice between the security goal of non-repudiation and the privacy goal of plausible deniability. For instance, when considering a payment service, most service providers do not want their users to be able to deny having authorized a transaction whereas, when considering a service used by whistleblowers, not being able to deny certain actions might compromise the safety of the user.

Step 7 *Identify the threats to the system.* Using the DFD(s) created in *Step 5* and the STRIDE/LINDDUN threat types from *Step 6*, possible threats to the system are identified based on the mapping in Table 2.2 (all intersections marked with X), i.e., based on the knowledge that per DFD element type only a subset

[7] IOIs can be anything the adversary is interested in, e.g., subjects, messages, actions, attributes, or relations (cf. [161]).

Table 2.2.: Mapping of STRIDE and LINDDUN Threats to DFD Elements (cf. [82] and [43])

DFD Element Types	Security Threat Types						Privacy Threat Types						
	S	**T**	**R**	**I**	**D**	**E**	**L**	**I**	**N**	**D**	**D**	**U**	**N**
External Entity	X		X				X	X				X	
Data Flow		X		X	X		X	X	X	X	X		X
Data Store		X	X	X	X		X	X	X	X	X		X
Process	X	X	X	X	X	X	X	X	X	X	X		X

STRIDE: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege

LINDDUN: Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, content Unawareness, and policy/consent Non-compliance

of STRIDE/LINDDUN threat types apply. For example, regarding external entities, the only relevant STRIDE threat types are spoofing and repudiation and the only relevant LINDDUN threat types are linkability, identifiability, and content unawareness.

Specific threats to a system can be identified using threat tree patterns (cf. [82], Chapter 22 for the STRIDE patterns and [211] for the LINDDUN patterns as an updated version of the old LINDDUN patterns from [43], Section 6). For every valid intersection in Table 2.2, STRIDE and LINDDUN provide a threat tree showing the possible security- and privacy-related preconditions for that specific threat. For instance, the STRIDE threat tree for information disclosure of a process identifies the direct preconditions of this threat as either a corrupt process, a side channel, or the spoofing of an external entity, the last of which is itself the root of another threat tree. As another example, the LINDDUN threat tree for linkability of a data store identifies the direct precondition of this threat as a combination of weak access control to the data and insufficient data minimization.

LINDDUN recommends the documentation of the resulting threats as misuse cases [174] in the form [43]:

Summary: A description of the misuse case.
Assets, Stakeholders, and Threats: A description of the threatened assets, concerned stakeholders, and potential damage.
Primary Misactor: The type of adversary performing the misuse case.
Basic Flow: The flow of actions leading to a successful misuse.
Alternative Flow: Optionally, other flows leading to a successful misuse.
Trigger: How and when the misuse case is triggered.
Preconditions: The preconditions for the misuse case to be feasible.

Note that LINDDUN does not provide its own information disclosure threat trees but simply references the ones from STRIDE (which themselves reference other STRIDE threat trees). This dependency resulted in the notion of LIND(D)UN in [209], i.e., LINDDUN without considering the threat of information disclosure, with the recommendation of conducting a separate full security threat analysis (e.g., using STRIDE) instead of the partial security analysis that results from referencing STRIDE's information disclosure threat trees.

Step 8 *Determine risk.* In this step, the potential risk of each previously identified threat is assessed. In general, risk can be defined as a function over the adverse impact and the likelihood of an event (cf., e.g., the

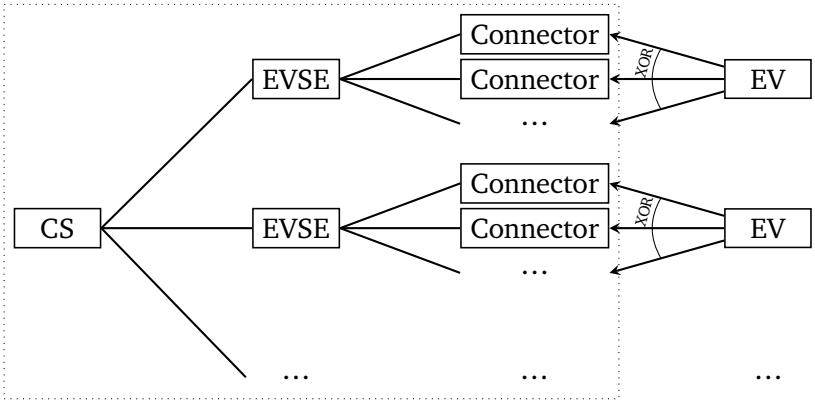


Figure 2.2.: 3-Tier CS Model with at Most One EV Per EVSE

definitions of risk used by NIST [99] (based on [151]) or by BSI [25]). Hence, a risk analysis can be used in order to compare and thus rank the identified threats, which provides the basis for an efficient planning of proportionate mitigation strategies. Neither STRIDE nor LINDDUN prescribe a specific risk assessment method for this step. Hence, it is up to the analyst to choose an appropriate method.

Step 9 *Plan mitigations.* The final step is to plan appropriate countermeasures and defenses addressing the risks of identified threats. In general, possible mitigation strategies are: (i) doing nothing (i.e., accepting the risk), (ii) removing-/turning off the feature (i.e., removing the risk), (iii) warning the user (i.e., transferring the risk), and (iv) countering the threat with technology (i.e., mitigating the risk). Both STRIDE and LINDDUN provide a mapping of possible mitigation technologies that can be used to counter their respective threat types (cf. [82], Table 9-9 and [43], Table 7).

The system model (cf. Section 4) addresses *Step 1* to *Step 4* of the process. The adversary model (cf. Section 5.1) further defines additional assumptions (*Step 3*). The threat analysis (cf. Section 5.2) covers *Step 5* to *Step 7* of the process. The scope of threats that this thesis addresses is limited by the goals (cf. Section 4.3) and assumptions (cf. Section 5.1) and no further risk-based prioritization of threats (*Step 8*) is conducted. That is, all threats that are in scope are considered to be likely enough (as they result from the given adversary model) and are considered to have a high enough impact (as they violate the set goals) in order to be addressed via requirements (cf. Section 6.4). Section 7 covers *Step 9* of the process.

2.4. EV Charging Infrastructure and Protocols

EV charging primarily involves the EV itself and the CS from which it receives electrical energy. CSs can generally be represented in a three-tier model (shown in Fig. 2.2; cf. [147, 53]) starting with the CS as a whole on the highest tier. The CS includes one or more Electric Vehicle Supply Equipments (EVSEs) (i.e., the part of the CS that can transfer energy to one EV at a time) on its second tier. The final tier is the connector (i.e., the physical outlet) of which an EVSE may have one or more (e.g., to support different EV types).

Besides EV and CS, EV charging also involves several secondary actors. For one, a CS is managed by its CSO. The CSO, inter alia, provides its CSs with the information needed to authorize an EV user to charge their EV, sends load balancing signals to its CSs, and receives transaction related meter values from its CSs for billing the EV user. Second, the EV user usually has a contract with an eMSP for all services related to EV operation.

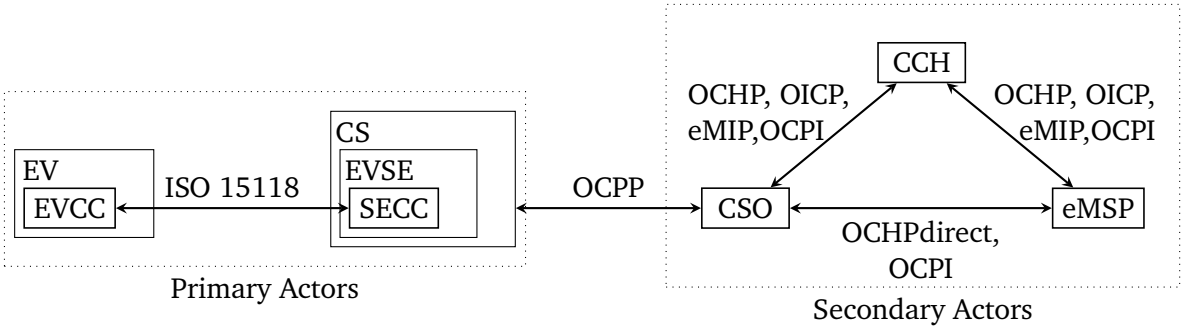


Figure 2.3.: EV Charging Actors and Protocols

For instance, the eMSP handles charge authorization and billing and may offer a service for the navigation to available CSs.

An eMSP usually also acts in other roles, e.g., as electricity provider, as EV Original Equipment Manufacturer (OEM), or as CSO. Hence, if the EV user charges their EV at a CS, which is operated by their eMSP, the charging session can be directly billed based on the information that the eMSP/CSO received from their CS. If, however, this is not the case, charging may still be possible if the CSO has a roaming agreement with the EV user's eMSP. For this, the CSO has to (i) verify that the EV user is authorized to charge by (at least) one eMSP with respective roaming agreement and (ii) provide the EV user's eMSP with all information that is required for billing the charging session.

Communication for the roaming case could be implemented between CSOs and eMSPs directly. Alternatively, to reduce the amount of connections a third backend actor is usually introduced, namely the CCH. A CCH serves as intermediary between multiple CSOs and eMSPs to handle, inter alia, the distribution of authorization data from an eMSP to all respective CSOs and the forwarding of billing information from a CSOs to the right eMSP.

A user-friendly process of charge authorization and billing as well as the introduction of roaming and load balancing functionality, require the provision of high-level communication between all actors involved in the EV charging infrastructure. As this communication takes place between actors from different domains and between systems from different manufacturers, the development of standardized protocols is important to achieve the required level of interoperability.

In the EV charging domain, several protocols have been developed over the past years [50]. Most notably, ISO 15118 [97] is an international standard offering a protocol for high-level communication between EV and CS, OCPP [145, 148] is the de facto standard for communication between a CS and its CSO [50]. Additionally, multiple roaming protocols serve for the communication between CSOs and eMSPs either directly or indirectly over a CCH as intermediary. Based on [106], the roaming protocols Open Clearing House Protocol (OCHP) [179] (with its extension OCHPdirect [178]), Open InterCharge Protocol (OICP) [88, 87], eMobility Interoperation Protocol (eMIP) [75], and Open Charge Point Interface (OCPI) [142] can be considered to be the most widely used, open protocols in Europe and are thus considered in this thesis. The general EV charging architecture is shown in Fig. 2.3 and the relevant protocols are summarized in the following sections.

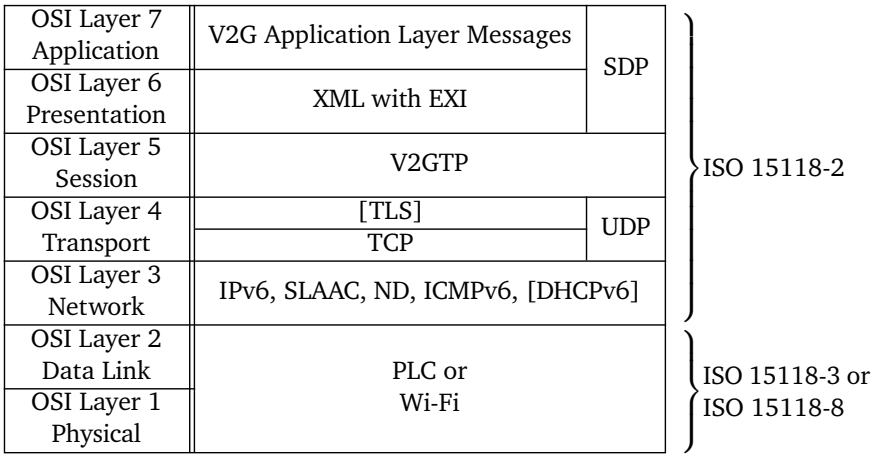


Figure 2.4.: ISO 15118 Network Stack OSI Layers

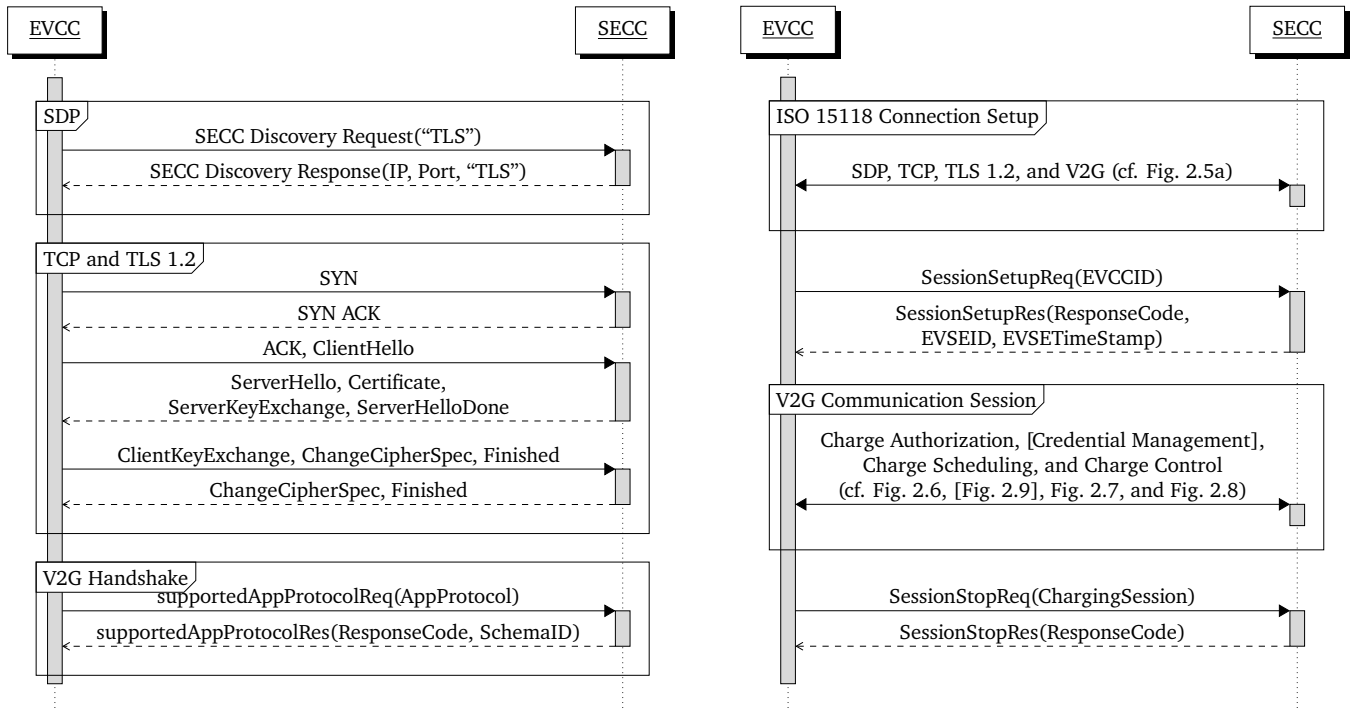
2.4.1. ISO 15118

ISO 15118 [96, 97] is an international standard, describing the communication interface between EV and the EVSE of a CS; more precisely, between the Electric Vehicle Communication Controller (EVCC) and the Supply Equipment Communication Controller (SECC) as the respective ECUs that are responsible for handling communication (cf. Fig. 2.3). ISO 15118 covers several features, including, the identification of the EV user (via an identifier of their contract with the eMSP), the authorization of a charging session based on External Identification Means (EIM) (e.g., using RFID cards or proprietary apps) or PnC (i.e., the EV authenticates itself to the CS with a cryptographic signature using contract credentials, which it received from an eMSP), support for a stable grid operation (via load balancing with the possibility for re-scheduling and V2G power transfer), as well as the provisioning of Value-Added Services (VAS).

The ISO 15118 standard is split over several parts. Part 1, ISO 15118-1, offers general information and high-level use case descriptions. Part 2, ISO 15118-2, defines the communication protocol (message format and sequence definitions, security requirements, timing constraints, etc.) and with that is the most relevant part in the context of this thesis. Depending on the physical connection between EV and CS, OSI Layers 1 and 2 are regulated by ISO 15118-3 (in case of conductive charging using Power Line Communication (PLC)) or ISO 15118-8 (in case of inductive charging using Wi-Fi) while ISO 15118-2 regulates OSI Layers 3-7 in any case (cf. Fig. 2.4).

ISO 15118 Communication Setup

ISO 15118 communication starts with the SECC Discovery Protocol (SDP) over User Datagram Protocol (UDP), which is used by the EVCC to get the IP address and port number of the SECC (cf. [97], Section 7.10). The SDP only consists of two messages, namely the *SECC Discovery Request* sent by the SDP client (usually the EVCC) as IP multicast on the local link, and the *SECC Discovery Response*, sent by the SDP server (usually the SECC) as unicast to the source adders of the requester. The *SECC Discovery Request* indicates if the EVCC wants to use Transport Layer Security (TLS) for the following exchange of V2G application layer messages (ISO 15118 currently relies on TLS 1.2 [44]) and the *SECC Discovery Response* includes the SECC's IP address, port, and indicates if the SECC supports TLS. After the SECC discovery is complete, ISO 15118 communication continues with the exchange of V2G application layer messages (cf. [97], Section 8).



(a) General ISO 15118 Connection Setup

(b) V2G Communication Session Setup

Figure 2.5.: Start of ISO 15118 Communication

V2G messages use XML as representation schema and are transmitted in Efficient XML Interchange (EXI) encoding (for faster processing and smaller message sizes; cf. [97], Section 7.9). The session layer for both SDP and V2G messages uses the V2G Transfer Protocol (V2GTP). A V2GTP data unit consists of an 8 byte header (including protocol version, payload type and payload length) and a variable length payload (i.e., the SDP message or EXI encoded V2G message). Depending on the results of the SDP, V2G messages are either transported over Transmission Control Protocol (TCP) with or without TLS. However, without TLS, only EIM authorization can be used (i.e., PnC is not possible). For SDP messages, UDP serves as transport layer.

ISO 15118 uses IPv6 for its network layer (cf. [97], Section 7.6). The EVCC uses Stateless Auto Address Configuration (SLAAC) to automatically assign an IPv6 address to its interface. In order to ensure that this automatically assigned address is unique, Neighbor Discovery (ND) is used. Additionally, Internet Control Message Protocol version 6 (ICMPv6) is used for network-level error messaging and Dynamic Host Control Protocol version 6 (DHCPv6) may optionally be implemented for address assignment (support for SLAAC is mandatory).

The ISO 15118 V2G message exchange is initiated with a handshake between SECC and EVCC (cf. [97], Section 8.2). This handshake is used to negotiate an application layer protocol (currently only ISO 15118-2) and version, whereby the version consist of a major and minor number. While a connection cannot be established with deviating major numbers, a deviation in minor numbers is acceptable and indicates that additional data elements should be expected and (if they cannot be processed) ignored by the entity with lower minor number (cf. [97], Section 8.2.4.2). After a successful handshake, the ISO 15118 communication setup is complete (cf. Fig. 2.5a).

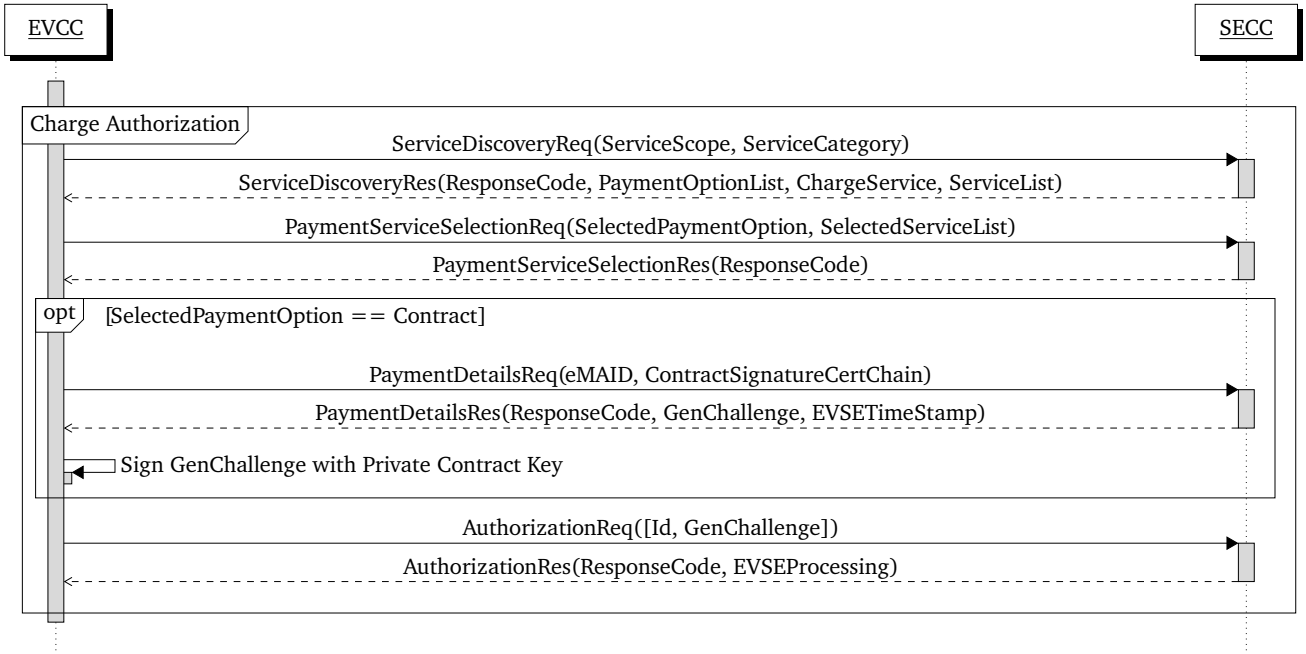


Figure 2.6.: ISO 15118 Communication Session - Charge Authorization

ISO 15118 Communication Session

After the initial communication setup, a V2G communication session can begin (cf. [97], Section 8.4). This session includes all messages that are related to managing the charging process. A session is started or resumed with the *SessionSetupReq/Res* messages and paused (e.g., if the battery overheats) or terminated with the *SessionStopReq/Res* messages (cf. Fig. 2.5b). Afterwards, the general messaging sequence of a communication session – divided into charge authorization, scheduling, and control – is as follows:

Charge Authorization (cf. Fig. 2.6): In order to start charging, the CS first has to verify if the EV is authorized to do so with the respective CSO (either directly or via a roaming agreement). This process is started with *ServiceDiscoveryReq/Res* messages, which are used to indicate the SECC's supported payment options (EIM and/or PnC), charge services (AC/DC), and optionally additional services (anything besides charging, e.g., VAS). The EVCC can request additional information about an SECC's services using the *ServiceDetailReq/Res* messages (not shown in Fig. 2.6).

Afterwards, the EVCC selects a payment option and which of the offered services to use in the *PaymentServiceSelectionReq/Res* messages. If the EVCC chose PnC as payment option, it sends its contract certificate (i.e., an X.509 public key certificate; cf. [35]) and the respective certificate chain to the SECC using the *PaymentDetailsReq* message. A contract certificate is issued by an eMSP to an EV user after the conclusion of their e-Mobility Service Contract and the certificate uniquely identifies this contract (via the inclusion of an e-Mobility Account Identifier (eMAID) in the certificate's subject field). The SECC validates the received certificate chain (using "Certification Path Validation"; cf. [35]) and sends a challenge (i.e., a randomly generated nonce) to the EVCC in the *PaymentDetailsRes* message. The *PaymentDetailsReq/Res* messages are not used for EIM.

Finally, the EVCC's charge authorization is provided using the *AuthorizationReq/Res* messages. In case

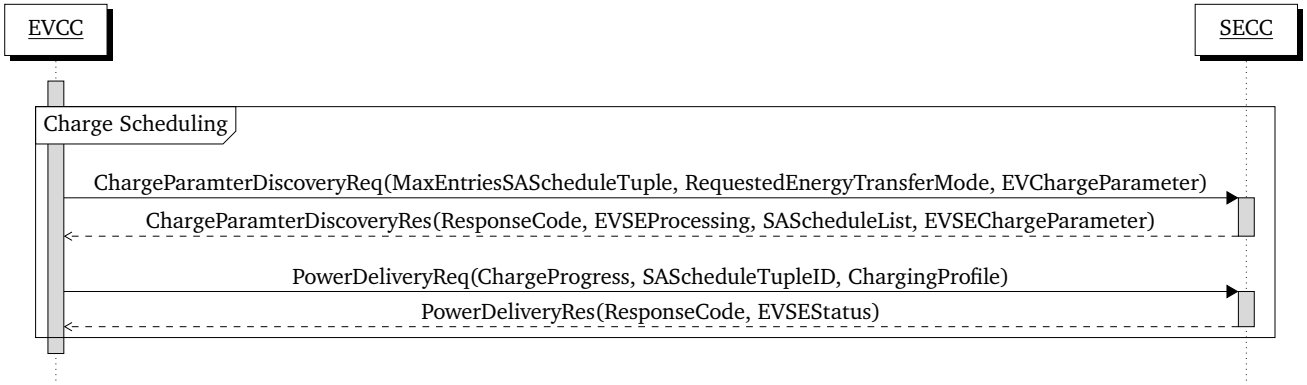


Figure 2.7.: ISO 15118 Communication Session - Charge Scheduling (AC)

of PnC, the *AuthorizationReq* message includes a cryptographic signature[8] by the EVCC with its private contract key over the SECC's challenge. The SECC uses the public contract key from the previously received contract certificate to verify the signature and uses the contract certificate's eMAID for the verification of the EV user's charge authorization status. In case of EIM (e.g., based on the user's RFID card), the EVCC periodically sends empty *AuthorizationReq* messages until the SECC indicates that the authorization verification process is complete in its *AuthorizationRes* message. Note that the actual verification of authorization for both PnC and EIM is out of scope for ISO 15118 and involves OCPP (cf. Section 2.4.2) as well as roaming protocols (cf. Section 2.4.3).

Charge Scheduling (cf. Fig. 2.7): After the EV user was successfully authorized to charge their EV, the EVCC and SECC negotiate charging parameters (e.g., to ensure that energy consumption stays within the capacity of the energy grid or to leverage the volatile production of renewable energy sources). The EVCC first sends a *ChargeParameterDiscoveryReq* message with the requested charging mode (e.g., single or three phase AC), the expected departure time, the estimated amount of energy that is required in order to reach the EV users configured charging goal, and information on the EV charging system's capabilities (e.g., maximum supported voltage and current). Afterwards, the SECC provides its own charging capabilities and a list of charge schedules in the *ChargeParameterDiscoveryRes* message. Each charge schedule includes the maximum amount of power that can be drawn from the EVSE over a defined time interval (supplied to the EVSE by its CSO) and optionally a sales tariff from the eMSP (i.e., the cost of charging over time and/or consumption). Each sales tariff is signed by the eMSP such that the EVCC can verify its authenticity. The SECC's information can be used by the EVCC to determine an optimized charging profile (e.g., only charge during low cost periods if this does not violate the user's defined charging goal).

For DC charging, *CableCheckReq/Res* messages (to guarantee charging safety, e.g., verifying that the connectors are locked) and *PreChargeReq/Res* (to adjust the EVSE's voltage based on the EV's battery) messages are exchanged (not shown in Fig. 2.7).

The EVCC requests to start charging with the *PowerDeliveryReq* message. This message identifies the selected charge schedule and optionally the maximum amount of power that the EV intends to draw over time (must be less than or equal to the maximum from the charge schedule). If the SECC accepts the EVCC's charge profile in its *PowerDeliveryRes* message, the EVSE provides voltage to its connector and the EV can start charging.

[8] ISO 15118-2 uses the XML signature mechanism (cf. [13]) with Secure Hash Algorithm (SHA)-256 and Elliptic Curve Digital Signature Algorithm (ECDSA) based on the *secp256r1*, aka *NIST_P256*, curve.

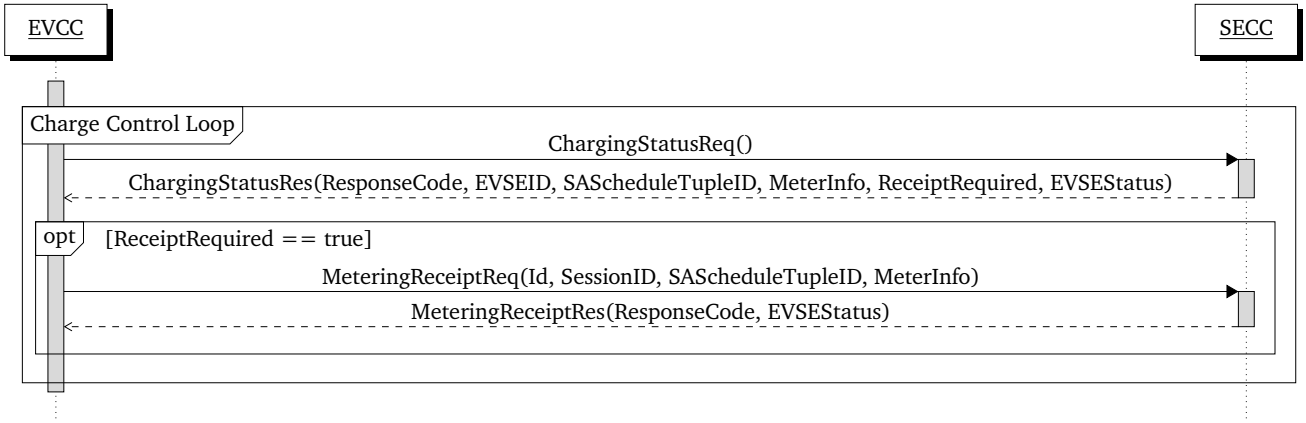


Figure 2.8.: ISO 15118 Communication Session - Charge Loop (AC)

Charge Control Loop (cf. Fig. 2.8): Which messages are exchanged while the EV is charging depends on the type of charge (AC or DC). For AC charging, the EVCC periodically sends empty *ChargingStatusReq* messages. The corresponding *ChargingStatusRes* messages from the SECC contain the EVSE's current electricity meter readings (allows the EV to perform sanity checks) and the currently valid charge schedule. *ChargingStatusRes* messages also allow the SECC to request meter receipts from the EVCC (i.e., a signature over the current meter values with the EVCC's contract key), and enables the SECC to request a renegotiation of the charging profile as well as stop the charging session. Note that the EVCC can always initiate a renegotiation or stop a charging session by sending a *PowerDeliveryReq* instead of *ChargingStatusReq* message.

If a renegotiation was requested, the Charge Scheduling phase is entered (i.e., EVCC sends *ChargeParameterDiscoveryReq* message, etc.). If a metering receipt was requested, the EVCC creates a signature over the current session ID, charge schedule, and meter values (as received from the SECC) with its private contract key and sends the result to the SECC via the *MeteringReceiptReq/Res* messages. A charge session is ended with a *PowerDeliveryReq/Res* messages (not shown in Fig. 2.8).

For DC charging, the EVCC periodically sends *CurrentDemandReq* messages (instead of the *ChargingStatusReq* messages used for AC charging), which allow the EV to inform the EVSE about the requested amount of current and specify the target voltage (not shown in Fig. 2.8). After stopping a DC charging session (with *PowerDeliveryReq/Res* messages), the EVCC may additionally request a welding detection with the *WeldingDetectionReq* message (not shown in Fig. 2.8).

EVCC Credentials

As mentioned in the description of an ISO 15118 communication session, the PnC identification mode requires the EVCC to possess valid contract credentials (i.e., a valid contract certificate and the corresponding private key), issued by an eMSP. In order to supply contract credentials to the EVCC with minimal user interaction, ISO 15118 includes a credential management functionality as follows:

Credential Management (cf. Fig. 2.9): ISO 15118 provides two message pairs that allow an EVCC to request new contract credentials during a communication session, namely *CertificateInstallationReq/Res* (for the initial provisioning of contract credentials) and *CertificateUpdateReq/Res* (for an update contract credentials). One of these message pairs may be exchanged between the *PaymentServiceSelectionReq/Res*

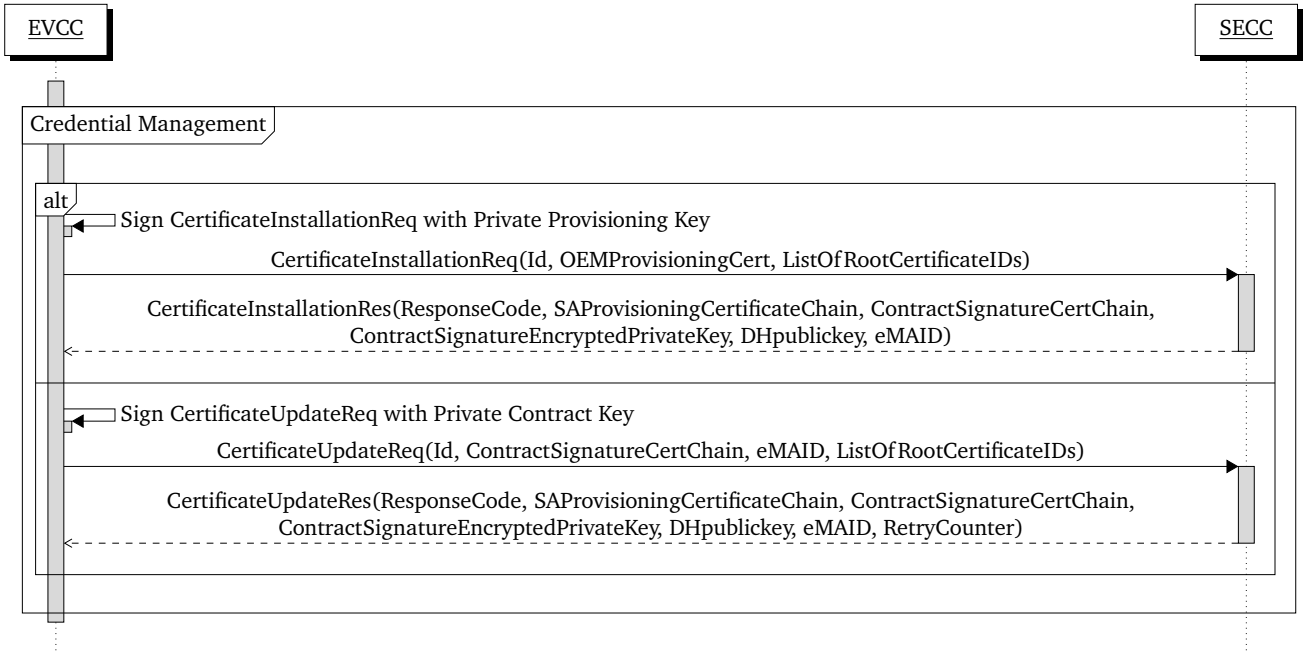


Figure 2.9.: ISO 15118 Communication Session - Credential Management

and *PaymentDetailsReq/Res* messages (cf. Fig. 2.6), i.e., after the SECC has confirmed the EVCC's selection of the contract payment option but before the EVCC sends its contract certificate to the SECC.

A prerequisite for the installation of contact credential into an EVCC over ISO 15118 is that the EVCC possesses valid provisioning credentials. These credentials are issued by the EVCC's OEM and installed into it during its manufacturing process. The provisioning credentials contain a Provisioning Certificate Identifier (PCID) (uniquely identifies them), serve as the EVCC's long term identity and are intended to be valid throughout the entire lifetime of the EV (30 years; cf. [97], Annex E). The second prerequisite of ISO 15118's credential installation process is that the EV user registers their EV's PCID (which the user received, for example, in an information sheet after purchasing the vehicle) with their eMSP after the conclusion of their e-mobility service contract. The third prerequisite is that the SECC supports the use of the credential management messages, which is indicated by the service list in the SECC's *ServiceDiscoveryRes* message.

If these prerequisites are met, the EVCC can send one of the credential request messages. The *CertificateInstallationReq* is used if the EVCC does not possess any valid contract credential, e.g., during its first charging session or after its previous contract credentials have expired. This message includes the EVCC's provisioning certificate and is signed with its private provisioning key. The *CertificateUpdateReq* is used if the EVCC's current contract credentials are about to expire but currently still valid, e.g., if the contract certificates validity ends in less than 20 days. This message includes the EVCC's current contract certificate and is signed with its private contract key.

The credential request message is forwarded from the SECC to the Certificate Provisioning Service (CPS).[9] The CPS serves as trustworthy intermediary in the Credential Management process (cf. [97],

[9] An SECC/CS usually only directly communicates with its CSO, i.e., the message is sent to the CSO from where it is forwarded to the CPS. Since ISO 15118 does not specify requirements for communication with the backend, details on the backend communication are not included in this section.

Section 7.9.2.5) and verifies the signature over the EVCC's request before forwarding the request to the eMSP (cf. [97], Requirement V2G2-894).[10] The eMSP identifies the EV user's contract based on the PCID (or based on the eMAID in case of a *CertificateUpdateReq*). If the signature is valid and a contract exists, the eMSP generates new contract credentials and encrypts the new private contract key with the received public provisioning key (or with the old public contract key in case of a *CertificateUpdateReq*). Encryption of the private contract key uses a hybrid encryption scheme: The eMSP generates a new ephemeral Elliptic Curve Diffie Hellman (ECDH) key pair, uses the private ephemeral ECDH key together with the received public provisioning/contract key in the ECDH algorithm to generate a shared secret, generates a symmetric key based on this shared secret, and encrypts the new private contract key with this symmetric key using the AES-CBC-128 algorithm (cf. [97], Section 7.9.2.4.3).

The eMSP then generates a *CertificateInstallationRes/CertificateUpdateRes* message, including the contract credential's certificate chain (with the Sub-Certificate Authority (CA) certificates of the eMSP and the actual contract certificate), the encrypted new private contract key, and the public ephemeral ECDH key. This response message is then sent to the CPS where the CPS's certificate chain is added and the message is signed in order to protect its authenticity.[11] Afterwards, the repose message is sent to the EVCC over the SECC. The EVCC can then, after verifying authenticity, encrypt its new private contract key (by generating the same symmetric key as the eMSP based on ECDH with the received public ephemeral ECDH key and its private provisioning or old contract key) and store the contract credentials for later use.

ISO 15118 Public Key Infrastructure

The security concepts of ISO 15118 rely on the Public Key Infrastructure (PKI) as shown in Fig. 2.10 (cf. [97], Annex E). All certificate chains at most four certificates long, starting with the respective root CA certificate under which at most two Sub-CAs are included before the leaf certificate. The ISO 15118 PKI specifies three different kinds of root CA that are used as trust anchors for different Sub-CA and leaf certificates as follows:

- An **EV OEM root CA** is used to certify automotive OEM Sub-CAs, which in turn certify the EVCC's OEM provisioning credentials. These provisioning certificates are used in the process of requesting contract credentials from an eMSP in order to protect the authenticity of *CertificateInstallationReq* messages and to protect the confidentiality of private contract keys in *CertificateInstallationRes* messages. Hence, the validity of provisioning certificates only needs to be verified by eMSPs.
- An **eMSP root CA** is used to certify eMSP Sub-CAs, which in turn certify the EVCC's contract credentials. An eMSP Sub-CAs may additionally sign sales tariffs for inclusion in an SECC's *ChargeParameterDiscoveryRes* messages during ISO 15118's Charge Scheduling phase. Hence, the validity of eMSP Sub-CA certificates needs to be verified by EVCCs. Contract credentials are used to sign an SECC's challenge for PnC authorization, to protect the authenticity of *CertificateUpdateReq* messages and the confidentiality of private contract keys in *CertificateUpdateRes* messages, and to sign metering receipts during ISO 15118's Charge Control Loop phase. Hence, the validity of contract certificates needs to be verified by SECCs (or by CSOs if they handle certificate validation for their CSs) and by eMSPs.

[10] The CPS may be operated by the eMSP, by the CSO, or by a third party (cf. [97], Section 7.9.2.5).

[11] The message is signed by the CPS since the EVCC does not necessarily posses the required root CA certificate to verify a signature from the eMSP.

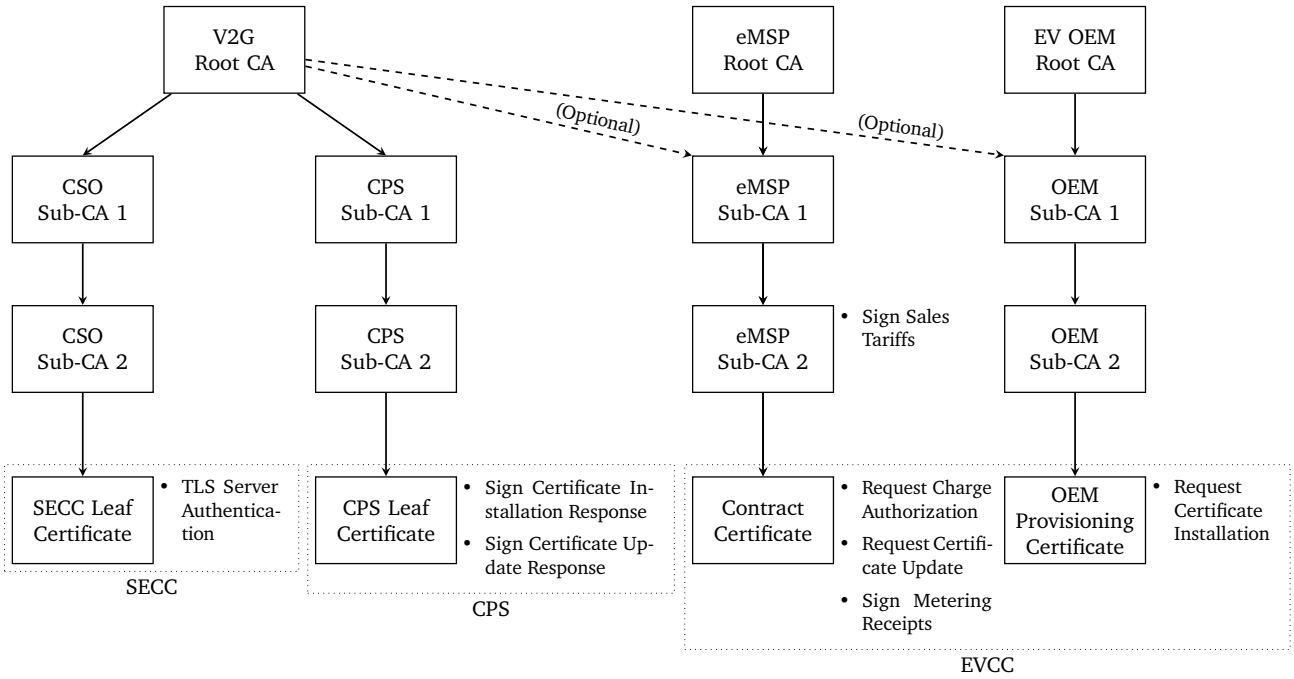


Figure 2.10.: ISO 15118 PKI

- A **V2G root CA** serves as trust anchor for any leaf certificates that need to be validated by the EVCC. Hence, they are the only roots that need to be installed in EVCCs. The V2G root CA is the basis of two distinct certificate chain, namely:

  1. The CSO chain down to the individual SECC leaf certificates. These leaf certificates (and their respective private keys) are used by SECCs to authenticate themselves against an EVCC using unilateral TLS server authentication, i.e., during the TLS session establishment before ISO 15118 V2G messages exchange.

  2. The CPS chain down to the individual CPS leaf certificates. These leaf certificates (and their respective private keys) are used to ensure the authenticity of an eMSP's *CertificateInstallationRes/CertificateUpdateRes* message in ISO 15118's Credential Management process via a cryptographic signature. Note that this is a signature in ISO 15118 messages and the contract certificate itself is still signed by the eMSP. The additional signature by the CPS is only to reduce storage requirements on the EVCC (i.e., the EVCC only has to store relevant V2G root certificates instead of all possible eMSP root certificates to verify the authenticity of a received certificate installation response).

2.4.2. OCPP

OCPP [145, 148] is the de facto standard for communication between a CSO and their CSs [50]. OCPP is developed by the Open Charge Alliance (OCA) and exists in multiple versions, most importantly OCPP 1.6 [145], which was one of the main drivers in OCPP's wide adoption [2], and OCPP 2.0.1 [146, 147, 148], which at the time of writing is the newest version of the protocol and contains significant advances in terms of functionality and security [155].

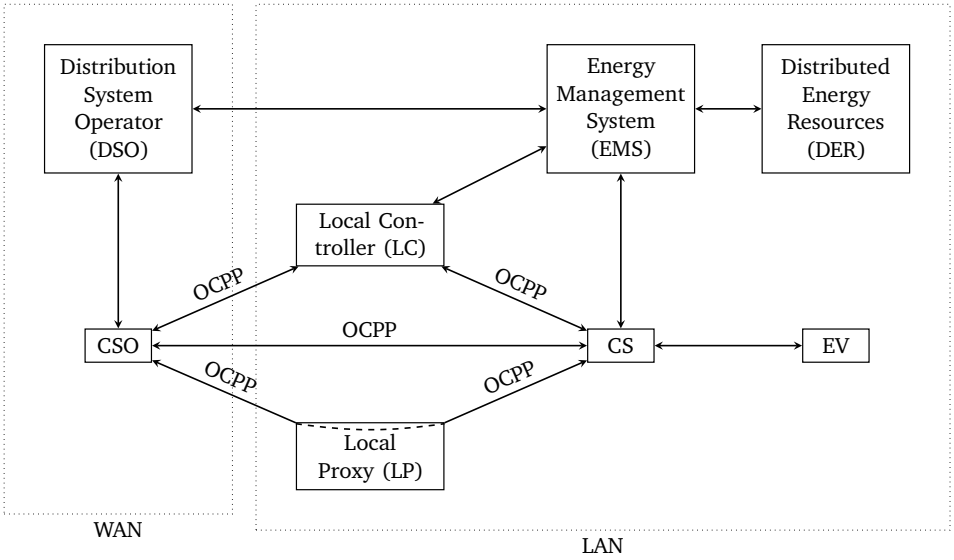


Figure 2.11.: OCPP Topologies

OCPP 1.6 [145] offers support for the authorization of charging sessions, the exchange of (transaction-related) meter values (for billing purposes), smart charging (to stabilize grid operation via load balancing), CS reservations, and general CS management (configuration, diagnostics, firmware updates, etc.). Furthermore, OCPP 1.6 is extendable via its *DataTransfer* messages, allowing the transfer of arbitrary data. OCPP 1.6 supports two kinds of communications stacks: (i) OCPP 1.6-S [150] using Simple Object Access Protocol (SOAP) over Hypertext Transfer Protocol (HTTP) and (ii) OCPP 1.6-J [149] using JavaScript Object Notation (JSON) over WebSockets. Both are based on TCP/IP with optional TLS (cf. [150], Section 6.2 and [149], Section 6.2).

OCPP 2.0.1 [146, 147, 148] further extends on the functionalities of OCPP 1.6. The main additions of OCPP 2.0.1 are (cf. [146]): Unified device management,[12] more detailed security concepts,[13] extended smart charging capabilities (including support for a local Energy Management System (EMS)), and ISO 15118 support (including PnC authentication, smart charging with EV input from the Charge Scheduling phase, and credential management). In addition to *DataTransfer* messages, OCPP 2.0.1 can be extended via an optional *CustomData* element, allowing the transfer of arbitrary data in all existing messages OCPP 2.0.1 only supports communication using JSON over WebSockets as this method incurs a lower overhead than the SOAP variant. A minimal overhead is especially relevant for OCPP as CSs may operate over a cellular network connection, which is billed based on traffic volume. We focus on OCPP 2.0.1 in the remainder of this thesis due to its native support for ISO 15118. Any following references to OCPP refer to OCPP 2.0.1 unless specified otherwise.

OCPP Topologies

OCPP can be used with different topologies (cf. [147], Section 8 and Fig. 2.11). The simplest approach is the direct control of CSs via the CSO without the involvement of other actors. Additionally, the CSO may receive control signals from a Distribution System Operator (DSO) in order to adjust the consumption of its CSs to the constraints of the local grid. Moreover, a Local Proxy (LP) may be used to route messages between a CSO and

[12] The device management feature of OCPP 2.0.1 allows the representation of CSs via a generalized device model (based on the 3-tier CS model, cf. Fig 2.2) in order to simplify a CSO's tasks of CS configuration, monitoring and inventory reporting.
[13] Note that the OCA has released a white paper on the use of the security concepts from OCPP 2.0.1 in OCPP 1.6-J (cf. [144]).

the CSs at one site. For instance, if the CSs have little to no access to the cellular network (e.g., in a parking garage), an LP (with cellular uplink and wired connection to the CSs) could be used to increase connectivity. Furthermore, a Local Controller (LC) could also be used as intermediary between CSO and CSs. Opposed to the LP, however, an LC does not only forward messages but can also initiate messages to the CSs. An LC could, for instance, be used for local smart charging. For this, the LC receives information on the maximum allowed total power usage of the local site from the CSO and can than independently distribute this power to its CSs. Additionally, an EMS could be integrated into the topology in order to incorporate local Distributed Energy Resources (DER) (e.g., battery storage or solar panels) and/or control signals from a DSO into the smart charging decisions. For this, the EMS could be connected to the CSs or if available to the LC.

OCPP Communication Setup

The OCPP communication setup starts with the establishment of a WebSocket connection, i.e., a TCP connection over which an HTTP Upgrade is sent, indicating the switch to WebSockets. However, depending on the configured security profile, this connection establishment slightly varies as follows (cf. [148], Section A.1.3):

SP_1 **Unsecured Transport with Basic Authentication:** For this security profile, no TLS connection is established. The CS is authenticated via the HTTP Basic method from [62] (i.e., the CS' ID and password are sent in plaintext in the HTTP Upgrade message). Additionally, the CSO is not authenticated. The OCA recommends to only use this profile if the network is trusted, e.g., if CS and CSO are connected over a Virtual Private Network (VPN).

SP_2 **TLS with Basic Authentication (cf. Fig. 2.12):** For this security profile, a TLS connection is established with unilateral server-side authentication. Hence, only the CSO is authenticated with TLS using its certificate. The CS is again authenticated via the HTTP Basic method.

SP_3 **TLS with Client Side Certificates:** For this security profile, a TLS connection with bilateral authentication is established. Hence, the CSO and CS both are authenticated with TLS using their respective certificate.

For the management of security profiles and their required data, OCPP includes messages for changing a CS' HTTP Basic password, updating a CS' certificate (the CS can generate a new key pair locally and send a Certificate Signing Request (CSR) to the CSO), and upgrading a CS' security profile (note that downgrades are not allowed over OCPP).

OCPP Communication

Following the CS' initial connection setup after a boot, the CS first needs to be registered with its CSO (cf. [148], Section B and Fig. 2.13). For this, the CS sends a *BootNotificationRequest* message (including information about itself, e.g., its serial number and firmware version) and receives a *BootNotificationResponse* message, indicating if it was accepted at the CSO, the current time (for clock synchronization), and an interval for sending *Heartbeat* messages. *Heartbeat* messages are used to indicate that the CS is still responsive and for a periodic clock synchronization[14] (based on the CSO's current time included in the response). A

[14] In earlier versions of OCPP (using SOAP over HTTP), the main purpose of *Heartbeat* messages was to ensure that the connection between CS and CSO is available (cf. [145], Section 4.6). However, since WebSockets include their own heartbeat mechanism to periodically validate connectivity (cf. Ping/Pong in [60]), the OCPP *Heartbeat* is mainly used for time synchronization since the standardization of OCPP 1.6-J.

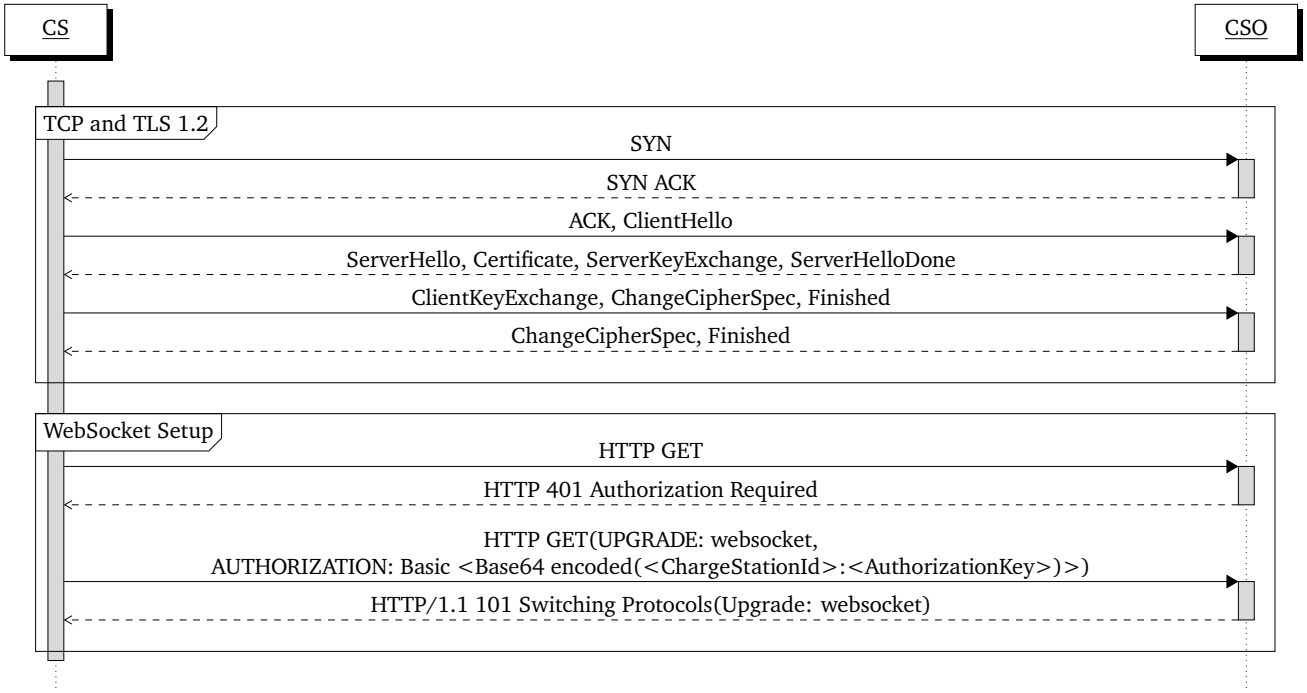


Figure 2.12.: OCPP Connection Setup with Security Profile SP_2

Heartbeat is sent if no other message was sent for the defined interval or at least every 24 hours for time synchronization. After the *BootNotificationRequest/-Response* messages, the CS informs the CSO of the status of all of its connectors using the *StatusNotificationRequest* message (cf. Fig. 2.13). This status information is based on the CS' self-checks in order to detect, e.g., possible hardware faults.

After the CS was accepted by its CSO and assuming no hardware faults were detected, the CS can start its normal operation, i.e., waiting to start a charging session. A charging session is typically started after the authorization of the EV user and is encapsulated in an OCPP transaction (for the reporting of billing-relevant data) as follows:

EV User Authorization: Before the CS allows an EV to charge, the EV user first has to be authorized by the CSO. In OCPP, charge authorization is based on ID Tokens (cf. [148], Section C). For this, the EV user presents their ID Token to the CS (e.g., using an RFID card). Afterwards, the CS sends an *AuthorizeRequest* message (including the ID Token) to the CSO and receives an *AuthorizeResponse* message, indicating the status of the ID Token (accepted, blocked, not allowed at this location, etc.) and further information (e.g., the expiration date, a charge priority, and a group to which the ID Token belongs). The EV is cleared to charge if the ID Token was accepted by the CSO.

For PnC authentication based on ISO 15118, the same messages are used with the eMAID from the EVCC's contract certificate as ID Token. In the PnC case, the *AuthorizeRequest* additionally includes the data that is required to check the revocation status of the certificates in the EVCC's contract certificate chain using Online Certificate Status Protocol (OCSP) (cf. [168]) requests (i.e., for each certificate: A hash over the issuer's distinguished name, a hash over the issuer's public key, the certificate's serial number, and the URL of the OCSP responder). After receiving this message, the CSO sends the corresponding OCSP requests or (if available) checks the certificates' revocation status via cached responses and additionally includes the results (i.e., whether the certificates are revoked or not) in the *AuthorizeResponse* message.

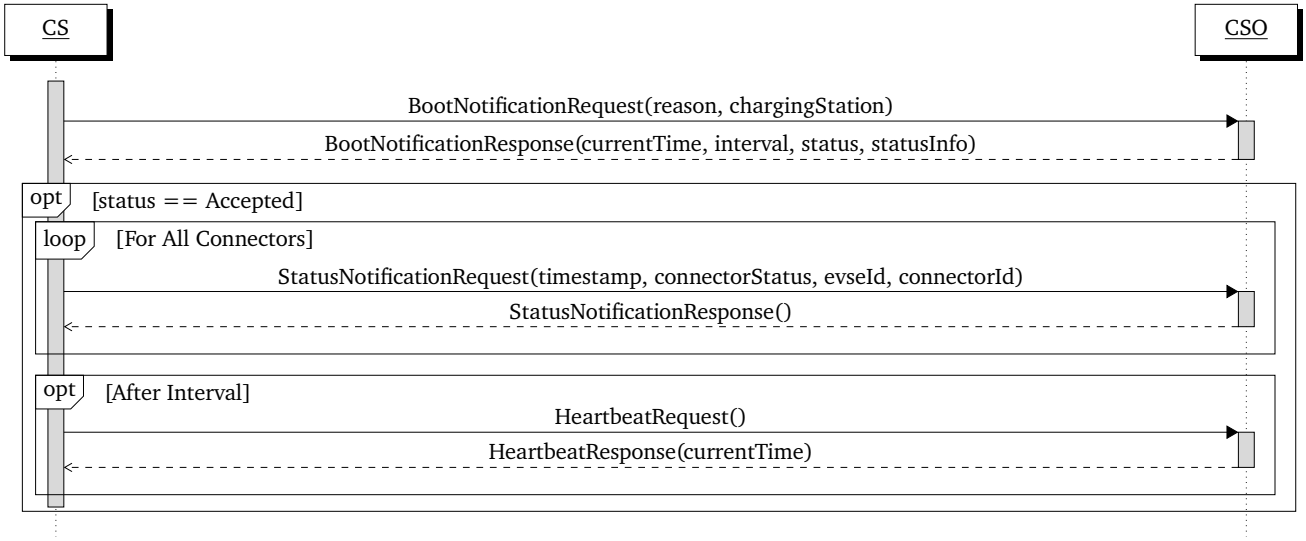


Figure 2.13.: OCPP Communication Start

Optionally, if the CS is unable to validate the EVCC's contract certificate chain (e.g., because it does not have the necessary root certificate), it can include the entire chain in the *AuthorizeRequest* for the CSO to validate.

In addition to starting a charging session, the EV user also needs to be authorized to stop the session. For this, the same *AuthorizeRequest/-Response* messages are used. However, a session may only be stopped by the same ID Token that started the session or by another ID Token in the same group (for cases where multiple persons share the same EV, e.g., in families or businesses).

Requiring the *AuthorizeRequest/-Response* messages before (and at the end of) each charging session, however, has the drawbacks that it requires the CSs to be online, incurs additional overhead, and increases delays. For this reason, OCPP includes two methods for offline authorization as follows:

Authorization Cache: The authorization cache is used by the CS to automatically store the status of previously seen ID Tokens (cf. [148], Section C.1.3). Anytime the CS receives information on an ID Token from its CSO (e.g., in *AuthorizeResponse* messages), the CS updates its cache with the new information (the status of the ID Token, its expiration time, its group, etc.). When an EV user presents their ID Token to the CS (i.e., to start or stop a charging session), the CS first checks if the presented ID Token is stored in its local cache. If this is the case and the ID Token's cached status is set to accepted, the user is authorized to charge. If the ID Token is not stored in the cache or if its cached status is not accepted, a new *AuthorizeRequest* is sent to the CSO.

Local Authorization List: The local authorization list allows the CSO to explicitly provide a list with ID Token information to its CSs (using the *SendLocalListRequest* message; cf. [148], Section C.1.4). OCPP allows a CSO to send a full list as well as differential updates (i.e., adding, changing, or deleting specific entries). When an EV user presents their ID Token to the CS, the CS first checks its local authorization list after which it may fall back to online authorization. If both methods for offline authorization are present, local authorization list entries have priority over the authorization cache.

Transactions: In OCPP, a transaction encapsulates all relevant events that relate to a charging session (cable plug in, user authorization, start of power transfer, etc.; cf. [148], Section E). The CS informs the CSO

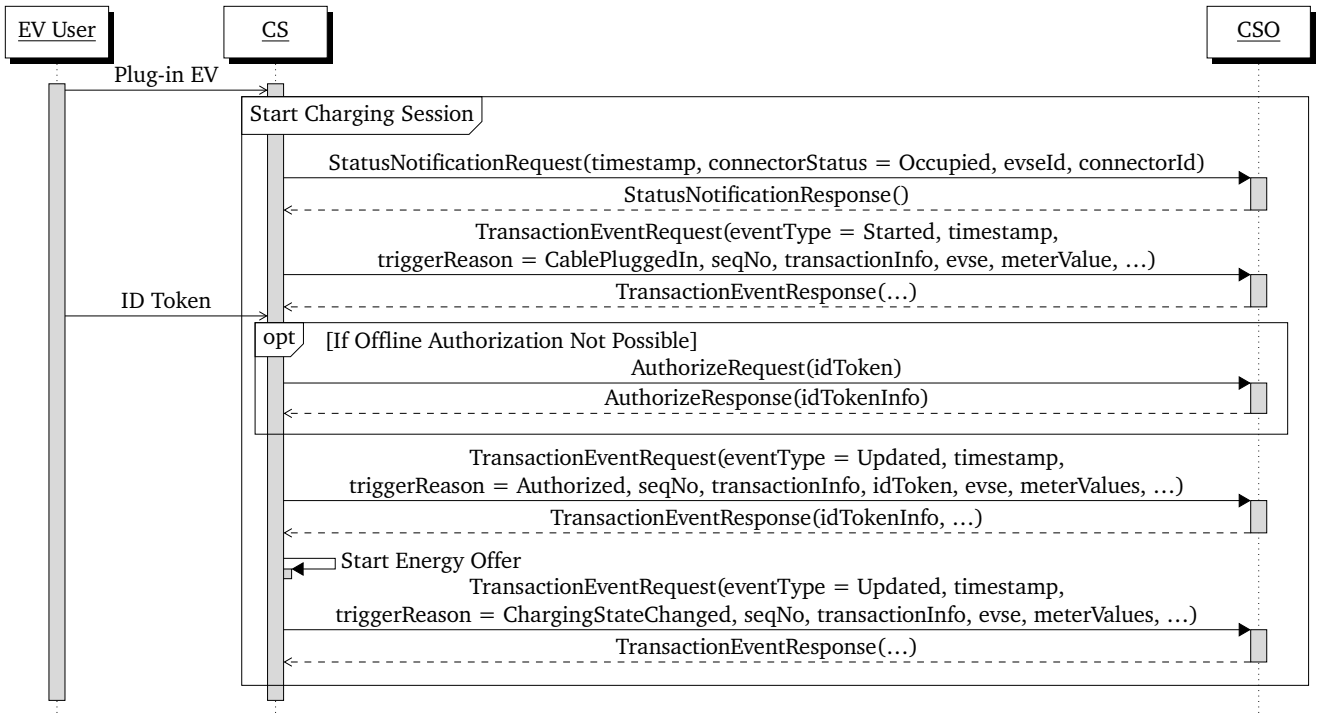


Figure 2.14.: Start of an OCPP Charging Session

about these transaction-related events with its *TransactionEventRequest* messages, including, inter alia, the event trigger, a timestamp, meter values (optionally signed by the CS' energy meter or by the EVCC via an ISO 15118 metering receipt), the EVSE's ID, the user's ID Token, a transaction ID, and a sequence number.

The start of a transaction is indicated by the *TransactionEventRequest(eventType = Started)* message (cf. Fig. 2.14). The respective start trigger can be configured by the CSO (e.g., start a transaction after the EV was detected, after the charging cable was plugged-in, or after the EV user was authorized). The start message may contain the initial meter values if they are known at that point in time (which might not be the case, e.g., if the transaction is started by user authorization before the EV is connected to a specific EVSE).

Fig. 2.14 shows an example for the start of an OCPP charging session. After the EV is plugged-in, the CS first informs the CSO that the respective EVSE is now occupied (*StatusNotificationRequest*) and starts the transaction (i.e., cable plug-in was defined as start trigger). Afterwards, the EV user presents their ID Token to the CS and (if offline authorization is not possible) the CS sends a request for the authorization status. After the user is authorized to charge, the CS informs the CSO of this with a *TransactionEventRequest(eventType = Updated)* message. Note that a *TransactionEventResponse* message includes the current status of the ID Token from the respective request message. Hence, the CS may use this information to verify that a locally authorized ID Token is still valid and to update its authorization cache. Finally, the CS starts the energy offer of which the CSO may again be informed via a *TransactionEventRequest(eventType = Updated)* message.

Additionally, *TransactionEventRequest(eventType = Updated)* messages can be sent during a transaction

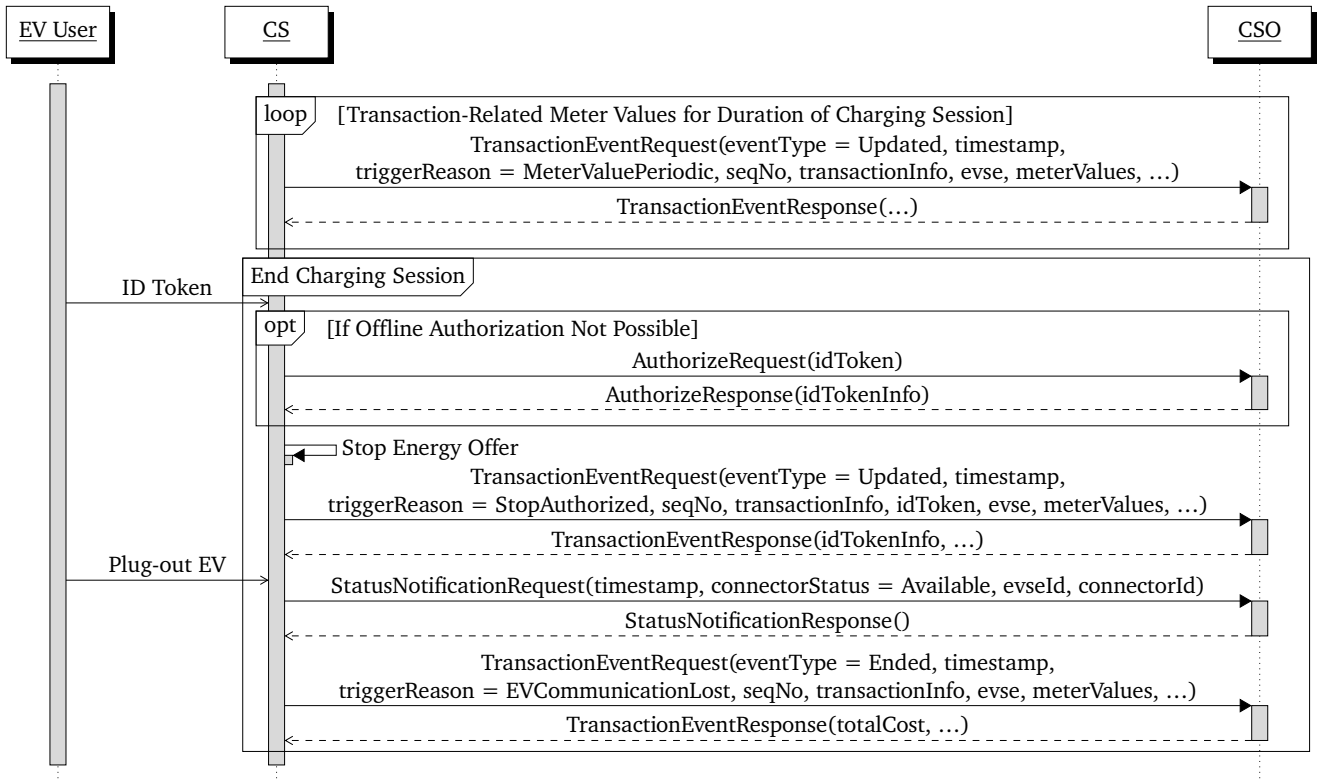


Figure 2.15.: OCPP Charging Session Loop and End

in order to periodically report transaction-related meter values to the CSO (cf. Fig. 2.15).[15] Periodic meter values can be collected relative to the start of the transaction as well as clock-aligned. The end of a transaction is indicated by the *TransactionEventRequest(eventType = Ended)* message. The stop message may contain the initial, final, as well as periodically measured meter values. Fig. 2.15 shows an example of the end of a transaction, after an OCPP charging loop. First, the EV user presents their ID Token (equal to or in the same group as the one that started the session), the EV checks its authorization status and may inform the CSO of this event. After the EV is unplugged, the CS informs the CSO that the respective EVSE is now available and ends the transaction.

Because the transaction-related meter values are used for billing a charging session, OCPP includes a retrying mechanism to make the transmission of these messages more reliable: If the CS is unable to deliver a *TransactionEventRequest* message to the CSO (e.g., due to an error during transmission or if the CS was offline during the transaction), it queues this message and re-sends it after a delay (in case of a previous transmission error) or once the connection to the CSO is restored (in case of an offline transaction). The CSO can later reorder the messages based on the included transaction IDs and sequence numbers.

Besides the functionalities that are required for the charging and billing of EVs (i.e., authorization and reporting meter values), OCPP also supports several auxiliary use cases. For instance, OCPP supports the ISO 15118 Credential Management functionality whereby the CS simply forwards the respective ISO 15118 messages as Base64 strings to the CSO (cf. [148], Section M). In the context of this thesis, the OCPP functionalities of smart charging and remote control are of relevance, as detailed in the following:

[15] Meter values that are not related to a transaction can be sent using the *MeterValuesRequest* message.

Smart Charging: Smart charging of EVs (i.e., scheduling their charging sessions to be more favorable under the given conditions, e.g., by shifting them to off-peak hours) is seen as an important factor in enabling an efficient and reliable operation of the energy grid while supporting the desired growth of the e-mobility market [71]. OCPP enables smart charging via charging profiles, sent to the CS in *SetChargingProfileRequest* messages. OCPP distinguishes between three smart charging use cases (cf. [148], Section K), namely:

1. **Internal Load Balancing** is used within a CS. The CS is assigned a maximum power limit (e.g., based on the physical limit of the respective grid connection), which it can distribute among its EVSEs. The limit for internal load balancing can be set by the CSO by sending a charging profile of type *ChargingStationMaxProfile*.
2. **Central Smart Charging** allows the CSO to influence the power consumption of specific charging sessions, of a single CS, or of a group of CSs. This way, the CSO can adjust a sites consumption based on external signal, e.g., from a DSO (cf. Fig. 2.11). A central smart charging schedule can be configured via the *TxDefaultProfile* (default for all transactions) and *TxProfile* (for a single transaction) charging profiles.
3. **Local Smart Charging** is possible via the inclusion of an LC (cf. Fig. 2.11). The LC can control specific charging sessions, single CSs, or a group of CSs. This use case also enables the inclusion of a local EMS's control signals (cf. Fig. 2.11). Local smart charging uses the same charging profiles as central smart charging.

If multiple charging profiles are active at the same time, the CS chooses per charging profile type the currently active profile with the highest priority and from the chosen profiles uses the one with the lowest limit. The only exception is that an active *TxProfile* always overrules a *TxDefaultProfile*.

In cases where a CS' charging schedule is influenced by an external party (e.g., signals from an EMS), the CS' behavior would diverge from the CSO's expectations. To prevent this divergence, OCPP enables a CS to inform the CSO about schedule changes via *NotifyChargingLimitRequest* messages, including a charging profile of type *ChargingStationExternalConstraints*. Additionally, a CS can request charging schedule for an ISO 15118 *ChargeParameterDiscoveryRes* message via the *NotifyEVChargingNeedsRequest* message and inform the CSO about an EV's charging schedule (received in an ISO 15118 *PowerDeliveryReq*) with the *NotifyEVChargingScheduleRequest* message.

Remote Control: OCPP enables the CSO to remotely start/stop a transaction on a CS (e.g., for charge authorization via a smartphone app; cf. [148], Section F). To start a transaction, the CSO sends a *RequestStartTransactionRequest* message to the CS, which includes the EV user's ID Token. After receiving this message, the CS may check the authorization status of the given ID Token (i.e., using an *AuthorizeRequest* or one of the offline methods) before it finally starts offering energy to the EV. Afterwards, the normal charging loop is executed. A remotely started transaction may be stopped locally (using the process from Fig. 2.15) or remotely by the CSO (e.g., requested by the EV user through their smartphone app). For a remote stop, the CSO sends a *RequestStopTransactionRequest* message to the CS, only containing the respective transaction ID. Remote stops are also possible for locally started transactions. Fig. 2.16 shows the remote start/stop of a transaction, which would replace the EV user presenting their ID Token to the CS in Fig. 2.14 and Fig. 2.15 respectively.

In addition to transaction control, the CSO can remotely trigger the CS to send any message using the *TriggerMessageRequest* message, i.e., it allows the CSO to request messages that are initiated by the CS like a *MeterValuesRequest*. Furthermore, the CSO can reserve a CS/EVSE/connector for a specific EV user (e.g., after a reservation request from the user via an online platform), in order to guarantee

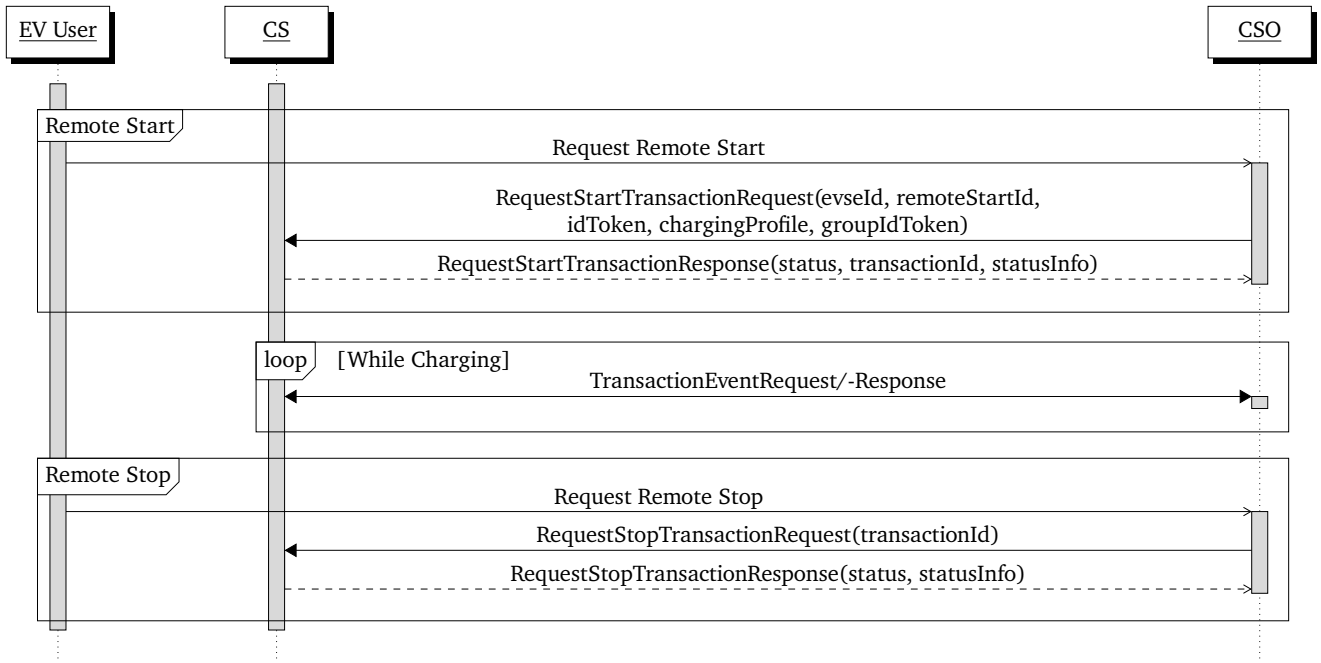


Figure 2.16.: Remote Start and Stop of an OCPP Charging Session

its availability once the user arrives. For this, the CSO sends a *ReserveNowRequest* message to the CS, including, inter alia, the EV user's ID Token. The CS reserves an appropriate EVSE/connector for the user until the reservation expires, i.e., if the CS has multiple EVSEs/connectors available, other users (identified by their ID Token) can still charge at the CS as long as at least one fitting EVSE/connector remains available. Only a user with the reserved ID Token or (if available) an ID Token in the same group can use the reservation.

2.4.3. Roaming Protocols

The scenario for EV roaming is as follows: An EV user has an existing e-mobility contract with an eMSP and the EV user wants to charge their EV at a CS that is not operated by this eMSP. Enabling EV roaming requires a preexisting roaming agreement between the EV user's eMSP and the concerned CSO as well as the exchange of the data that is required of charge authorization (e.g., ID Tokens) and billing (e.g., meter values) between the backend systems of eMSP and CSO. In order to handle the mentioned data exchange, a CCH is introduced as intermediary to route messages between sets of eMSPs and CSOs. The main argument for the inclusion of a CCH is that it can reduce the complexity of interconnecting a large amount of actors (e.g., by providing standard communication interfaces and standard roaming contracts) when compared to a peer-to-peer approach [107].

Several protocols exist to handle the communication needs of the EV roaming use case. This section considers a selection of EV roaming protocols, namely: OCHP [179] (with its extension OCHPdirect [178]), OICP [88, 87], eMIP [75], and OCPI [142]. The selection is based on [106], which considers these protocols to be the most widely used in Europe that have a public and complete documentation and are implementable by any party. In general, all surveyed roaming protocols support the use of a CCH as intermediary. Only OCPI and OCHP via its OCHPdirect extension also support direct peer-to-peer communication between CSO and eMSP.

Table 2.3.: EV Roaming Protocol Functionality

	OCHP w/o OCHPdirect	OCHP w/ OCHPdirect	OICP	eMIP	OCPI
Roaming Authorization					
Whitelist Authorization	✔	✔	(✔)*	✔	✔
Live Authorization	✘	✘†	✔	✔	✔
Remote Authorization	✘	✔	✔	✔	✔
Reservations	✘	✔	✔	✔	✔
Charge Session Data					
CDRs for Billing	✔	✔	✔	✔	✔
Live Session Data	✘	✔	✘	✘	✔
CS Information					
Static CS Data	✔	✔	✔	✔	✔
Live CS Status	✔	✔	✔	✔	✔

Protocol versions: OCHP 1.4 [179], OCHPdirect 0.2 [178], OICP 2.2 [88, 87], eMIP 1.0.13 [75], OCPI 2.2-d2 [142]

*Limited to CCH (cf. [87], Section 3).

†Planned for OCHP version 1.5 (cf. [177]).

All roaming protocols support the basic functions of authorizing charging sessions (based on ID Tokens) with the possibility of reserving EVSEs, reporting billing relevant data of a charging session (based on Charge Detail Records (CDRs)), and publishing CS information (e.g., tariffs, location, and availability) and only the specific implementation of these functions varies slightly per protocol. Table 2.3 provides a summary of the functionalities found in the surveyed protocols. The following text details the findings:

Roaming Authorization: In order for a CSO to allow an EV user to charge at one of their CSs, the CSO first needs to know if the user is authorized to do so. In the case of roaming, the CSO needs the respective eMSP's confirmation for this authorization. In all existing protocols, this authorization is implemented via ID Tokens, i.e., the eMSP generates a list of authorized ID Tokens (corresponding the EV users that the eMSP has a contract with) and the CSO verifies that an ID Token is present in one of those lists in order to confirm charge authorization. Existing protocols generally provide three different (not mutually exclusive) methods for this process of authorization, namely:

1. **Whitelist Authorization**: The eMSP uploads the list of authorized ID Tokens to the CCH (either completely or as differential update). Depending on the protocol, the CCH can push the new list (full or update) to concerned CSOs and/or the CSO can regularly pull a new list (full or update) form the CCH. The CSO may push this list to their CSs over OCPP as Local Authorization List (outside the scope of roaming protocols). When the CSO receives a request for charge authorization from one of their CSs, they can first check if the received ID Token is authorized in their local whitelist. If this is not the case, the CSO can forward the request to the CCH and the CCH checks if the ID Token is authorized based on their local whitelist (the CCH might, e.g., have newer information that was not yet pulled by the CSO). Most roaming protocols support the Whitelist Authorization approach fully. The exception is OICP, which only supports the push of an ID Token list from eMSP to CCH, i.e., it does not enable CSOs to receive this list and thus limits Whitelist Authorization to the CCH.

2. **Live Authorization**: The CSO sends an authorization request for a specific ID Token (over the CCH) to the eMSP. If the eMSP's ID is known (e.g., if an eMAID[16] is used as ID Token), the CCH can directly forward the request to the right eMSP. If the eMSP's ID is not known (e.g., if an RFID UID is used as ID Token), the CCH has to iterate over all eMSPs with roaming agreement to the requesting CSO. The eMSP answers with the respective authorization status. Some protocols automatically revert to Live Authorization if the Whitelist Authorization approach failed, e.g., in OICP the CCH tries to send a Live Authorization request to the eMSP if it could not authorize the ID Token based on its whitelist. All roaming protocols except for OCHP currently support the Live Authorization approach.[17]

3. **Remote Authorization**: An eMSP authorizes the start of a charging session for an EV user at an EVSE. For this, the eMSP sends a request, including the user's ID Token and target EVSE, (over the CCH) to the corresponding CSO. The Remote Authorization function is used by the eMSP to start charging sessions on behalf of the EV user, for instance, to enable charge authorization via a smartphone app. Similar to the Remote Authorization function is the Remote Reservation function whereby the eMSP reserves an EVSE for a user (i.e., with the user's ID Token as an implicit authorization) for a specified duration. The main difference is that a Remote Reservation usually requires the user to authenticate oneself at the EVSE (by presenting their authorized ID Token) whereas the Remote Authorization is usually intended for an already connected EV, i.e., authorizes to start a charging session with the connected EV. All roaming protocols support the Remote Authorization/-Reservation functions (OCHP only with the OCHPdirect extension).

Charge Session Data: For a roaming charge session, the CSO bills the eMSP (based on the conditions of their roaming contract and session data from the CS) and the eMSP bills the EV user (based on the conditions of their e-mobility service contract and session data from the CSO), which might entail two different pricing systems. Hence, for each charging session, the CSO must inform the eMSP about all billing relevant information. For this, all surveyed roaming protocols use the concept of CDRs. A CDR describes the billing relevant information of a single session. Generally, a CDR includes the start/end time of the session, the ID Token of the EV user, an ID of the used EVSE, the relevant meter values, the total transferred energy, and the total cost. Additionally, CDRs identify the tariff(s) that are applicable for the session (e.g., if the CSO uses time- or load-based tariffs) as well as additional fees (e.g., for parking time or a fixed base fee added to all sessions). Depending on the protocol, CDRs are sent from the CSO (over the CCH) to the eMSP via a (near) real-time push and/or uploaded to the CCH from where they are regularly pulled by the eMSP. Some protocols also include a method for the CSO to inform the eMSP of the current status of a charging session, i.e., provide data that is not billing-relevant but might be interesting to the EV user (e.g., periodic meter values or smart charging information in order to track the charging status and/or accumulating cost via a smartphone app). All protocols provide the ability to send CDRs and OCPI as well as OCHP with OCHPdirect additionally enable a CSO to send live charge session status data.

CS Information: In order to enable the previously mentioned Remote Reservation function and to offer navigation/routing services, eMSPs need information on existing CSs. CS Information is generally divided into the categories: Static CS Data and Live CS Status. Static CS Data includes, inter alia, information on the CS' location, exiting EVSEs and connectors, charging capabilities (AC or DC, maximum power, etc.), applicable tariffs (for charging, parking, etc.), and supported payment options (RFID, PnC,

[16] eMAIDs always include the eMSP's ID (cf. [97], Annex H).

[17] Note that support for Live Authorization is planned for the next version of OCHP, version 1.5, via an update to the OCHPdirect extension (i.e., using a direct request from the CSO to the eMSP) [177].

etc.). Live CS Status on the other hand simply indicates the availability of the CS' EVSEs/connectors (i.e., available, occupied, reserved, etc.). All surveyed protocols support Static CS Data and Live CS Status.

2.5. Trusted Platform Module 2.0

The TPM 2.0 is a cryptographic co-processor that provides a security anchor for computer systems [4]. A TPM can be implemented as a separate hardware chip or as software running in a protected execution environment, i.e., in a special execution mode with hardware-based memory partitioning, of the host system's processor [195]. The TPM specification [195] is developed by the Trusted Computing Group (TCG) and has been adopted as the international standard ISO 11889 [95].

A TPM can provide several security-relevant functionalities including secure key generation, -storage, and -usage as well as more advanced security functions like measuring and reporting the host system's software integrity [4]. TPMs are the most popular form of trusted hardware, are recommended as a security anchor in vehicular communication controllers by security experts [54], and have already seen adoption in the automotive industry [93] making them an ideal security anchor for the EV charging use case.

2.5.1. Key Management

One of the most important functions of a TPM is key management. A TPM provides methods for secure key generation (cf. [195], Section 11.4), -storage (cf. [195], Section 11.7), and -usage (cf. [195], Section 11.4) such that the corresponding sensitive values never have to leave the TPM in clear text. By providing a standardized means of key management with strong security measures, a TPM can reduce the risk of security-critical implementation errors, while protecting the keys from being compromised even from manipulated software or side-channel attacks [4].

A TPM's keys can either be stored directly in a shielded location, i.e., in its internal Non-Volatile (NV) memory, or outside of the TPM's shielded location, using the protected storage mechanism (cf. [195], Section 22). The protected storage mechanism offers cryptographic protection for keys, ensuring their confidentiality and integrity. While TPM's internal memory can be very limited, the protected storage mechanism enables an effective expansion of this memory by enabling the secure storage of keys even outside of the TPM.

In particular, the TPM supports protected storage hierarchies (cf. [195], Section 23). A hierarchy starts with a primary seed, i.e., a random value that is persistently stored in the TPM (cf. [195], Section 14). A primary seed can be used to generate a primary key, in the case of a storage hierarchy, a primary storage key (or storage root key). Primary keys can be re-generated as long as the primary seed does not change or they can be persistently stored in the TPM's NV memory.

A (primary) storage key is a restricted decryption key, i.e., its secret key can only be used for decryption and is restricted to certain commands and specifically formatted input values (cf. [195], Section 25). These attributes allow it to be used as parent of other keys in the hierarchy, i.e., other (child) keys can be generated underneath it whereby the child key's private value is encrypted based on the parent. The encrypted private key of the child can then securely be stored outside of the TPM. If the child is itself a storage key, this mechanism allows the formation of multi-level key hierarchies.

The following paragraphs describe the relevant TPM key hierarchy management commands:

TPM2_CreatePrimary: This command creates and loads a primary key[18] under a primary seed (cf. [197], Section 24.1). The key is created based on a provided template that defines its type (e.g., an asymmetric Rivest-Shamir-Adleman (RSA) or Elliptic Curve Cryptography (ECC) key pair), parameters (e.g., key length), attributes (e.g., restricting the key's usage or its possibility to be exported of the TPM), and usage authorization (e.g., via password or policy; cf. Section 2.5.2). The command returns a transient handle to the loaded key, which enables the keys use in subsequent commands.

TPM2_EvictControl: This command can be used to persistently store a loaded key in the TPM's internal memory or evict a persistently stored key from the TPM's internal memory (cf. [197], Section 28.5). If a transient handle is provided as input, the corresponding key is persisted and a persistent handle to the key is returned. If a persistent handle is provided as input, the corresponding key is evict from memory.

TPM2_Create: This command is used to create a key underneath a loaded storage key as its parent (cf. [197], Section 12.1). The command takes as input a handle to the loaded parent key along with the public template for the new key (defining its type, parameters, attributes, and usage authorization). The private values of the newly created key are returned in encrypted form (using the parent with the protected storage mechanism for encryption) and the corresponding public values (the public template with the addition of the new public key) are returned in plaintext.

TPM2_Load: This command is used to load a previously generated key into the TPM (cf. [197], Section 12.2). The command takes a handle to the loaded parent key along with the child key's encrypted private values and plaintext public values as input. The loaded parent is used to decrypt the child's private values and a transient handle to the loaded key is returned. Since the parent is a storage key, i.e., a restricted decryption key, it cannot be used for general purpose decryption making it impossible to decrypt the child key's private values in any other way than with the `TPM2_Load` command.

TPM2_Import: This command is used to import keys into the TPM's hierarchy (cf. [197], Section 13.3). It can be used to move keys within a TPM's hierarchy or to import externally generated keys. In order to enable the import of a key, it's private values are encrypted based on a storage key in the TPM using the duplicate protection mechanism (cf. [195], Section 23.3).[19] The command takes as input a handle to the loaded storage key and the encrypted ("duplicated") private key along with the plaintext public values. The encrypted private key is decrypted using the storage key and then re-encrypted using the protected storage mechanism, i.e., the storage key serves as the new parent of the imported key. The re-encrypted private key is returned. Note that the duplicate protection mechanism includes a hash over the public values of the "duplicated" key[20] in both the en- and decryption processes in order to ensure the integrity of these values, i.e., a key cannot be imported if wrong public values are provided. Since the new parent is a storage key, i.e., a restricted decryption key, it cannot be used for general purpose decryption making it impossible to decrypt the encrypted ("duplicated") private key without using the `TPM2_Import` command.

A loaded key can be used by the TPM in further commands. Depending on the key's type and attributes, possible commands may be restricted. The following paragraphs describe the relevant TPM key usage commands:

[18] In general, the TPM's create commands can be used to create TPM protected objects. Objects can be keys or data that needs to be TPM protected. In the context of this thesis, keys are the only relevant objects.

[19] The duplicate protection mechanism allows different methods of encryption that provide different security guarantees (including no encryption, i.e., a plaintext private key). In the context of this thesis only the "Outer Duplication Wrapper" method (cf. [195], Section 23.3.2.3), i.e., encryption based on a symmetric key and protection of the symmetric key based on the parent storage key (hybrid encryption), is relevant.

[20] A hash over the public values of a key is referred to as the key's "name" in the TPM specification and used as a unique identifier (cf. [195], Section 16).

TPM2_Sign: This command is used to create a signature over an externally provided hash (cf. [197], Section 20.2). It takes as input a handle to a loaded signing key (an asymmetric key which is allowed to create signatures based on it public template attributes) and a digest to be signed. The TPM calculates a signature over the digest based on the private signing key and this signature is returned. If the signing key has the restricted attribute set, it is only possible to sign TPM-generated hashes.

TPM2_HMAC: This command is used to create a Hash-based Message Authentication Code (HMAC) over externally provided data (cf. [197], Section 15.5). It takes as input a handle to a loaded HMAC key (a symmetric key which is allowed to create keyed hashes based on it public template) and a buffer with arbitrary data. The TPM calculates an HMAC over the data based on the loaded HMAC key and this HMAC is returned.

TPM2_Certify: This command is part of the TPM's attestation capabilities, i.e., the TPM is used as root of trust for reporting in order to sign an internally generated data structure (cf. [197], Section 18). The `TPM2_Certify` command in particular is used to prove to an external verifier that a specific key is loaded in the TPM (cf. [197], Section 18.2). The command takes as input a handle to a loaded signing key, a handle to the loaded key that is to be certified, and arbitrary qualifying data. The contents of the qualifying data can be defined by the caller and, for instance, used to ensure freshness by including a nonce (cf. [195], Section 31.4). The TPM builds an attestation data structure including, inter alia, the qualifying data and a hash over the public values of the key that is to be certified (cf. Table 2.4 and [196], Section 10.12). The attestation data structure is then hashed and signed by the TPM with the private signing key and returned along with the signature. Note that the amount of confidence a verifier can place in the attestation strongly depends on the signing key (cf. [195], Section 31.1). For example, if the signing key does not have the restricted attribute set, i.e., if it can sign digests that were not generated by the TPM, it is possible to sign forged attestations using the general purpose `TPM2_Sign` command.

2.5.2. Enhanced Authorization

TPMs support a comprehensive concept for managing key usage authorization called "enhanced authorization" (cf. [195], Section 19.7). Enhanced authorization makes it possible to condition the authorization to use a TPM-protected key on a set of tests, which is encapsulated in the key's "authorization policy". For example, authorization could be conditioned on an assertion that a selection of the TPM's Platform Configuration Registers (PCRs) (shielded locations for transient data, commonly used to store measurements of the host system's software state; cf. [195], Section 11.6.2) contain specific values (using the `TPM2_PolicyPCR` command).

An authorization policy is represented as a hash-chain digest, starting with an all zero digest. The `TPM2_PolicyPCR` command, for instance, sets the new policy digest to a hash over the current policy digest appended with a static identifier of the `TPM2_PolicyPCR` command, a bitmap identifying the selected PCRs, and a digest of the selected PCRs. A key's authorization policy is defined at its creation via the key's public template. In order to satisfy a key's policy, i.e., to authorize its usage, a policy session must be built in the TPM and the sessions policy digest must be equal to the key's authorization policy digest.

An example of a sophisticated policy, ensuring the host system's software integrity while enabling firmware upgrades, is demonstrated in [68]. The authors of [68] use a `TPM2_PolicyPCR` command in order to condition the policy on the host system's software integrity. The host system's software integrity is measured during its boot, i.e., using a measured boot. During a measured boot, each piece of code in the boot chain

Table 2.4.: TPM2_Certify Attestation Data Structure (cf. [196], Section 10.12)

Parameter	Description
magic	An indicator that the structure was generated by the TPM. It is used to prevent a restricted signing key from signing a forged attestation. Restricted signing keys can only sign TPM-generated hash digests and the general purpose TPM hashing command does not provide a valid digest (meaning a digest that can be signed with a restricted key) if the input starts with the "magic" indicator.
type	Specifies the type of the attestation data structure (e.g., `TPM_ST_ATTEST_CERTIFY` in case of the `TPM2_Certify` command).
qualifiedSigner	Qualified name of the signing key. A key's name is a hash over its public values. A key's qualified name is recursively defined as a hash over the qualified name of the key's parent appended with the key's name (cf. [195], Section 26.5). The recursion stops with the primary seed as the qualified name of a primary seeds is equal to their handle, which is statically defined (cf. [196], Section 7.4).
extraData	Contains the supplied qualifying data.
clockInfo	Contains information on the TPM's clock (milliseconds that advanced while the TPM was powered), reset-, and restart counter (cf. [195], Section 36.3)
firmwareVersion	A vendor-specific identifier of the TPM's firmware.
attested	Contains the type-specific attestation information (e.g., "name" and "qualifiedName" in case of the `TPM2_Certify` command).
name	The name of the certified key.
qualifiedName	The qualified name of the certified key.

measures, i.e., calculates a hash over, the next piece of code before executing it [156]. The respective measurements are stored in the TPM's PCRs. The measured boot process is started at the core root of trust for measurement, i.e., the initial boot code that starts the measurements but itself cannot be measured and must be trusted in order for the resulting measurements to be trusted (cf. [195], Section 34).

However, since a key's policy is provided during generation and cannot be changed afterwards, directly conditioning a key on a `TPM2_PolicyPCR` policy would prevent the key's usage in case of a legitimate firmware upgrade. In order to enable later firmware upgrades, [68] instead sets the direct authorization policy to a `TPM2_PolicyAuthorize` command. The `TPM2_PolicyAuthorize` command enables the definition of a new policy that can be used to authorize key usage after a key's initial creation. For this, the `TPM2_PolicyAuthorize` command specifies a public key for the direct authorization policy. Afterwards, any policy digest that is signed, i.e., authorized, by the corresponding private key, can serve to authorize key usage. By authorizing, i.e., signing, a `TPM2_PolicyPCR` policy digest, a key's usage authorization can be conditioned on the host system's software integrity while supporting firmware upgrades (every firmware upgrade requires a new authorized `TPM2_PolicyPCR` policy digest) [68].[21]

[21] The authors of [68] additionally include a `TPM2_PolicyNV` command in authorized policies in order to prevent downgrade attacks. While downgrade attacks are out of scope for this thesis (cf. Section 5), the following text briefly summarizes the downgrade prevention mechanism from [68]: In general, the `TPM2_PolicyNV` command can be used to condition a policy on the contents of the TPM's NV memory. The authors of [68] use the TPM's capability of implementing strongly monotonic counters in its NV memory, i.e., counters that can only be incremented and not directly written to, reset, or decremented. A strongly monotonic NV counter is used by [68] to represent the current firmware version and each authorized policy is conditioned on the counter's value being smaller or equal to the current firmware version. After a firmware upgrade, the NV counter is incremented accordingly, which prevents the reuse of an old authorized policy after a downgrade attack.

The relevant enhanced authorization commands are summarized in the following paragraphs:

TPM2_StartAuthSession: This command is used to start a policy session (cf. [197], Section 11.1). A policy session is used to build a policy digest in the TPM, starting with a zero digest. The policy digest can subsequently be used to provide authorization for key usage. The command returns a handle to the started session. The returned handle can be used in following policy commands (to build the policy digest) or in key usage commands (to provide authorization).

TPM2_PolicyAuthorize: This command is used to enable the authorization of a different policy digest after the creation of a key (cf. [197], Section 23.16). The command takes as input a handle to a policy session, the authorized policy digest, the name of the key that signed the authorized policy digest, and a "ticket"[22] proving that a signature over the authorized policy digest was verified by the TPM with the public key that is identified by the provided name. The TPM verifies the ticket and checks that the provided authorized policy digest is equal to the current digest of the policy session. If both validations succeed, the current digest of the policy session is reset to the all zero digest and afterwards extended with the provided signing key's name. Hence the resulting policy digest is independent of the previous digest.

In order to generate the required verification ticket, additional commands are required, namely:

TPM2_LoadExternal: This command is used to load a key that is not protected by the TPM into the TPM (cf. [197], Section 12.3). The command is supplied with the public signing key that shall be used to verify the signature over the authorized policy digest. The TPM loads the key and returns the key's name as well as a transient handle to it.

TPM2_VerifySignature: This command is used to verify a signature with the TPM (cf. [197], Section 20.1). The command is provided with the handle of a loaded key, a digest, and a signature over the digest. After a successful verification, the TPM returns a ticket, which proofs to itself proving that the provided digest was verified with the provided public key.

TPM2_PolicyPCR: This command is used to verify that the contents of the TPM's PCRs are as expected (cf. [197], Section 23.7). The command takes as input a handle to a policy session and a data structure that identifies a set of PCRs. The TPM creates a hash over the selected PCRs and extends this hash (along with a data structure that identifies the set of used PCRs) to the current policy session digest. The command optionally takes the expected PCR digest as an input and fails if it is not equal to the calculated PCR digest. If the expected digest is not provided the failure occurs when trying to use the resulting policy for authorization. Additionally, the command has a deferred assertion when the resulting policy session is used for authorization, which verifies that the PCRs have not changed since the execution of the `TPM2_PolicyPCR` command (cf. [195], Section 19.7.7).

2.5.3. Credential Protection

TPMs supports a credential protection protocol, which enables a credential provider (e.g., a CA) to provide a credential (e.g., a public key certificate) to a specific key while ensuring that this key belongs to an authentic TPM (cf. [195], Section 24). The initiator of the protocol (e.g., the TPM-equipped host) sends the credential provider two TPM public keys. One of the keys is a public storage key and can be verified by the credential

[22] The ticket is an HMAC that is generated using a TPM internal secret key over a data structure including the authorized policy digest and the signing keys name. The ticket is generated by the `TPM2_VerifySignature` command.

provider (e.g., via a public key certificate). The second key is the one for which the initiator requests a new credential.

On receiving these two keys, the credential provider verifies that the public storage key belongs to an authentic TPM (e.g., by validating its certificate and trusting the issuer to only provide certificates for authentic TPMs) and creates a credential for the second key. Afterwards, the credential provider encrypts the newly created credential using the credential protection mechanism (cf. [195], Section 24.3 ff.), i.e., using hybrid encryption based on the public storage key. More precisely, a symmetric session key is generated and used to encrypt the credential and to generate an HMAC over the encrypted credential. The symmetric key itself is protected based on public storage key (e.g., in the case of an ECC storage key using ECDH with the public storage key and a private ephemeral key). Note that the process of encrypting the credential includes a hash over the second public key, i.e., the key for which the credential was generated.

The encrypted credential is sent to the initiator. The initiator can use its TPM to decrypt the credential based on the private storage key. Decryption is only possible if both the storage key and the second key, i.e., the key for which the credential was generated, are currently loaded in the TPM. Hence, decryption is only possible if the second key belongs to the same TPM as the storage key, i.e., to the same TPM that the credential provider believes to be authentic.

The decryption of protected credentials is implemented using the following command:

TPM2_ActivateCredential: This command enables the decryption of a protected credential that is associated with a specific TPM key (cf. [197], Section 12.1). It is used to implement the TPM's credential protection protocol (cf. [195], Section 24). The command takes as input the handle of a storage key (the public key of which was used to encrypt the credential), the handle of a second key (which is associated with the credential and the hash of which was used in the encryption process), and the encrypted credential. The TPM uses the storage key to recover the symmetric session key (e.g., in the case of an ECC storage key using ECDH for which the public ephemeral key of the credential provider is additionally required). The symmetric key is then used to verify the HMAC over the credential and to decrypt it. Note that the process of decrypting the credential requires the inclusion of a hash over the public values of the second key, which must be the same as used during encryption. Finally, the decrypted credential is returned. Note that the length of the decrypted credential is restricted to the output length of the TPM's largest supported hash algorithm (e.g., 512 bit for SHA512). For this reason, the credential protection method is commonly used to encrypt a symmetric key and this symmetric key is used to encrypt the actual credential (cf. [195], Section 24.3).

2.5.4. Direct Anonymous Attestation

DAA is a signature scheme that allows the generation of anonymous signatures (different signatures of the same entity cannot be linked) as well as pseudonymous signatures (different signatures of the same entity may be linked via a pseudonym) (cf. [195], Section C.4.2). DAA was developed in order to enable the authentication of a TPM while preserving the privacy of the corresponding host platform owner, i.e., a verifier should only learn that a TPM is used but not which particular TPM in order to prevent linkability [22].

Historically, the TCG first developed the privacy CA scheme in order to provide privacy-preserving TPM authentication [22]. For the privacy CA scheme [194], a TPM uses a different key pair (called Attestation Identity Key (AIK)) for each authentication in order to prevent linkability. AIKs are certified by a "privacy CA" such that a verifier can validate the authenticity of these AIKs. The requests for AIK certificates are protected based on a static TPM key pair (called Endorsement Key (EK)), which is trusted by the "privacy CA". The

privacy CA scheme has the drawbacks that it requires a TPM to get a new AIK certificate for each transaction, i.e., the privacy CA must be highly available, and that it enables the privacy CA to still link a TPM's AIKs, i.e., if the privacy CA's records are leaked to the verifier, then the verifier can link different authentications to a TPM [22]. The RSA-based DAA scheme [22] aimed to address these drawbacks and was adopted by the TCG in their TPM 1.2 specification [193].

In general, DAA is a special kind of group signature scheme [22]. In a group signature scheme only the members of a group can create valid signatures and a verifier can only validate that a signature was created by a member of the group but not by which specific member, i.e., the identity of the signer is kept secret [30]. While group signatures usually include a method of "opening" a signature, i.e., a method to reveal the exact identity of the signer (e.g., via a trusted third party) [30], DAA signatures cannot be opened [22]. Additionally, since DAA is designed for the use with a TPM, which has very restricted storage and computational resources, the signer role is split between the TPM and the corresponding host, whereby the TPM still holds the private key but the host is able to aid in the signature computation [22]. Furthermore, DAA can provide different levels of privacy by allowing signatures to be not at all linkable or selectively linkable [22]. Linkability is possible by providing a "basename" during signature creation, whereby signatures can only be linked if they were generated with the same basename [22].

Due to the high overhead in terms of performance, signature-/key lengths, and code size of the RSA-based DAA scheme [22, 31], several ECC-based DAA schemes, promising better efficiency than the RSA-based ones, have been proposed (e.g., [24, 214, 33]); the TPM 2.0 specification [195] supports some of these ECC-based schemes [213]. We focus on the DAA scheme from [31] as extended in [205] as it is one of the most recent DAA schemes with TPM call-level details, is implementable with the current TPM specification, and since it provides symbolic security proofs including low-level TPM call details [205].

The DAA scheme involves three different kinds of actors, namely issuers, signers, and verifiers [31, 205]. The issuer is responsible for verifying the legitimacy of signers and for issuing DAA credentials to signers. Thus, the issuer is the manager of a DAA group and can allow new members to join the group. A signer, consisting of TPM and corresponding host, can use the provided DAA credentials to prove to a verifier that they are a member of the issuer's group. The process of proving group membership neither reveals the signers credentials nor their identity. In particular, the DAA scheme involves five protocols, namely [31, 205]:

Setup: The system parameters (including security parameter, bilinear group pair, and generators) are chosen and the issuer generates a group key pair. The system parameters and group public key are published and available to signers and verifiers.

Join: In order to join an issuer's group, a signer creates a DAA key pair in its TPM and receives the corresponding DAA credentials from the issuer.

Sign: A signer uses its private DAA key together with its DAA credential and an optional basename to create a signature of a message. Specifically, the TPM creates a signature using the private DAA key and the host converts the signature using the DAA credential. All signatures that were generated with the same combination of private DAA key and basename can be linked,[23] i.e., the DAA signing scheme provides user-controlled linkability.

Verify: A verifier is provided with a message, basename, and DAA signature. The verifier can decide whether the signature is valid or not.

[23] Part of an ECC-DAA signature is a deterministic signature created with the private DAA key over the basename. This deterministic signature serves as a pseudonym to enable linkability.

Link: Given two signatures, a verifier can link them if they were generated based on the same DAA key and under the same basename. If signatures are generated with either different keys, different basenames, or no basename, linking is not possible.

Given a secure channel between TPM and host and an insecure channel, i.e., an adversary controlled channel, between host and issuer/verifier, the TPM-based ECC-DAA scheme provides the following security properties [205]:

Correctness: The scheme is consistent and can execute correctly in the absence of an adversary.

User-controlled linkability: Using the same private key and the same basename, it is hard to create two signatures without the signatures being linked.

Unforgeability: Given a set of private keys and credentials, it is hard to forge a valid signature for a private key and credential not in this set.

Non-frameability: No combination of dishonest issuers and signers can create a new, valid signature that can be linked to a signature generated by an honest signer.

Anonymity: Without knowing a signer's private key, it is hard to recover the identity of the corresponding TPM from a signature.

User-controlled unlinkability: Given two signatures generated using different basenames, it is hard to tell if the signatures originate from the same signer.

Agreement during the join: Both the signer and issuer confirm each other's identity and agree on the credentials provided during the join protocol.

More detail on the join and sign/verify protocols of the TPM-based ECC-DAA scheme [205] is provided in the following sections.

ECC-DAA Join Protocol

For the DAA join protocol, the signer is equipped with certified endorsement credentials from its manufacturer. More specifically, the TPM stores the private endorsement credential key EC_{sk} and the host stores the public endorsement credential key EC_{pk} (or endorsement credential certificate). Note that an endorsement key pair (EC_{pk}, EC_{sk}) is a restricted decryption key pair, i.e., a storage key. Additionally, EC_{pk} (or the endorsement credential certificate) is available to the issuer and the issuers group public key I_{pk}^{group} has already been published in the previous setup process. Fig. 2.17 shows the join protocol, whereby the available data per entity at the beginning of the protocol is shown directly beneath each entity except for the public system parameters (including security parameter, bilinear group pair, and generators), which are available to all entities but not shown.

The join process starts with the host instructing its TPM to generate a DAA key pair as a restricted signing key (cf. Create($DAA_{template}$) in Fig. 2.17). Note that, for simplicity, the "TPM2_" prefix of all TPM commands in Fig. 2.17 (and following) is omitted. Additionally, a key's abbreviation without its public/private subscript (e.g., EC) is used as key handle and the details of TPM key management (cf. Section 2.5.1) are omitted, i.e., private keys are shown to remain in the TPM instead of explicitly loading them.

After the generation of the DAA key pair (DAA_{pk}, DAA_{sk}), the host sends EC_{pk} and DAA_{pk} to the issuer. The issuer validates EC_{pk} (based on the endorsement credential certificate), generates a nonce c, and encrypts c using the TPM's credential protection mechanism (cf. [195], Section 24; as previously described in

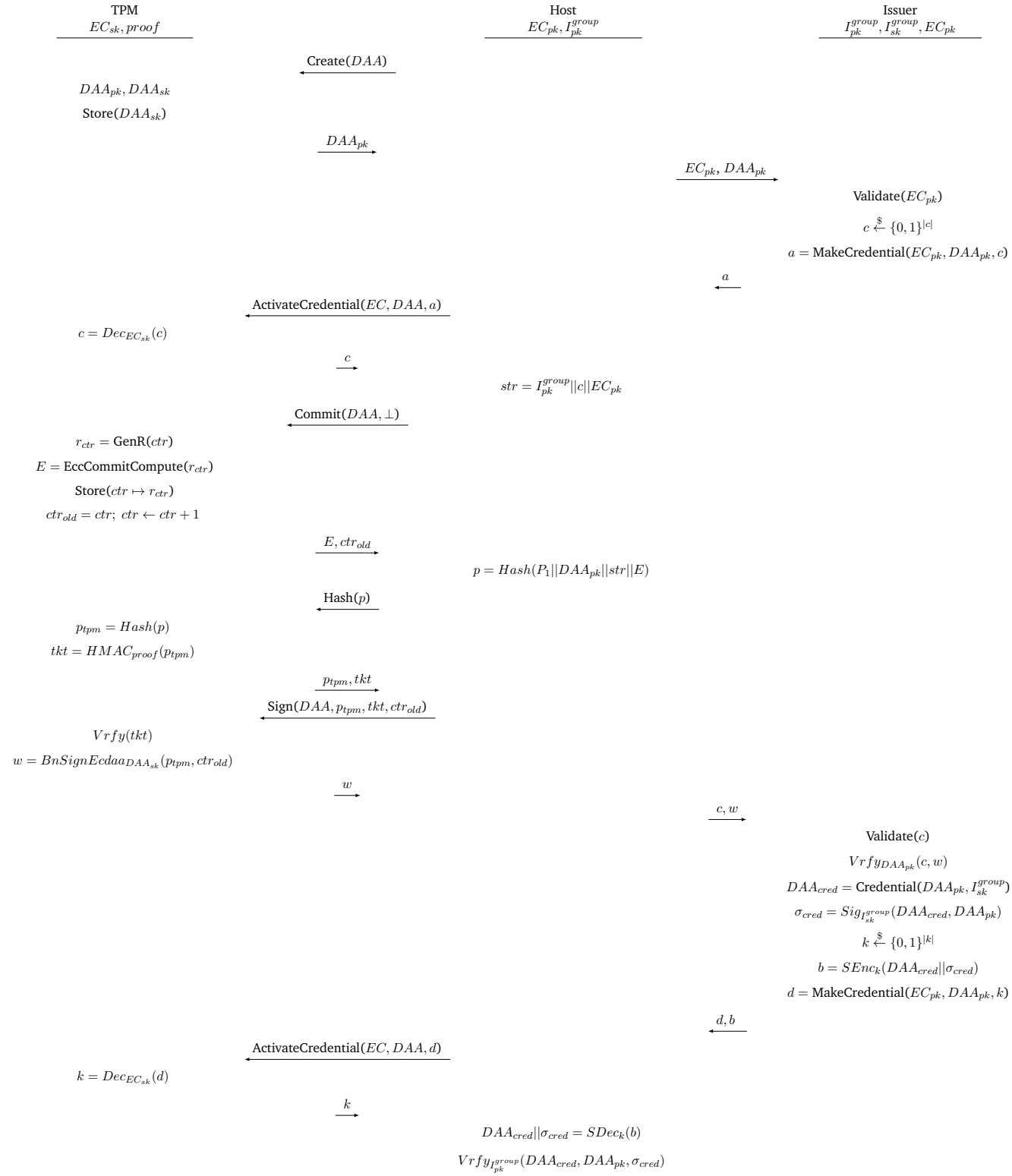


Figure 2.17.: TPM-Based ECC-DAA Join Protocol [205]

Section 2.5.3; called MakeCredential in Fig. 2.17). For the credential protection-based encryption of c, EC_{pk} serves as the storage key and DAA_{pk} serves as the key that is associated with the credential. The result of the encryption, the ciphertext a, is returned to the host.

The host uses its TPM to decrypt a with EC_{sk} using the `TPM2_ActivateCredential` command and receives c. Note that, since the credential protection mechanism is used, a can only be decrypted with EC_{sk} if additionally the DAA key pair (DAA_{pk}, DAA_{sk}) is currently loaded in the TPM (as mentioned, the loading of keys is not explicitly shown in Fig. 2.17). Afterwards, the host instructs its TPM to sign $str = I_{pk}^{group}||c||EC_{pk}$ using DAA_{sk} in order to provide a proof of possession of DAA_{sk} to the issuer.

In more detail, creating the DAA signature requires three TPM commands, namely `TPM2_Commit` and `TPM2_Sign` for DAA signature creation as well as `TPM2_Hash` to generate the input hash for `TPM2_Sign` since the DAA key pair is a restricted signing key, i.e., it can only sign a TPM-generated hash. The following paragraphs describe the TPM commands with respect to their usage in the DAA scheme:

TPM2_Hash: This command is used to calculate a hash over arbitrary data in the TPM (cf. [197], Section 15.4). The command takes as input a data buffer and a hash algorithm. The TPM calculates a hash over the contents of the data buffer using the indicated hash algorithm. Additionally, the TPM verifies that the contents of the data buffer do not start with the fixed byte sequence that indicates the content as TPM generated (called "magic" for a TPM generated attest data structure in Table 2.4). Only if the content does not start with this identifier, the TPM calculate a "ticket" proving that the hash was calculated by itself and that the content did not start with the identifier (cf. [195], Section 11.4.5.3). The ticket is calculated as an HMAC over the digest using a TPM internal symmetric key (called hierarchy $proof$; cf. [195], Section 14.4). A restricted signing key can only be used to sign a hash (with `TPM2_Sign`) if a valid ticket is provided.

TPM2_Commit: This command is the first part of the DAA signature calculation (cf. [197], Section 19.2). The command takes as input the handle of a loaded DAA key. Additionally, the command can be supplied with three optional values P1, s2, and y2. The value P1 is used to provide a part of the signer's DAA credential. The values s2 and y2 are used to provide a basename for the DAA signature. The TPM starts by generating a random value r_{ctr} for its current commit counter ctr, increments the commit counter, and uses r_{ctr} to calculate the commit value E. If P1 was provided, it is included in the calculation of E (see [205] for details on the corresponding point multiplications). If a basename was supplied, the TPM additionally calculates the signature pseudonym based on DAA_{sk} and the basename values. The TPM returns the calculated values and the old commit counter ctr_{old}. The commit values can be used by the host for further computations before calling the `TPM2_Sign` command to finish the DAA signature (cf. [197], Section 19.1).

TPM2_Sign: As described in Section 2.5.1, this command is used to create a signature over an externally provided hash (cf. [197], Section 20.2). For a DAA signature, this command additionally takes a commit counter value (ctr_{old} as output by `TPM2_Commit`) as input. The commit counter value is used to recover the same random value (r_{ctr}) that was generated in the `TPM2_Commit` command. The random value is used during signature generation with DAA_{sk} (see [205] for details on the corresponding point multiplications).

First, `TPM2_Commit` is used with empty ($\perp$) arguments to generate the random value r_{ctr} and the commit value E. The host builds a hash over $P_1||DAA_{pk}||str||E$, whereby P_1 is one of the generators from the public system parameters, and uses the TPM to calculate a hash over the result in order to receive the digest p_{tpm} and the HMAC ticket tkt. Finally, the host instructs its TPM to sign p_{tpm} with DAA_{sk} resulting in w.

The host sends the decrypted nonce c and the DAA signature w to the issuer. The issuer validates that c is correct nonce and verifies the DAA signature, which requires the recalculation of str (as $I_{pk}^{group}||c||EC_{pk}$), E (based on parts of the signature, P_1, and DAA_{pk}), and p_{tpm} (as $hash(hash(P_1||DAA_{pk}||str||E))$) by the issuer (see [205] for details). Note that the issuer can now be sure that the DAA key pair (DAA_{pk}, DAA_{sk}) was generated in the same TPM as the trusted endorsement key pair (EC_{pk}, EC_{sk}), since otherwise the TPM would not be able to decrypt c with the `TPM2_ActivateCredential` command.

Afterwards, the issuer creates the DAA credential DAA_{cred} for DAA_{pk} using I_{sk}^{group} and signs the DAA_{cred} with I_{sk}^{group}. The DAA_{cred} is a Camenisch-Lysyanskaya signature [26] over the private DAA key DAA_{sk} (which can be calculated by the issuer blindly, i.e., without knowledge DAA_{sk}, based on the public key DAA_{pk} [23]) using the issuer's private key I_{sk}^{group} [31]. Note that one of the properties of this signature is that DAA_{cred} can be transformed into a different signature on DAA_{sk} ($rDAA_{cred}$) without knowledge of the private key I_{sk}^{group} [24]. That is, DAA_{cred} can be randomized (by multiplying the values of the credential with a random number) into a different valid credential $rDAA_{cred}$ on DAA_{sk}, which cannot be linked to the original credential or to a different randomized credential, while still being verifiable using the issuer's public key I_{pk}^{group} [31].

The DAA credential DAA_{cred} concatenated with the signature σ_{cred} are symmetrically encrypted ($SEnc$) by the issuer using a newly generated symmetric key k resulting in the ciphertext b. Subsequently, k is encrypted using the TPM's credential protection mechanism (cf. [195], Section 24; as previously described in Section 2.5.3), again using EC_{pk} as the storage key and DAA_{pk} as the key that is associated with the credential, resulting in the ciphertext d. Finally, the issuer sends d and b to the host.

The host uses its TPM to decrypt d with EC_{sk} using the `TPM2_ActivateCredential` command and receives k. Afterwards, the host uses k to decrypt b, resulting in DAA_{cred} and σ_{cred}. Finally, the host verifies the credential signature σ_{cred} using the issuer's public key I_{pk}^{group}.

Note that the inclusion of EC_{pk} in the TPM's signature w, i.e., the inclusion of EC_{pk} in the input hash p_{tpm} via its inclusion in str, is one of the extensions from [205] over [31]. This inclusion addresses the attack shown in [207], i.e., a Man-in-the-Middle (MitM) attack on the join process. The attack is possible if a single TPM is corrupted, i.e., if a TPM's endorsement key is leaked and the issuer is unaware of this leak, after which the join process cannot guarantee that the issuer authenticates any signer and the join process cannot guarantee the confidentiality of a signer's DAA credential [207].

The authors of [207] recommend the inclusion of EC_{pk} in the TPM's signature in order to prevent the attack, as adopted by [205]. In fact, the scheme from [205] provides the injective agreement property (based on the commonly used authentication definitions of Lowe [126]) during the join, which, according to [205], means in the DAA case that whenever a signer (identified by its EC_{pk}) completes a join with an issuer, then the issuer has been running the join in the issuer role with the signer and both the issuer and the signer agree on the exchanged credentials.[24]

ECC-DAA Sign/Verify Protocol

After a signer has received valid DAA credentials from an issuer, they can use these credentials for authentication via the DAA sign/verify process. Specifically, the TPM-based DAA sign process could be used to sign arbitrary

[24] Note that while the textual description of injective agreement in [205] as cited here appears to be missing the injective agreement requirement that each run of the signer corresponds to a unique run of the issuer (cf. [126]), the lemma used by [205] to prove the injective agreement property appears to include it (cf. `https://github.com/tamarin-prover/tamarin-prover/blob/522dda8ef7d91b0fe552a7c8e27ee864cbf807a1/examples/asiaccs20-eccDAA/CERTIFY/TPM_DAA_JoinCertify_with_fix_noBSN.spthy#L1948`; visited on 2020-12-14).

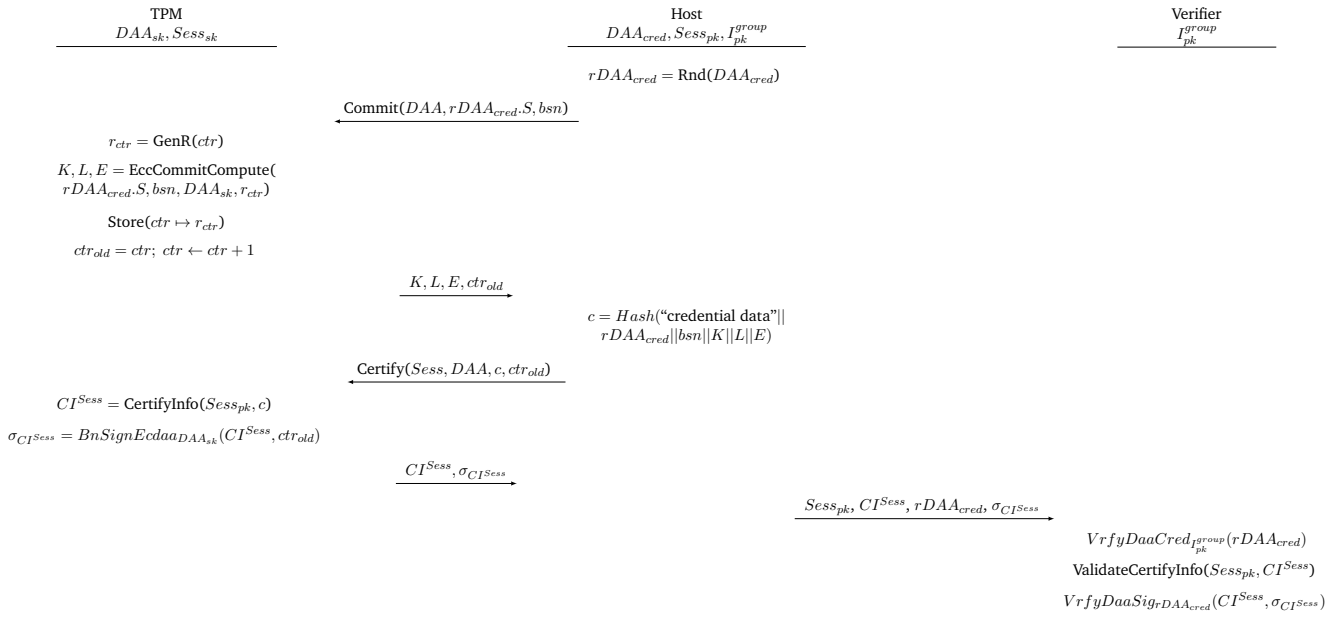


Figure 2.18.: TPM-Based ECC-DAA Sign/Verify Protocol Using TPM2_Certify [205]

data (using the `TPM2_Sign` command) or for one of the TPM's attestation capabilities (e.g., attesting another key in the TPM with the the `TPM2_Certify` command or using the `TPM2_Quote` command to attest the contents of the TPM's PCRs) [205]. We focus on the DAA-based certify case as it is the most relevant in the context of this thesis.

The DAA sign/verify process is executed between the signer (host and TPM) and a verifier (cf. Fig. 2.18). The verifier and host know the issuer's public key I_{pk}^{group}. The TPM stores the private DAA key DAA_{sk} and additionally a private sessions key $Sess_{sk}$, i.e., the key that is supposed to be certified. The host stores the DAA credential DAA_{cred} and the public session key $Sess_{pk}$.

The host starts by randomizing the credential DAA_{cred} into $rDAA_{cred}$. As described in Section 2.5.4, $rDAA_{cred}$ is still a valid credential for the DAA key and can be verified using I_{pk}^{group} [31]. Afterwards, the host calls the TPM with the `TPM2_Commit` command providing a part of its randomized credential (called S in [205]) via the P1 value and optionally a basename bsn via the s2 and y2 values. The TPM calculates the commit values K, L, E and the counter values ctr_{old} (K and L are only calculated if a basename is provided; K is always the same if the same DAA_{sk} and bsn are used). The host calculates the hash c over a static "credential data" string appended with $rDAA_{cred}$, bsn, and the commit values. The hash c serves as qualifying data in the `TPM2_Certify` command that is used to certify the session key pair $(Sess_{pk}, Sess_{sk})$ with DAA_{sk}. The TPM returns the certification data structure CI^{Sess} as well as the DAA signature ($\sigma_{CI^{Sess}}$) over it. The DAA signature $\sigma_{CI^{Sess}}$ in combination with the randomized credential $rDAA_{cred}$ provide a signature of knowledge and can be used to prove to a verifier that the signer has knowledge of a DAA_{sk} (without revealing DAA_{sk}) as well as knowledge of a valid credential for the same DAA_{sk} [32, 31].

Note that the certification data structure is slightly different for anonymous signature schemes (i.e., for ECC-DAA). Namely, the "qualifiedSigner" and "qualifiedName" fields (cf. Table 2.4) are left empty as the "qualifiedSigner" field would reveal the unique identity of the signing key and the "qualifiedName" field would make it possible to establish hierarchical relationships between certified keys (cf. [195], Section 31.5).

Additionally, qualifying data is not included in the "extraData" field of the certification data structure but instead appended to this data before it is hashed and the hash subsequently signed (cf. [195], Section C.4.2.3).

The host sends $Sess_{pk}$, CI^{Sess}, $rDAA_{cred}$, $\sigma_{CI^{Sess}}$ to the verifier. The verifier can verify the authenticity of the randomized DAA credential $rDAA_{cred}$ based on the issuer's public key I_{pk}^{group}. Afterwards, the verifier can check that CI^{Sess} is a valid certification data structure for $Sess_{pk}$. Finally, the verifier can verify the DAA signature over CI^{Sess} with $rDAA_{cred}$ and optionally link the signature if the same bsn was used multiple times (see [205] for details on the computations). If all verification are successful, then the verifier can be sure that the session key pair ($Sess_{pk}, Sess_{sk}$) was certified by a member of the issuer's group, i.e., the session key pair belongs to the same TPM as an issuer-certified DAA key pair.

2.5.5. Trusted Platform Module for ISO 15118

Previous work [66, 67], to which the author of this thesis contributed, has investigated the use of a TPM as a security anchor for ISO 15118's PnC mechanism. In order to integrate DAA into EV charging protocols, this thesis builds upon the contributions of [66, 67]. The relevant contributions of [66, 67] are summarized in this section.

In [66], the authors propose an extension to ISO 15118 enabling the generation of an EV's PnC keys in the EV's TPM, i.e., the EV's OEM provisioning credential key pair (PC_{pk}, PC_{sk}) and contract credential key pairs (CC_{pk}, CC_{sk}) are generated in its TPM.[25] For this, the ISO 15118 *CertificateInstallationReq* and *CertificateUpdateReq* messages are modified to allow the inclusion of the public contract key CC_{pk} in order to enable the eMSP to generate a contract certificate CC_{cert} for the TPM generated contract key pair. Additionally, the eMSP encrypts CC_{cert} for the EV's TPM using the credential protection mechanism (cf. [195], Section 24; as previously described in Section 2.5.3), such that decryption is only possible if the contract key pair was generated in the same TPM as the provisioning key pair and with the expected attributes.[26] Furthermore, usage authorization for both PnC key pairs is optionally conditioned to a `TPM2_PolicyAuthorize` authorization policy for the OEM's public key, allowing the OEM to provide further policies at a later point in time (e.g., a `TPM2_PolicyPCR` policy such that the keys can only be used if the EV booted into a trusted software state; cf. Section 2.5.2).

In order to implement the credential protection mechanism for contract certificates, a new key pair is introduced to serve as the storage key for credential protection, namely a storage root key pair SRK (the key that is associated with the credential is CC_{pk}). Note that the provisioning key pair cannot be used as the storage key for credential protection since PC_{sk} is used to sign *CertificateInstallationReq* and *CertificateUpdateReq* messages and a storage key cannot be used for signature creation. For the encryption of contract certificates with the credential protection mechanism, the eMSP requires the SRK_{pk} (for the protection of the symmetric encryption key as described in Section 2.5.3) and the authorization policy of the contract key pair (for the calculation of the key's name as described in Section 2.5.3). In order to provide these values to the eMSP in an authentic manner without requiring changes to the ISO 15118 messages, they are encoded in the EV's OEM in the provisioning certificate PC_{cert} via a certificate extension. The following paragraphs describe the respective TPM key templates and the certificate extension.

[25] For simplicity, EV is used to refer to the EVCC. The terms EV and EVCC are used interchangeably since, in the context of this thesis, the EVCC is the only relevant ECU of the of the EV.

[26] More precisely, the eMSP encrypts CC_{cert} with a symmetric key and encrypts the symmetric key using the credential protection mechanism. For simplicity, this process is described as encrypting CC_{cert} with the credential protection mechanism.

Table 2.5.: TPM 2.0 Key Templates [66]

Attribute	SRK	PC and CC
type:	TPM2_ALG_ECC	TPM2_ALG_ECC
nameAlg:	TPM2_ALG_SHA1	TPM2_ALG_SHA256
objectAttributes:	fixedTPM, fixedParent, restricted, decrypt, sensitiveDataOrigin, userWithAuth, noDA	fixedTPM, fixedParent, sign, decrypt, sensitiveDataOrigin
authPolicy:	n/a	Pol_{Auth} (TPM2_PolicyAuthorize)
curveID:	TPM2_ECC_NIST_P256	TPM2_ECC_NIST_P256

TPM Key Templates: The TPM key templates as defined in [66] are shown in Table 2.5. All keys are ECC keys on the NIST_P256, i.e., the same curve as used by ISO 15118. The SRK's template follows the TCG's recommendation for storage root keys [192], except for the use of SHA1 as "nameAlg" (the algorithm used to calculate a key's name; also used in some commands).[27] The PC's and CC's template defines them as general purpose (sign and decrypt) keys, which is required for their use in ISO 15118. Additionally, they have the fixedTPM, fixedParent attributes set (making it impossible to export the keys out of the TPM or move them within its hierarchy) and the sensitiveDataOrigin attributes set (indicating that their private values were generated by the TPM). Furthermore, PC and CC are provided with the `TPM2_PolicyAuthorize` authorization policy for the OEM's public key.

Certificate Extension: The OEM provisioning certificate extension as defined in [66] is shown in Figure 2.20. The values of the public storage root SRK_{pk} and the CC's authPolicy Pol_{Auth} are included in PC_{cert} via a custom, non-critical X.509 certificate extension. Both values are encoded as octet strings. The extension is marked as non-critical (^{nc}) in order to allow for backwards compatibility with regard to eMSPs that do not support it (a non-critical extension can be ignored by any actor that does not recognize it; cf. [35], Section 4.2). If an eMSP supports the extension, they can use the values for the encryption of contract certificates with the credential protection mechanism. Additionally, since old contract credentials can be used by an EV to request new ones (using the *CertificateUpdateReq* message), the same extension is also included in contact certificates. The addition of this extension is the only change to the provisioning-/contract certificate profiles from ISO 15118-2 (cf. [97], Annex B).

The integration of [66] into ISO 15118 starts with the EV initialization. During production, the OEM instructs an EV's TPM to generate a storage root key pair (SRK_{pk}, SRK_{sk}). Additionally, the EV's TPM is instructed to generate the OEM provisioning credential key pair (PC_{pk}, PC_{sk}) underneath the storage root key, whereby provisioning credential key pair's authorization policy is set to the Pol_{Auth} (`TPM2_PolicyAuthorize`) policy for the OEM's public key. Afterwards, the OEM reads out the corresponding public keys SRK_{pk}, PC_{pk} and creates the OEM provisioning certificate PC_{cert} for PC_{pk} including SRK_{pk} and Pol_{Auth} via the certificate extension (cf. Fig. 2.19).

For the EV's first charging process, it generates a new contract key pair (CC_{pk}, CC_{sk}) in its TPM. Afterwards, the EV sends a *CertificateInstallationReq* message, including the new CC_{pk} via an additional field in the message, to the CS. The *CertificateInstallationReq* message is signed with the PC_{sk}, which requires a correct assertion of the key's policy Pol_{Auth}. The CS forwards the message (over the CSO) to the CPS, who verifies the

[27] The use of SHA1 as the SRK's nameAlg is required by [66] for compatibility with ISO 15118's size constraints. In [66], the SRK's nameAlg is only used by the TPM for Key Derivation Function (KDF) and HMAC calculations, i.e., in cases where SHA1 is still considered secure [12].

OEM Provisioning Certificate (PC_{cert})		
Version:		X.509v3 (0x2)
Serial Number:		12345 (0x3039)
Signature Algorithm:		ecdsa-with-SHA256
Issuer:		CN=OEMSubCA2, O=ISO
Validity	Not Before:	May 7 08:40:32 2020 GMT
	Not After:	May 6 08:40:32 2050 GMT
Subject:		DC="OEM", CN=WDKA2B3C4D5E6F7G84, O=ISO
Subject Public Key Info	Public Key:	OCTET STRING (PC_{pk})
	Algorithm:	id-ecPublicKey
	Parameters:	namedCurve secp256r1
X509v3 Extensions	Basic Constraints:[c]	CA:FALSE
	Key Usage:[c]	Digital Signature, Key Agreement
	Subject Key Identifier:[nc]	64 bit ID of PC_{pk}
	Authority Key Identifier:[nc]	64 bit ID of $OEMSubCA2_{pk}$
	CRLDistributionPoints:[nc]	URI:http://example.com/example.crl
	Authority Information Access (OCSP):[nc]	URI:http://ocsp.example.com/
	TPM 2.0 Extension:[nc]	**TPM 2.0 ECC Public Storage Root Key (SRK_{pk}) 512 bit OCTET STRING**
		TPM 2.0 SHA256 Policy Digest (Pol_{Auth}) 256 bit OCTET STRING
Signature	Algorithm:	ecdsa-with-SHA256
	Value:	OCTET STRING (Signature with $OEMSubCA2_{sk}$)

Figure 2.19.: Provisioning Certificate with Custom Extension [66]

signature and forwards the request to the eMSP. If the eMSP supports [66], they generate a new CC_{cert} for CC_{pk} and encrypt CC_{cert} using SRK_{pk} from the PC_{cert} extension. This encryption uses the TPM's credential protection mechanism with CC_{pk} as associated key, i.e., the name of CC_{pk} (which includes Pol_{Auth}) is included in the encryption process. If the eMSP does not support [66], they generate new contract credentials as define in ISO 15118.

The resulting contract credentials are sent to the CPS, who signs them and forwards them to the CS (via the CSO). The CS sends the credentials to the EV after which the EV verifies the CPS' signature and decrypts CC_{cert} with SRK_{sk} in its TPM using the `TPM2_ActivateCredential` command. Note that decryption is only possible if the contract key pair was generated in the same TPM as the SRK, i.e., the same TPM as the provisioning key pair, and with the expected attributes (including the OEM defined policy Pol_{Auth}). Afterwards, the EV can use its contract credentials for PnC-based authentication in ISO 15118, whereby key usage requires a correct assertion of the Pol_{Auth} policy.

In [67], the authors propose an extension to [66]. In particular, [67] adds considerations for the draft version of ISO 15118-20 [98], i.e., the successor to ISO 15118-2, for existing CA processes, and for out-of-band contract certificate installations based on the DKE application guideline VDE-AR-E 2802-100-1 [202]. The relevant considerations and changes to [66] are summarized in the following paragraphs.

ISO 15118-20: The draft version of ISO 15118-20 includes new features, like support for bi-directional power transfer and wireless charging, as well as changes to existing features. For example, TLS is always mandatory (in ISO 15118-2 TLS is optional if EIM-based authorization is used), the *CertificateUpdateReq/Res* messages are removed (old contract credentials cannot be used to request new ones anymore), and the installation of multiple contract certificates is supported (the EV sends multiple *CertificateInstallationReq* messages in a loop and the responses indicate how many more certificates are available).

<table>
<tr><td colspan="4">OEM Provisioning Certificate (PC_{cert})</td></tr>
<tr><td colspan="2">Version:</td><td colspan="2">X.509v3 (0x2)</td></tr>
<tr><td colspan="2">Serial Number:</td><td colspan="2">12345 (0x3039)</td></tr>
<tr><td colspan="4">(no changes to certificate profile form ISO 15118-20; cf. [98], Annex B)</td></tr>
<tr><td rowspan="4">X509v3 Extensions</td><td rowspan="4">Subject Information Access:^{nc}</td><td colspan="2">OID:1.0.20.4</td></tr>
<tr><td colspan="2">TPM 2.0 ECC Public Storage Root Key (SRK_{pk})
512 bit OCTET STRING in Base64</td></tr>
<tr><td colspan="2">OID:1.0.20.5</td></tr>
<tr><td colspan="2">TPM 2.0 SHA256 Policy Digest (Pol_{Auth})
256 bit OCTET STRING in Base64</td></tr>
<tr><td>Signature</td><td>Algorithm:</td><td colspan="2">ecdsa-with-SHA256</td></tr>
<tr><td></td><td>Value:</td><td colspan="2">OCTET STRING (Signature with $OEMSubCA2_{sk}$)</td></tr>
</table>

Figure 2.20.: Provisioning Certificate with Custom SIA Extension [67]

One of the changes in [67] over [66] with regard to ISO 15118-20 is that contract certificates do not include the custom extension anymore and the decrypt attribute for the CC's TPM key template is not set. Additionally, a non-critical Subject Information Access (SIA) certificate extension [35] is used to include SRK_{pk} and Pol_{Auth} in PC_{cert} instead of the custom extension (cf. Fig. 2.20). The SIA extension consists of a list of `AccessDescription` elements. Each element consists of an Object Identifier (OID) and a data field, whereby the OID indicates how the data is supposed to be interpreted (OID 1.0.20.4 for the SRK_{pk} and 1.0.20.5 for the Pol_{Auth}, both encoded as Base64 strings). A SIA extension is chosen in [67] as it is already used by ISO 15118-20 to encode use case-specific information in contract certificates (cf. [98], Annex B).

Existing CA Processes: CAs commonly require a CSR [143] for a public key in order to create a corresponding certificate (e.g., [86]). In order to address this, [67] includes a PKCS #10 CSR [143] for CC_{pk} in *CertificateInstallationReq* messages (instead of only the CC_{pk}).

VDE-AR-E 2802-100-1: The application guideline VDE-AR-E 2802-100-1 [202] describes an out-of-band mechanism for the provisioning of contract credentials. The guideline describes two certificate pools, namely the OEM Provisioning Certificate Pool and the Contract Certificate Pool. An OEM stores the provisioning certificates of its EVs in the OEM Provisioning Certificate Pool from where eMSPs can download them (after an EV user registers their PCID but without waiting for an *CertificateInstallationReq* message from an EV). An eMSP can then generate contract credentials for the user's EV, encrypt CC_{sk} with the PC_{pk} from the downloaded certificate, and send the new contract credentials to the CPS where they are signed and stored in the Contract Certificate Pool. When the EV now sends a *CertificateInstallationReq* message, the pre-generated credentials can simply be retrieved from the Contract Certificate Pool.

In order to support this out-of-band mechanism, [67] additionally proposes that the EV's TPM generates contract key pairs and corresponding CSRs during production. The CSRs are uploaded by the OEM, along with the EV's provisioning certificate (including the SIA extension), into the OEM Provisioning Certificate Pool. An eMSP downloads the provisioning certificate along with one CSR, which allows them to generate and encrypt (using the credential protection mechanism) a CC_{cert}.[28] The encrypted CC_{cert} is signed by the CPS and stored in the Contract Certificate Pool from where it can be retrieved.

[28] More precisely, only the signature of CC_{cert} is encrypted in order to allow the EV to identify for which of its contract keys CC_{cert} was generated.

3. Related Work

This section provides an overview of related work regarding privacy-preserving methods for charge authorization. As local EIM-based charge authorization is usually based on RFID technology, Section 3.1 provides an overview of common problems and solutions in the field of privacy-preserving RFID protocols. Afterwards, Section 3.2 presents an overview of existing privacy-preserving EV charging architectures with a focus on charge authorization and billing.

3.1. Privacy-Preserving RFID Protocols

RFID devices (often referred to as RFID tags) are small microchips, designed for wireless data transfer [100]. Passive RFID tags, i.e., tags without an on-board power source that derive their transmission power from an interrogating RFID reader, are commonly used as an inexpensive method for the automatic identification of objects and people [100]. RFID technology is used in a wide variety of applications, including keyless entry for vehicles, animal tracking, payment systems, and supply chain management [73]. The use of RFID, however, comes with privacy concerns, mainly as it allows the possibility of clandestine tracking since most RFID tags emit a Unique Identifier (UID) that can be used to track the individual carrying the tag [100].

One of the simplest solution to address RFID privacy concerns in the retail supply chain management setting is to "kill" RFID tags at the point-of-sale [73, 100], i.e., the tag turns permanently unusable after receiving a kill command from a reader (authenticated via a tag-specific password) in order to prevent the tracking of the respective consumer. Tag killing, however, has the disadvantage that it prevents any post-purchase benefits of RFID tags for consumers and that it is not usable in most other settings like keyless entry systems or payment systems [73, 100]. A different approach is the use of "blocker tags," i.e., a special kind of RFID tag that can selectively simulate other tags in order to prevent an unauthorized reader from singling out specific tags [102]. In order to remove the need for special tags, a different solution is "soft blocking" whereby the RFID reader implements a specific scanning behavior based on a set privacy policy (e.g., the reader does not return data for tags that are identified as private) [101]. As soft blocking is vulnerable to rogue readers that do not comply with the privacy policy, it would require appropriate auditing and legislative measures [101].

The previously mentioned RFID privacy concepts mainly consider basic tags [100] (tags that cannot execute cryptographic functions). However, in order to achieve the higher level of security that could be required for certain use cases (e.g., authentication for a payment systems), the use of more sophisticated tags that are able to perform cryptographic functions (with the disadvantage of being more expensive than basic tags) might be required [157]. While asymmetric cryptographic functions are often argued to be too expensive (in terms of cost and chip area) for the use in RFID tags, symmetric key-based functions (using hashes or symmetric encryption) provide reasonable trade-off between efficiency and security [103, 9].

The use of a symmetric key-based authentication generally requires the use of a shared secret key between a tag and a reader (or a backend system). Since (usually) multiple tags should be able to authenticate themselves

at a reader, the reader must be able to identify which key to use for the presented tag. Although this problem could be solved by the tag sending its ID, this solution is problematic when considering privacy as any reader can learn the identity of the tag [100]. The author of [100] identifies the straightforward solution to this privacy problem to be a "key search" procedure on the reader, i.e., the tag generates an authentication value based on some input using its secret key and the reader tries to verify the authentication value with every available secret key. The author further points out that, in order to ensure the privacy of the tag owner, the tag's authentication value must be different for every session, i.e., either the input or the secret key (or both) must change for every session.

A privacy-preserving, symmetric key-based RFID authentication, following the key search procedure, can be implemented in different ways. One of the first approaches using the key search procedure is presented in [204]. The authors of [204], inter alia, propose a method whereby the tag generates a random nonce n, builds a hash over its key k_t appended with the nonce n (d = Hash($k_t||n$)), and sends the resulting digest d and the nonce n to the reader. Afterwards, the reader can iterate over all known keys k_t in order to find the one that results in the same digest, i.e., by building d' = Hash($k_x||n$) until $d' = d$. A problem with the approach from [204], however, is that its computational cost is linear in the amount of tags (or the amount of keys on the reader) [100]. A method to address this problem is presented in [132]. The authors of [132], inter alia, propose a tree-based key management method that reduces the computational cost on the reader to $O(log\ m)$ (for m tags). Each tag is associated with the leaf of a balanced binary tree and each edge in the tree is associated with a secret key. The reader knows all secret keys and each tag stores the secret keys that correspond to the path from the tree's root to the tag's leaf. At run-time, a tag authenticates using each of its secret keys, starting from the root and going to the leaf, such that for each of the authentications, there are only two possible keys for a reader to try. While the scheme from [132] reduces the computational overhead on the reader to $O(log\ m)$, it also requires $O(log\ m)$ interactions between a tag and a reader for each run and it requires $O(log\ m)$ storage on the tag. Furthermore, as there is a overlap in keys between tags, the compromise of a single tag can reduce the privacy of other tags as discussed in [8].

A possibility to avoid the key search overhead is to maintain a synchronized state between tag and reader [100]. An example for implementing such a synchronized state is identified in [100] to be a counter-based approach: Each tag could maintain a counter (that is incremented each time the tag interacts with a reader) and use this counter as input together with its secret key when generating its authentication value. Assuming the reader knows the counter value of each tag, the reader could maintain a searchable database for the next output values of all tags. A different approach at changing the tag's state, instead of maintaining a counter, is by updating its key. A commonly employed procedure for key updates is the hash-chain method [188]. For the hash-chain method, a tag renews its secret key by hashing it (k_t^{i+1} = Hash(k_t^i)) after each authentication such that the tag's current key cannot be used to infer old keys [188]. The main advantage of the key update approach is that it can provide forward security/privacy, i.e., if an adversary is able to extract a tag's current secret key, they cannot use this key to break the privacy of previous authentications as they were calculated using a different key [152].

While the synchronization approach is effective at addressing the overhead of the key search procedure, it also introduces the problem of desynchronization [100, 188]. That is, if a tag and reader lose their synchronization, e.g., because the tag is queried by a malicious reader, authentication is no longer possible. Several methods have been proposed in order to address the desynchronization issue. A straightforward solution is to maintain the next l authentication values for each tag in the searchable database (instead of only the one next value), such that if a desynchronization of less than l values occurs, re-synchronization is still possible [159]. While this solution can address desynchronizations to some extent, it is not very effective against a dedicated adversary trying to conduct a denial of service attack [123]. A different approach at addressing the desynchronization problem is presented in [45]. The authors of [45] describe a protocol for mutual authentication between

a tag and a reader and propose that a tag only updates its state after a successful mutual authentication. Assuming that only trustworthy readers can authenticate themselves to a tag, this method prevents the active desynchronization by an adversary. However, this method has the drawback that a tag always emits the same value between two sessions with a successful mutual authentication.

Many RFID protocols assume a constant connection between the reader and a backend server, whereby the reader mainly forwards data between the backend and the tag such that the secret key only needs to be shared between the tag and the (high-security) backend system. Many use cases, however, require readers to be able to operate offline (e.g., a mobile ticketing system for sports events), which often results in readers handling sensitive data [9]. Additionally, for a large-scale deployment of an RFID system it is likely that an adversary is able to compromise a single reader [70]. Considering that readers may be compromised while handling sensitive data highlights the importance of designing protocols that do not lose the security and privacy properties of the entire system based on single reader compromise [70]. The authors of [9] propose a method to reduce the resulting harm of a reader compromise by renewing a tag's authentication key after a reader compromise was detected. The backend system is only needed during the initial setup and for key renewal after a compromise was detected, such that the system can operate mostly offline. Similarly, the authors of [7] propose a solution that accepts a loss of privacy after reader corruption but aims to restore it later on. In [7], privacy restoration is achieved by informing tags about compromised readers after a compromise has been detected.

3.2. Privacy-Preserving EV Charging

In the vehicular context, privacy is a pervasive problem due to the ever-increasing integration of information technology into vehicles [83]. Especially, when considering the integration of communication capabilities into vehicles, the ease for a pervasive, automatic tracking of vehicles as well as the resulting threats to an individual's location privacy becomes apparent [21]. The EV charging use case, including its V2G and backend communication, is no exception and introduces various privacy problems of its own such as the threat of backend operators tracking EV users based on their charging behavior [119]. Exemplifying the possible scope of these problems, [78] lists existing privacy challenges in EV charging, including concealed charge data aggregation, charge session unlinkability, identity-/location privacy, and privacy-preserving billing. The rest of this section presents an overview of possible approaches at privacy in EV charging.

In [113], a privacy-preserving approach for the selection of a CS based on tariff options and travel distance is presented. The authors aim to address privacy issues in the demand response and dynamic pricing use cases such as allowing a user to find the CS with the lowest available price in a certain area while not revealing the user's location and needed energy. The authors propose a blockchain-based architecture in order to address their goals without requiring the addition of a trusted third party. An EV starts by placing a request in the blockchain containing its ID (which may be changed for every request) along with the required energy over a time interval within a geographic region. Afterwards, CSs within the desired region publish bids for the EV's request identifying their offered price. The EV can gather and evaluate the bids, computes a binding and hiding commitment [42] for the chosen CS (a hash over the EV's ID, the CS' ID, and a random nonce), and places the commitment in the blockchain. Finally, in order to charge, the EV approaches the CS and sends its ID and previously generated nonce such that the CS can verify the EV's hiding commitment. The authors argue that their architecture addresses the privacy issues under consideration of an honest-but-curious adversary model. In particular, it is argued to hide an EV's position (assuming the selected geographic region is broad enough), hide the EV's selected price from anyone but the selected CS (assuming more than one bid

was placed by CSs), and prevent the tracking of EVs over time (assuming EV IDs are changed per request). The privacy implications resulting from charging session billing, however, are not considered in [113].

In [216], a privacy-preserving approach for the matching of EVs with charge suppliers (e.g., CSs but the authors also consider the possibility of vehicle-to-vehicle charging) is presented. An efficient matching is argued to involve the in-advance scheduling of charging sessions, taking into consideration the distribution of EVs and the availability of charge suppliers. While this scheduling could be implemented by a centralized server, the authors argue that such an approach would significantly impact user privacy as the locations of suppliers and EVs as well as further charge parameters (e.g., the required energy) are revealed. Instead, the authors propose an architecture for the distributed matching of EVs and charge suppliers using local communication. The matching algorithm is based on a computation of the nearest neighbors while avoiding the need to reveal a user's location by using homomorphic encryption (encryption that allows computations on the ciphertext, which are reflected in the plaintext after decryption). Only a matched pair has to exchange their actual locations in order to perform the charging session. The authors argue that their architecture preserves the privacy of users under consideration of an honest-but-curious adversary model. Furthermore, simulations are used to show the feasibility of the approach.

In [124], a privacy-preserving approach for the reservation of CSs is presented. Due to the relatively short range and relatively frequent recharging of EV's, a CS reservation mechanism is argued to be an important feature with regard to EV usability. However, a reservation mechanism without privacy considerations is likely to reveal a user's identity and location and thus considered a significant threat to the user's location privacy. In order to address this problem, the authors propose a solution that can ensure the authenticity of a user before allowing a reservation while providing a user with anonymity towards CSs and preventing the linkability of multiple reservations. The privacy properties of the solution can even be guaranteed under consideration of a dishonest reservation service provider who colludes with CSs. Addressing the privacy implications of the billing of charge sessions, however, is beyond the scope of [124]. In particular, it is unclear if the solution could be applied to the case where charge session billing and reservations are handled by the same provider, which, e.g., is the case with eMSPs in current EV charging protocols (cf. Section 2.4.3).

In [111], a method for privacy-preserving charge scheduling/reservation is presented. Due to the potentially excessive load caused by EV charging a method for charge scheduling based on an EV's expected arrival time at a CS and required energy is argued to be an important factor in ensuring a stable electricity grid operation. As this process, however, involves sensitive information such as EV location and duration of stay, privacy considerations are argued to be necessary. While an anonymous system can combat the privacy threats it is argued to be vulnerable to impersonation or insider attacks (e.g., for a denial of service). Thus, the authors propose an anonymous authentication-based solution that supports the revocation of malicious EVs and counteracts denial of service attacks based on throttling. In more detail, an EV is issued an anonymous permit by a trusted third party which can be used to request a charge and the EV requires a receipt from a CS (issued after a charge) in order to receive an new permit. The anonymity of an EV can be revoked based on a complaint from a CS by cooperation between multiple third parties, modeled as honest but curious. The authors argue about the security and privacy of the solution. In particular, the solution prevents a CS from linking an EV's charge sessions and third parties cannot unilaterally revoke an EV's anonymity. The authors implement their solution in order to show its feasibility. Considerations for the privacy-preserving billing of a charging session, however, are not detailed in [111].

In [121], a privacy-preserving solution for the authentication of EVs in a dynamic, contactless charging scenario, supporting the billing of EV users is presented. For dynamic, contactless charging, charging pads are integrated into the road such that an EV can be charged while driving over these pads. For an efficient charging, many small pads have to be placed along several kilometers of a road, forming a charging section. In order to

enable the billing of a user, an EV thus has to be able to rapidly authenticate itself to a large amount of pads in a short time period. In order to preserve an EV user's privacy in this scenario, the authors propose a lightweight authentication protocol based on symmetric keys that ensures location privacy through pseudonyms. The proposed method starts with a key distribution phase in which a charge service provider generates sets of pseudonyms and corresponding symmetric keys and sends them to the operators of charging pads who forward them to their pads. The key distribution phase is repeated every night. Afterwards, the charge service provider assigns keys and pseudonyms to EVs, which can use this data to authenticate themselves to pads. For this, the EV periodically broadcasts a message containing its pseudonym and a symmetrically encrypted charge request. If a pad receives the message it can identify the correct key based on the EV pseudonym, decrypt the charge request, and verify the decrypted data. If the verification is successful, the pad removes the used key from its storage to prevent reuse. EV pseudonyms are only reused within a charging section. Periodically, pads inform their operators about their charging sessions, including the corresponding EV pseudonym, which is used for the billing process in cooperation with the user's charge service provider. The authors argue about the security and privacy of the solution. In particular, as an EV uses a different pseudonym per charging section and charge service providers are not informed about the location of a session, the solution is argued to preserve a user's location privacy. The authors implement their solution in order to show its feasibility. Additionally, in [122], an extension to [121] is presented that uses an encrypt-then-MAC method for authentication and further optimizes the success probability of authentication.

In [81], an architecture for the automatic and privacy-preserving charging and billing of EVs is presented. In particular, the authors design their solutions as modular enchantments of contract-based charging via ISO 15118. The authors propose a minimization of the involved PII, replace ISO 15118's PnC authentication mechanism with Idemix anonymous credentials [27] allowing an EV to anonymously prove that is has a valid charging contract to a CS, and replace ISO 15118's metering receipt mechanism with short group signatures [20] for which a new trusted third party is introduced as group manager. Furthermore, the authors propose the use of privacy-preserving information flows (e.g., based on an anonymous communication network) and the use of privacy-preserving payments between an eMSP and energy provider via a new trusted third party such that both entities are not informed about each other's identities. The authors argue about the privacy of their solution. In particular, it prevents the excessive use of PII, the linkability of an eMSP and EV identifier, and the linkability of a CS and EV identifier.[1] The authors implement their solution as a proof-of-concept showing that, while the resulting overhead is arguably significant (e.g., in total 195 seconds for the generation and verification of an EV's anonymous charging contract proof on Raspberry Pis), the solution is nonetheless feasible. The solution, however, is not fully compatible to existing roles and data flows as it adds new trusted third parties and communication paths. Additionally, support for CS reservations is not considered.

In [134], a privacy-preserving approach for EV charging and billing is presented. The approach focuses on the EV roaming case and aims to protect the EV user's identity and location privacy while supporting a multi-user scenario with fair billing. The solution requires the registration of an EV at a supplier (comparable to the role of eMSP). During the registration, the EV user and EV are identified to the supplier via a secure and authentic out-of-band channel. The user receives a smart card with a symmetric key and an asymmetric key pair (including a certificate over the public key) and the EV receives a set of pseudonym IDs. The multi-user scenario can be supported by providing multiple smart cards for one EV. To start a charging session, the EV sends one of its IDs to the CS and the CS forwards this ID along with the electricity price and a current timestamp to the EV user's smart card. The smart card further appends this data with the supplier's ID, generates a signature over the data with its private key, and encrypts the signature with its symmetric key. The data and the encrypted signature are forwarded to the supplier, who verifies the request and authorizes

[1] Note that in [59], a formal privacy analysis of [81] is conducted and improvements are suggested in order to address found weaknesses (cf. [59] for details).

the charging session. After a charging session, the CS generates a consumption report, which is signed by the EV user's smart card and, with a symmetrically encrypted signature, forwarded to the supplier. The supplier can use this data to bill the user. The authors argue about the security and privacy of the solution, whereby smart cards and CSs are assumed to be tamper-proof, suppliers are modeled as honest but curious, and all entities are assumed to be time synchronized. In particular, only the supplier can identify an EV and its user and only the supplier can link multiple charging sessions of the same EV/user. Furthermore, no entity can link an EV user to a location. Furthermore, a formal security validation using the AVISPA tool [3] is conducted. The solution, however, does not support offline charge authorizations or a PnC authorization mechanism (i.e., an automatic mechanism without user interaction) and CS reservations are not considered. Additionally, no evaluation of the incurred overhead is provided.

In [218], a privacy-preserving approach for EV charging and billing including a CS reservation mechanism is presented. The approach aims to provide EV user anonymity and the unlinkability of charging sessions by using a TPM-based DAA scheme for EV authentication. The EV's TPM is initialized with the help of a trusted third party and receives a long term ID, used to later create temporary IDs. Afterwards, the EV can be registered at a service provider (responsible for charge reservation and authorization), whereby the user pre-pays for future charging sessions, and the EV receives a randomizable credential (reflecting the pre-paid amount) from the provider. For a charging session, the EV starts by reserving a CS for which it sends a temporary ID (generated by the TPM based on the long term ID) to the CS along with the expected time of arrival and charge duration. When the EV arrives at the CS, it sends the temporary ID along with a randomized form of its credential. The CS validates the reservation based on the ID and forwards the randomized credential to the service provider for verification. Afterwards, the credential is updated to reflect the corresponding deduction in its pre-paid credit. The authors further consider a possibility to punish EV users when they do not use their reservation (based on cooperation between the CS and the trusted third party) and a possibility to recover stolen EVs (by registering the EV as stolen at all CSs based on cooperation between the EV user and the trusted third party). The authors argue about the security of their solution and that it provides unlinkability (with regard to CSs and service providers). The solution, however, does not support offline charge authorizations or contract-based charging and does not describe how the functions can be implemented with the TPM's interface (it appears to require changes to the TPM specification, e.g., for the generation of temporary IDs). Additionally, no evaluation of the incurred overhead is provided.

In [217], a method for the PnC-based privacy-preserving charging and billing of EVs is presented. The authors consider the contract-based EV roaming use case with a focus on the definitions of the ISO 15118 protocol. The authors propose an extension to the PnC mechanism using a TPM-based DAA scheme in order to provide anonymous EV charging with secure billing. EVs are required to be provided with DAA credentials from an eMSP by running a join protocol via a secure out-of-band channel. The EV's DAA key pair is proposed to be sealed to a verification of the integrity of its software state. In order to authorize a charging session, an EV uses its private DAA key via its TPM to sign a challenge from the CS using the CS's timestamp as a basename. During a charging session, the EV signs meter receipts again using its private DAA key and using the initial timestamp as a basename such that all signatures during a session are linkable. For the billing of the charging session, the EV encrypts its eMAID for the eMSP and the CS encrypts the meter values for the eMSP. The authors argue about the security and privacy properties of their solution, including the unlinkability of charging sessions, non-repudiation of session data, and secure key storage/usage. The solution, however, does not provide detailed consideration for EV charging backend protocols. Additionally, support for important features of current EV charging protocols such as the offline operation of CSs, CS reservations, and the automatic installation of an EV's contract credentials are not considered. Furthermore, no implementation-based evaluation of the resulting overhead is provided.

Arguably, none of the surveyed related work can address the full range of features and processes related to EV

charge authorization and billing. This work aims to close this gap by providing a detailed analysis of current EV charging protocols with regard to functionality, security, and privacy and presenting a feasible solution for addressing the identified aspects as elaborated in the following sections.

4. System Model

This section details the contract-based EV charging system model. We use OCPI as example roaming protocol since (based on the comparison of roaming protocols in Section 2.4.3) it covers the widest range of use cases and since [106] ranks it the highest in terms of transparency, openness, and impartiality of governance. Section 4.1 describes the existing/assumed security-related data and measures. Section 4.2 describes the EV charging processes and provides a detailed analysis/identification of the involved personal data.

4.1. Security Measures in the EV Charging Infrastructure

Fig. 4.1 provides an overview of the security-related data and measures that are involved in contract-based EV charging based on Section 2.4. The following sections detail the security-related data and measures per EV charging entity.

4.1.1. EV Security Measures

The EV or, more precisely, the EVCC[1] is equipped with the provisioning key pair and corresponding certificate. The provisioning certificate includes the PCID, which uniquely identifies the EV and is used by the EV user to register the EV at an eMSP after the conclusion of an e-mobility contract. The process for contract conclusion and PCID registration is out of scope. The provisioning credentials (key pair and certificate) are assumed to be provided by the EV's OEM in a secure environment (i.e., before deployment). The process for providing provisioning credentials is out of scope.

The EV uses its provisioning credentials to request new contract credentials by signing a *CertificateInstallationReq* message (including the provisioning certificate). We only consider the *CertificateInstallationReq* message since the *CertificateUpdateRes* message is going to be removed in future versions of the ISO 15118 standard (cf. [98]; as mentioned in Section 2.5.5). The *CertificateInstallationReq* message is forwarded via the CS to the backend. The corresponding *CertificateInstallationRes* message contains the EV's contract credentials, issued by the eMSP with an encrypted private contract key, which the EV can decrypt using its private provisioning key. The contract certificate contains the eMAID, which is used as ID Token as part of the PnC authorization mechanism. An ID Token is a unique identifier of an EV user or, more precisely, a unique identifier of the e-mobility contract between user and eMSP. The ID Token serves as a role pseudonym (cf. Section 2.1.2) of the user, i.e., it identifies them as a customer of the eMSP and can be linked across different communication relationships (with different CSs).

The EV uses its contract credentials during an ISO 15118 session to authenticate itself to the CS as part of the PnC authorization mechanism by signing a nonce from the CS with the private contract key. Additionally,

[1] The terms EV and EVCC are used interchangeably.

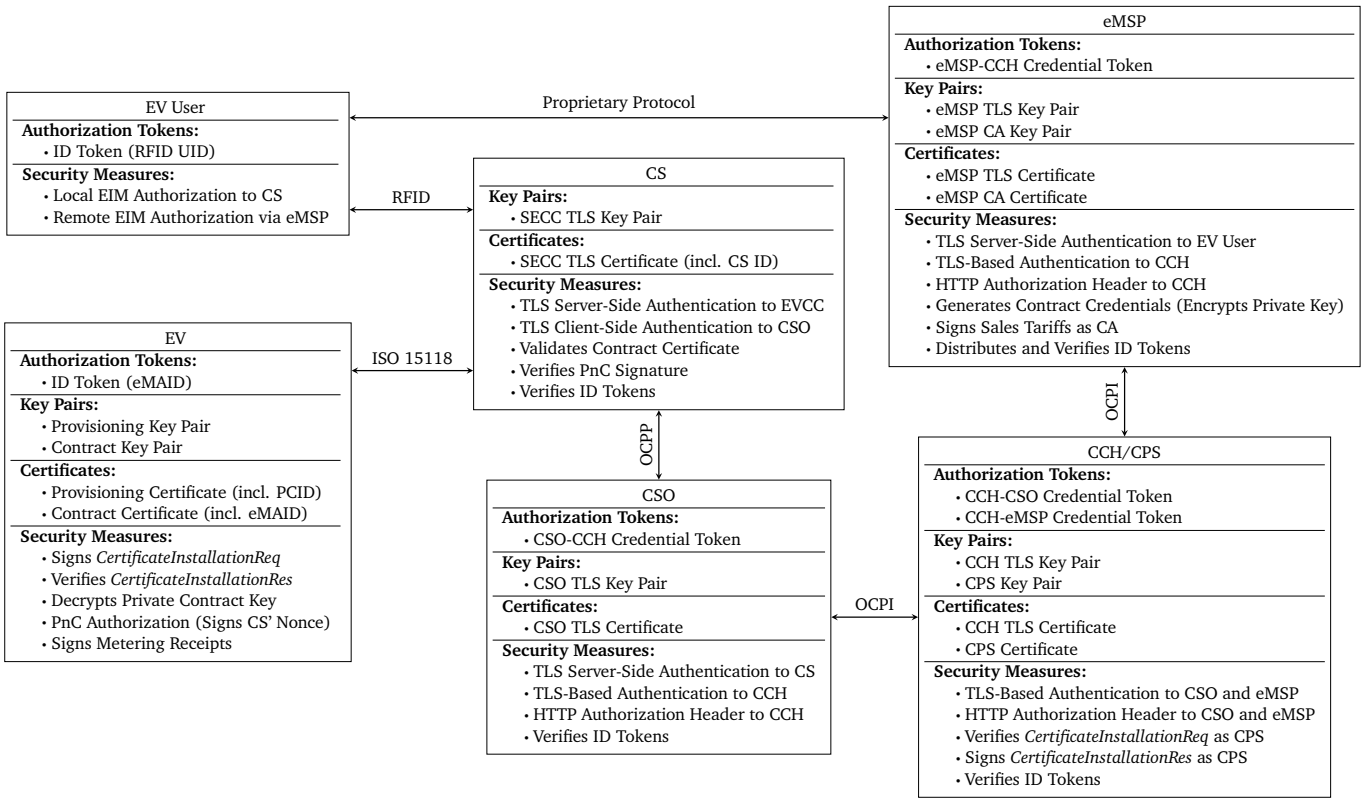


Figure 4.1.: Security-Related Data and Measures of EV Charging Entities

the EV optionally signs metering receipts during an ISO 15118 session with its private contract key if the CS requests the EV to do so.

4.1.2. EV User Security Measures

The EV user is provided with the option of EIM-based charge authorization by their eMSP after the conclusion of an e-mobility contract. For the local authorization at an CS, the user is provided with an RFID card. When held to an CS, the card's UID is read out and used as ID Token for the EIM authorization mechanism.

For the remote authorization via the eMSP, the EV user can use a smartphone app to communicate with their eMSP using a proprietary protocol. The definition of a specific protocol is out of scope. The proprietary protocol is assumed to authenticate the eMSP via TLS server-side authentication and the EV user via a secure client-side authentication mechanism. After mutual authentication, the eMSP imitates the remote authorization for the user.

4.1.3. CS Security Measures

The CS or, more precisely, the SECC[2] is equipped with TLS key pair and corresponding certificate. The SECC's TLS credentials (key pair and certificate) are assumed to be provided by the SECC's CSO in a secure

[2]The terms CS and SECC are used interchangeably.

environment (i.e., before deployment). The process for providing the SECC's TLS credentials is out of scope.

The SECC's TLS certificate contains the CS ID, which is required by the SECC's certificate profile as defined in ISO 15118 (cf. [97], Table F.2). The CS uses the SECC's TLS credentials to authenticate itself to an EV for an ISO 15118 session using the server-side authentication mechanism during the TLS handshake. While in ISO 15118-2, TLS is only mandatory for the use of PnC and/or VAS, we assume it is always used, i.e., even in the case of only EIM (cf. [97], Section 7.3.4). This assumption reflects the stricter security requirements of ISO 15118-20 where TLS is always mandatory (cf. [98], Section 7.3).

While OCPP allows the use of security profiles where the CS is authenticated using the HTTP Basic method, we assume that TLS client-side authentication is used as per OCPP security profile SP_3. Note that OCPP allows the use of the SECC's ISO 15118 TLS credentials for the connection to the CSO (cf. [148], Requirement A00.FR.428). Since we already assume that the CS always authenticates itself to the EV via TLS, the additional assumption of TLS client-side authentication towards the CSO is not further limiting.

For PnC authorization, the CS verifies the EV's contract certificate, signature, and ID Token (eMAID) locally. Alternatively, the contract certificate and ID Token can be forwarded to the CSO for verification. For EIM authorization, the CS either verifies the ID Token (RFID UID) locally or forwards it to the CSO for verification.

4.1.4. Security Measures of the Backend Actors

All backend actors are assumed to possess TLS credentials and to use them for TLS server-/client-side authentication.[3] Additionally, OCPI uses credential tokens (i.e., random nonces) sent via an HTTP Authorization header in every request (cf. [142], Section 7). OCPI requires the backend actors to exchange initial credential tokens via an out-of-band mechanism. This initial exchange is out of scope for the thesis. Initial credential tokens can be used to request new ones via an OCPI request message. The following paragraphs describe the backend actor-specific security measures.

CSO Security Measures

The CSO possesses TLS credentials for the TLS-based authentication towards its CSs and towards the CCH. Additionally, the CSO possesses the CSO-CCH credential token used in the HTTP Authorization header in requests to the CCH. Optionally, the CSO verifies contract certificates and ID Tokens for their CSs. If the CSO cannot verify an ID Token locally, it is forwarded to the CCH.

CCH/CPS Security Measures

The CCH possesses TLS credentials for the TLS-based authentication towards the CSOs and the eMSP. Additionally, the CCH possesses the CCH-CSO and CCH-eMSP credential tokens used in the HTTP Authorization header in requests to the CSO and eMSP respectively. Optionally, the CCH verifies ID Tokens for connected CSOs. If the CCH cannot verify an ID Token locally, it is forwarded to the eMSP. Furthermore, the CCH is assumed to also implement the role of CPS and possesses the corresponding credentials. In its role as CPS, it verifies the *CertificateInstallationReq* messages from EVs and signs the resulting responses from eMSPs.

[3] The OCPP security profile SP_3 requires TLS server-side authentication from the CSO. While OCPI only requires TLS server-side authentication, mutual authentication is possible (cf. [142], Section 4.1). OCPI does not stipulate which actor takes the role of server/client.

eMSP Security Measures

The eMSP possesses TLS credentials for the TLS-based authentication towards its EV users and towards the CCH. Additionally, the eMSP possesses the eMSP-CCH credential token used in the HTTP Authorization header in requests to the CCH. The eMSP generates contract credentials for its EV users using its eMSP CA credentials. The corresponding private contract keys are encrypted with the respective user's public provisioning key from their *CertificateInstallationReq* message. Furthermore, the eMSP may sign sales tariffs for its EV users using the same CA credentials.[4] Optionally, the eMSP verifies ID Tokens for the CCHs. The eMSP can distribute lists of authorized ID Tokens in order to enable the local verification of ID Tokens at the CCH, CSO, and CS.

4.2. Processes in the EV Charging Infrastructure

Given the functionalities of EV charging protocols as described in Section 2.4, the core processes of EV charging are: (i) initial connection setup, (ii) charge authorization, (iii) charge session including scheduling and control loop, and (iv) end of charge including billing. Furthermore, the ISO 15118 PnC functionality also requires the additional process of contract credential installation in between processes (i) and (ii) if no valid contract credentials are available on the EV.

The following sections describe these processes in detail with a focus on protocol interactions and the transmitted data. Additionally, any personal data that is exchanged in these processes is identified. Based on the GDPR definition as discussed in Section 2.2.1 and the PII/re-identification overview presented in Section 2.1.3, we consider the following types of information as personal data:

- pseudonyms of the EV user (cf., e.g., [129, 199]);
- identifiers of the EV (cf. [129]);
- pseudonyms of a small, well-defined group of users (cf. [129]);
- dates that directly relate to the individual user (cf. [199]);
- information that can be used to build mobility traces of a user (due to the results of [39]);
- information about the financial transactions of a user (due to the results of [40]);
- information that reveals a charge session's location and time (due to the results of [184]); and
- charging parameters that are specific to an individual EV or charging session (e.g., the vehicle's state-of-charge, required amount of energy, or the user's intended departure time; due to the results of [185]).

[4] Sales tariffs are optional (cf. [97], Section 8.4.3.8) and the signature is only required for PnC (cf. [97], Requirement V2G2-307). If the eMSP provides signed sales tariffs in the PnC case, the signature must be created using the same credentials that were used to issue the PnC credentials (cf. [97], Requirement V2G2-906).

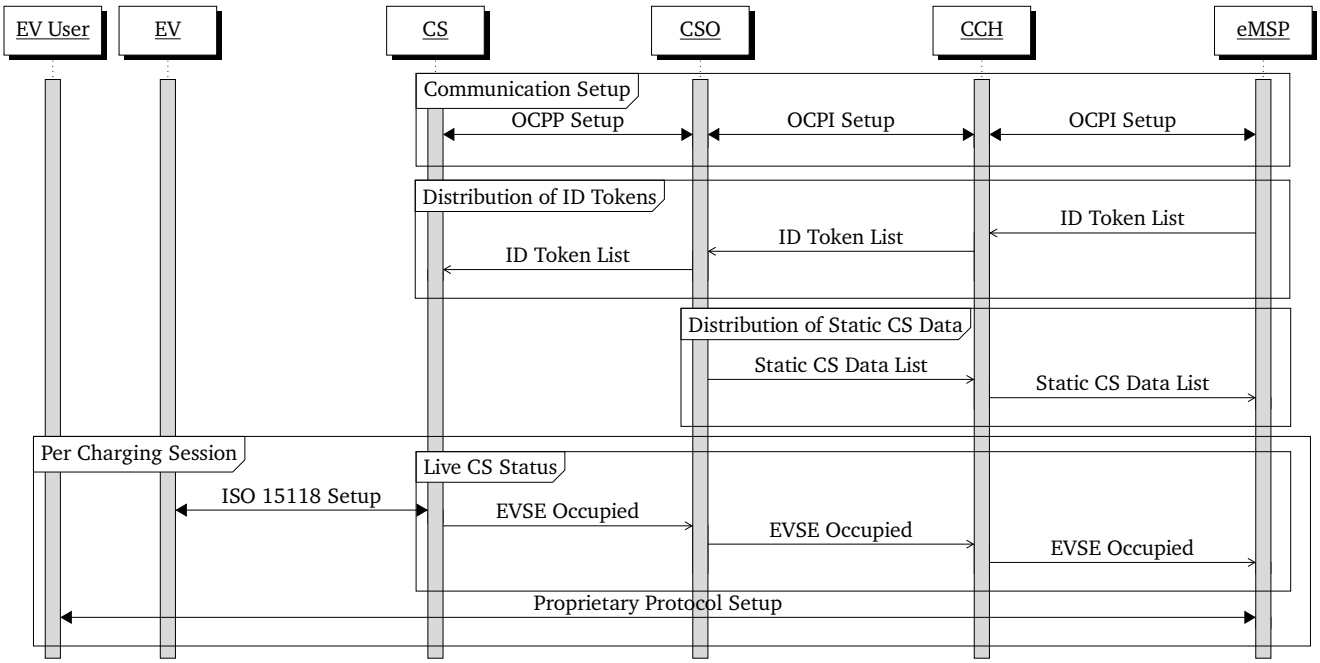


Figure 4.2.: Initial Communication Setup and Data Provisioning

4.2.1. Initial Communication Setup and Data Provisioning

Before a user can start charging, all involved entities need to be set up. This process includes the setup of the OCPI and OCPP communication sessions independently of a specific charging session as well as the setup of the ISO 15118 communication sessions per charging session. Additionally, the setup process includes the provisioning of static CS and ID Token (i.e., user authorization) data. Moreover, after an EV is connected to a CS, the CSO has to inform the associated eMSPs about the occupation status of this CS. Furthermore, a connection between an EV user and their eMSP might be setup (e.g., via a smartphone app) for remote authorization of a charging session. The process for initial communication setup and data provisioning is shown in Fig. 4.2 and detailed in the following paragraphs. Note that the “EV User” entity in sequence diagrams represents both their smartphone as well as their RFID card.

The data that is exchanged during this process is shown in Table 4.1. Data is grouped by messages per process and the letters “O” and “R” are used ot mark the originator and recipient respectively. Originator and recipient can be the “EV (User)” (in tables used to represent the user’s EV or other devices such as smartphone or RIFD card), the CS, the CSO, the CCH, and/or the eMSP. Personal data is marked in bolt fond and the text describing the corresponding processes provides the reasoning for why data is classified as personal.

OCPI Setup: OCPI communication is set up between eMSPs and the CCH as well as between CSOs and the CCH. As mentioned in Section 4.1.4, the OCPI connections use TLS with multilateral authentication. OCPI defines its own application layer protocol setup, starting with the exchange of available protocol versions/interfaces. Afterwards, new credential tokens (i.e., random nonces) are exchanged between the actors. These credential tokens are later used for request authorization via an HTTP Authorization header (cf. Section 4.1.4).

Personal Data: None of the OCPI setup steps involve personal data of the EV user.

OCPI Data Provisioning: After the communication is set up, the eMSP sends a list of their users' ID Tokens together with the respective authorization status to the CCH. The CCH distributes this list to all associated CSOs. Similarly, the CSO sends their static CS data (location, existing EVSEs, etc.) to the CCH from where it is distributed to associated eMSPs.

The list of ID Tokens sent via OCPI from the eMSP over the CCH to CSOs contains, for each token, the eMSP's ID, the token's ID, the user's eMAID, the token's validity, as well as the time when the token was created/last updated and optionally includes the token's group ID, the user's preferred language, the user's charging settings (fast, cheap, green, or regular), and an ID of the user's contract with their energy supplier.[5] The list of ID Token as sent over OCPP from a CSO to their CSs contains a subset of the OCPI data and optionally a charging priority, and EVSE IDs (if the token can only be used at specific EVSEs).

Personal Data: An ID Token uniquely identifies the contract between the eMSP and their customer. This customer is in most cases an identifiable natural person (i.e., the EV user). Hence, we assume that an ID Token also uniquely identifies the EV user in most cases (i.e., it is a pseudonym for this user) and with that clearly constitutes personal data. In particular, the Token UID, eMAID, and energy contract ID are pseudonyms of an EV user. Each one of these pseudonyms can be used on its own to uniquely identify an individual EV user. The group ID is a pseudonym of a small, well-defined group of individuals, i.e., EV user that are authorized to stop each others charging sessions, and thus also constitutes personal data. The token update time is personal data if it reflects a date that is directly related to the user, e.g., if it correspond to the start of the user's e-mobility contract. The user's language, their charging setting, the eMSP ID (i.e., with whom the user has a contract), the charging priority, and the validity status of the token are considered to be too general (i.e., not high-dimensional or sparse) to constitute personal data. While the EVSE ID can generally be assumed to uniquely identify a location, the ID Token list is sent independently of the actions of EV users. Thus, in the ID Token list provisioning process, the EVSE ID does not relate to the EV user's location and is not considered personal data.

The static CS data that is sent via OCPI from the CSO over the CCH to eMSPs is divided into locations and tariffs. Locations describe a group of EVSEs (i.e., one or more CSs) and include the CSO's ID, a tag for whether the location is public or private, geolocation data, as well as information on the existing EVSEs (including the applicable tariffs per connector) and possibly information on the operator and/or owner of the location. Tariffs include the CSO's ID, a charging type (i.e., the applicable charging settings of fast, cheap, green, etc.), the validity time, and prices (relative to consumption and/or time).

Personal Data: The static CS data on its own does not contain any personal data, however, could become personal data once it is linked to a specific EV user, e.g., once a user charges their EV at a CS and a message is sent to the backend that contains a pseudonym of the user and a reference to the previously sent static CS data. Since static CS data is sent without a relation to a charging

[5] OCPI enables the optional inclusion of an ID of the user's contract with their energy supplier in addition to the mandatory eMAID (an ID of the user's contract with their eMSP) in the ID Token information data structure in order to support the use of the user's own energy supplier at a CS [142]. In the current EV charging infrastructure the CSO usually has a energy contract with the energy supplier and the consumption cost is passed on the eMSP (based on their roaming contract) who bills the user (based on their e-mobility contract) [52]. However, some stakeholders are interested in the possibility of an energy contract between supplier and EV user (potentially via an eMSP) [52]. While OCPI does not offer a precise explanation for the optional inclusion of an energy contract ID in their ID Token information data structure, it is presumably present in order to enable the use case of an energy contract between supplier and EV user via an eMSP. While this use case is out of scope for our system model, the energy contract ID is still included in the analysis of personal data for completion.

session or to an EV user and independently of the actions of EV users, we do not consider it to be personal data at this point. However, references to the static CS data (e.g., via location ID) in other messages, which relate to an EV user or their actions are considered personal data.

OCPP Setup: OCPP Communication is set up between CSOs and their CSs. OCPP starts by setting up the WebSocket connection (cf. Fig. 2.12), whereby the CS is identified towards the CSO (as previously mentioned in Section 4.1.3, we assume by using a certificate as per security profile SP_3). Afterwards, the CS is registered at the CSO (cf. Fig. 2.13). During registration, the CS sends general data about itself (e.g., its serial number, model, and Subscriber Identification Module (SIM) card data like the International Mobile Subscriber Identity (IMSI)) to the CSO who can accept, deny, or postpone the registration. After the CS was accepted, it sends status information about all its EVSEs/connectors to the CSO. Finally, the CSO can send ID Token lists to their CSs for the offline authorization of EV users.

Personal Data: The general data during OCPP setup as well as the CS status information does not relate to EV users, i.e., is not personal data. The personal data of ID Token lists is detailed in the description of the OCPI data provisioning process.

ISO 15118 Setup: ISO 15118 communication is set up per charging session between the EV's EVCC and the CS' SECC. The ISO 15118 communication begins with the setup as shown in Fig. 2.5, during which general communication parameters as well as IDs of both EVCC and EVSE are exchanged. Afterwards, the session is started with service discovery and selection (cf. Fig. 2.6).

Personal Data: The EVCC's ID uniquely identifies the EV and with that clearly constitutes personal data. Any pseudonym of the CS that is sent for an ISO 15118 session (i.e., the SECC IP, CS ID, and EVSE ID) uniquely identifies a location (since it identifies a static CS) and can be linked to an EV (since different messages in the same session contain identifiers of the EV). Since this information can be used to link an EV (user) to a specific location, we consider the CS' pseudonyms during an ISO 15118 session as personal data. Similarly, we consider any timestamps in an ISO 15118 session to be personal data since they can be linked to the EV (user) and reveal the time and duration of the users' charging sessions. The EVCC's and SECC's supported transport layer information from the SDP exchange, i.e., security (TLS or not) and transport protocol (currently only TCP) is too general to constitute personal data. Similarly, information about the EVCC's supported application layer protocols (protocol namespace, version, schema ID and priority) as well as the SECC's selected schema ID are too general to be personal data unless a wide variety of different namespaces and versions is introduced in the future. The SECC's offered services and payment options, while they could relate to the EVCC (e.g., PnC is only offered if the EVCC supports TLS), are considered too be general to constitute personal data. While the EVCC's service and payment selection could be considered personal as they are influenced by the EV user's actions, the currently available selection are also considered to be too general[6] to have any significant relation to a particular individual. However, since ISO 15118 allows the definition of arbitrary custom services and arbitrary service parameter, a user's service selection could arguably become personal data in the future if the selection becomes specific enough to uniquely identify users.

Live CS Status: After the EV was connected to the CS, the CS sends a status update to its CSO informing them about the now occupied connector. The CSO distributes this update to associated eMSPs via the CCH.

[6] Possible options for services are EV charging (AC or DC), generic internet access, certificate installation, and custom. Possible options for payment are PnC and EIM.

Personal Data: Live CS status data is updated as a result of the EV user's actions (i.e., plugging in the EV) and provides information about the location and duration of charging session without an identifier of the corresponding EV user. As demonstrated in the previously mentioned work of Stegelmann and Kesdogan [184, 185] (cf. Section 2.1.3), even anonymous charge session locations provide the risk of being linked to the same user and could thus be used in a re-identification attack. Under consideration of this risk, we argue that live CS status data can be considered personal data.

Proprietary Protocol Setup: Communication between the EV user and their eMSP (e.g., via smartphone app) is assumed to be secured with TLS using unilateral server-side authentication for the eMSP (cf. Section 4.1.2). The proprietary protocol setup is assumed to authenticate the EV user to the eMSP.

Personal Data: The authentication of the EV user is assumed to contain personal data in the form of an identifier of the EV user (e.g., a username).

Table 4.1.: Data for Initial Communication Setup and Data Provisioning

Process	Data	EV (User)	CS	CSO	CCH	eMSP
OCPI Setup						
Versions (cf. [142], Section 6)	eMSP Protocol Version, Endpoints(Implemented Modules, URLs)				R	O
	CCH Protocol Version, Endpoints(Implemented Modules, URLs)			R	O	R
	CSO Protocol Version, Endpoints(Implemented Modules, URLs)			O	R	
Credentials (cf. [142], Section 7)	eMSP Credential, eMSP ID				R	O
	CCH Credential1, CCH ID				O	R
	CCH Credential2, CCH ID			R	O	
	CSO Credential, CSO ID			O	R	
OCPP Setup						
OCPP SP_3	CS ID		O	R		
Communication Start (cf. Fig. 2.13)	CS Data (Serial Number, Model, Vendor, Firmware Version, IMSI)		O	R		
	Registration Status, Heartbeat Interval		R	O		
	EVSE IDs, Connector IDs, Connector Status		O	R		
ID Token List						
Over OCPI (cf. [142], Section 12)	**List of ID Tokens (**incl. eMSP ID**, Token UID, eMAID, Token Group ID,** Validity, Language, Charging Settings, **Energy Contract ID, Update Time)**			R	R	O
Over OCPP (cf. [148], Section D)	**List of ID Tokens (incl. Token UID, eMAID, Token Group ID,** Validity, Language, Priority, EVSE ID**)**		R	O		
Static CS Data List						
CS Locations (cf. [142], Section 8)	List of CS Locations (incl. CSO ID, Location ID, Public/Private, Location Name, Geolocation Data (Address, City, Country, Coordinates), EVSEs (ID, Status, Capabilities, Connectors, Coordinates), Tariffs per Connector, Operator, Owner)			O	R	R

personal data in **bold** font, O = originator of data, R = recipient of data

Table 4.1.: Data for Initial Communication Setup and Data Provisioning *(cont.)*

Process	Data	EV (User)	CS	CSO	CCH	eMSP
Tariffs (cf. [142], Section 11)	List of CSO Tariffs (incl. CSO ID, Tariff ID, Type, Price Components, Start-/End Date)			O	R	R
ISO 15118 Setup						
General Setup (cf. Fig. 2.5a)	Security, Transport Protocol	O	R			
	SECC IP, Port, Security, Transport Protocol	R	O			
	CS ID[7]	R	O			
	App Protocols(Protocol Namespace, Version, Schema ID, Priority)	O	R			
	Schema ID	R	O			
V2G Session Setup (cf. Fig. 2.5b)	**EVCC ID**[8]	O	R			
	EVSE ID, Timestamp	R	O			
V2G Session Start (cf. Fig. 2.6)	EVCC's Services Scope	O	R			
	SECC's Offered Services, Payment Options, Service Parameters	R	O			
	EVCC's Selected Services, Parameters, Payment Option	O	R			
Live CS Status						
OCPP Update (cf. Fig. 2.14)	**EVSE ID, Connector ID, Status, Timestamp**		O	R		
OCPI Update (cf. [142], Section 8)	**Location ID, EVSE ID, Connector ID, Status, Timestamp**			O	R	R
Proprietary Protocol Setup						
General Authentication	**EV User ID**	O				R

personal data in **bold** font, O = originator of data, R = recipient of data

4.2.2. Charge Authorization

After the initial communication setup, the involved CS needs a confirmation that the EV user is authorized to charge. For this, each EV user received some kind of ID Token from their eMSP (e.g., in the form of an RFID card or PnC contract credentials) some time after concluding a contract with this eMSP (cf. Section 4.1). The CS can verify the user's authorization by confirming the user's ID Token with the respective eMSP. This process can be divided into three scenarios depending on how authorization is initiated (locally at CS or remotely over the eMSP) and with what kind of credentials (EIM or PnC) as described in the following paragraphs.

The data that is exchanged during the different charge authorization scenarios is shown in Table 4.2. Notation and formatting of the table correspond to that of Table 4.1.

Local EIM-Based Authorization (cf. Fig. 4.3): The EV user provides their ID Token, i.e., the RFID UID, locally to the CS. Afterwards, the CS determines the authorization status based on locally available

[7] The CS ID is included in the SECC's TLS certificate, which is required by the SECC's certificate profile as defined in ISO 15118 and the only relevant EV charging specific data defined in this profile (cf. [97], Table F.2).
[8] Uses the MAC address of the EVCC (cf. [97], [V2G2-879]).

data (i.e., offline authorization) or using a request to the backend. The EV periodically sends empty *AuthorizationReq* messages until it is cleared or denied to charge by the CS.

For authorization with local data, a precondition is that the eMSP distributes their users' ID Tokens as described in Section 4.2.1. If the user could not be authorized via local data at the CS, the CS needs to send an authorization request to its CSO. The CSO first tries to authorize the user based on locally available data (i.e., using the lists received from the CCH) and if this is not possible forwards the request to the CCH. If also the CCH cannot authorize the user based on its local data, the request is forwarded to the eMSP.

The OCPI request (from CSO to CCH to eMSP) contains the ID Token of the user and an ID of the location/EVSE at which the user is requesting to charge. The OCPI response contains the authorization status (accepted, blocked, etc.), the full ID Token information – including UID and eMAID (both are always required in the OCPI ID Token information data structure), group ID, preferred language, etc. – as well as the locations/EVSEs at which this token may be used. This response is sent along the reverse path of the request back to the CSO. The CSO's message to the CS generally contains the same data as mentioned for the ID Token list (cf. Section 4.2.1) with the addition of an optional cache expiry date, i.e., for how long this response shall be stored in the CS's Authorization Cache. Finally, the CS either offers or denies to charge the EV.

Personal Data: The user's pseudonyms (RFID UID, eMAID, and optional energy contract ID) uniquely identify the EV user and with that clearly constitutes personal data in any message. As previously discussed for the ID Token list, the group ID, and the token update time are considered personal data. Similarly the preferred language, the charging setting, the eMSP ID, the charging priority, and the token validity status are not considered personal data. The user's authorization status is considered to be too general[9] (not high-dimensional or sparse) to constitute personal data. The location IDs and EVSE IDs are considered to be personal data because they are sent in the same messages as the user's ID Token and in response to the user's actions (i.e., after the user's authorization attempt). Thus, this location information can be directly linked to the user, revealing the user's current location. The cache expiry date is presumably calculated in relation to the current date and time of authorization. Assuming that the method of calculation for the cache expiry date is known, we argue it constitutes personal data as this date reveals date and time of authorization and is directly linkable to the EV user's pseudonyms.

Remote EIM-Based Authorization (cf. Fig. 4.4): The EV user informs the eMSP that they want to start charging at a specific EVSE, e.g., by selecting the EVSE in the eMSP's app or scanning a QR-Code at the CS that contains the corresponding EVSE ID. The user's request uses a proprietary protocol between EV user and eMSP but is assumed to at least identify the user and the EVSE at which they want to charge. The eMSP identifies the correct CSO by the EVSE's ID (which includes the CSO's ID; cf. [97], Section H.2) and sends a remote authorization request over the CCH to this CSO. The OCPI remote authorization request contains the same data as the OCPI responses in the local EIM-based authorization scenario. The following OCPP remote authorization request from CSO to CS only contains the actual ID of the ID Token (e.g., RFID UID or contract certificate eMAID), the EVSE's ID, as well as possibly a group ID and a charging profile (cf. Smart Charging in Section 2.4.2). The EV simply waits at the CS until it is cleared or denied to charge.

Reservations can be seen as a special kind of remote EIM-based authorization. The EV user requests a reservation at a specific location/CS/EVSE from their eMSP. The user's request could identify a location

[9] Possible values for the authorization status in OCPI are allowed, blocked, expired, not allowed, and no credit.

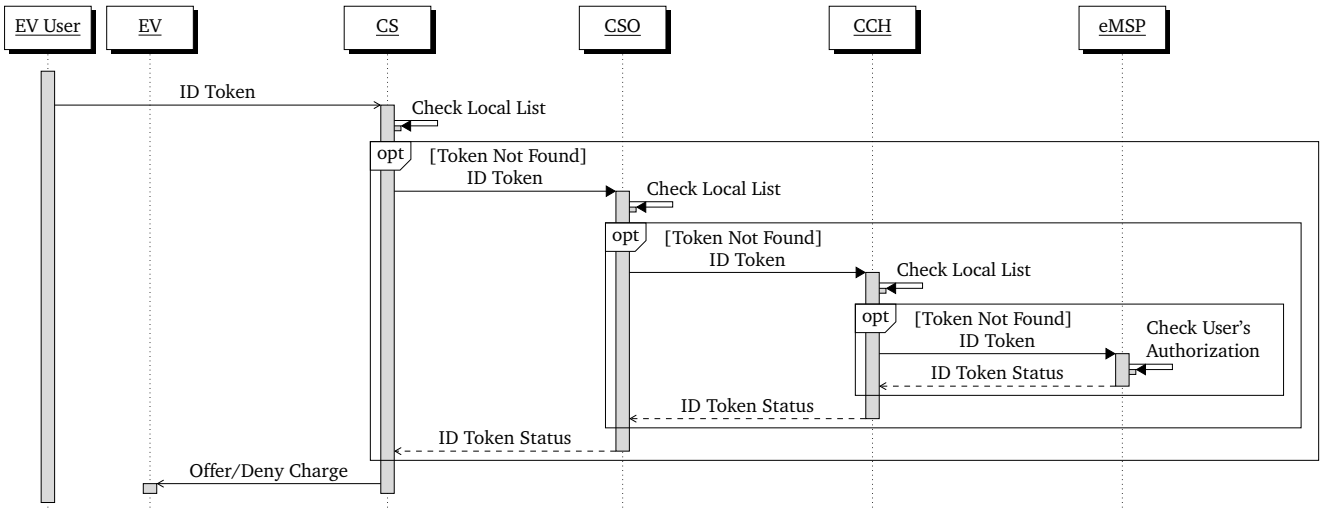


Figure 4.3.: Local EIM-Based Authorization

(instead of a specific EVSE) and general parameters like the expected time of arrival or the needed connector type. The eMSP sends a remote reservation request over the CCH to the CSO contains the same data as the remote authorization request with the addition of the reservation's expiry time. The OCPP remote reservation request contains the ID Token, an expiry date, as well as possibly a group ID, an EVSE ID, and/or a connector type.

The reservation request can be used as the user's authorization to charge once they arrive. However, opposed to remote EIM-based authorization case, the EV user still has to be identified/authenticated at the EVSE before they can start charging in order to verify that the reservation was for this user (cf. [148], Use case H03 and the reservation reference architecture from [15]). Hence, the EV user must present their ID Token at the CS, for which the local EIM- or PnC-based methods could be used (cf. the optional "Reservation" fragment in Fig. 4.4).

Personal Data: Remote EIM-Based Authorization generally involves the same personal data as the Local EIM-Based Authorization case, i.e., the same considerations apply. The only additions are the identification of the target charge location/EVSE, selected by the EV user, and the optional transaction-specific charging profile, calculated by the CSO. Since the user previously authenticated itself to the eMSP, the target charge location/EVSE can be linked to the user and indicates the user's current (or future in case of a reservation) location. This information is thus argued to constitute personal data. Transaction-specific charging profiles are considered personal data as they can be linked to a specific EV (user), contain timestamps that relate to the charging session, and contain charging parameters that are specific to the individual charging session.

In the case of a reservation, the EV user may additionally provide the expected time of arrival to the eMSP. The eMSP can use this information in the calculation of the expiry date, which is sent along with the reservation over OCPI and OCPP. The expected time of arrival and expiry date indicate the time at which the user is going to be at the location/EVSE and they can be directly linked to the user. This information is thus also argued to constitute personal data. The needed connector type in reservation requests is considered to be too general to constitute personal data.

Local PnC-Based Authorization (cf. Fig. 4.5): The EV (EVCC) authenticates itself at the CS (SECC). First, the EVCC sends its eMAID and locally stored contract certificate (including the chain of eMSP Sub-CA

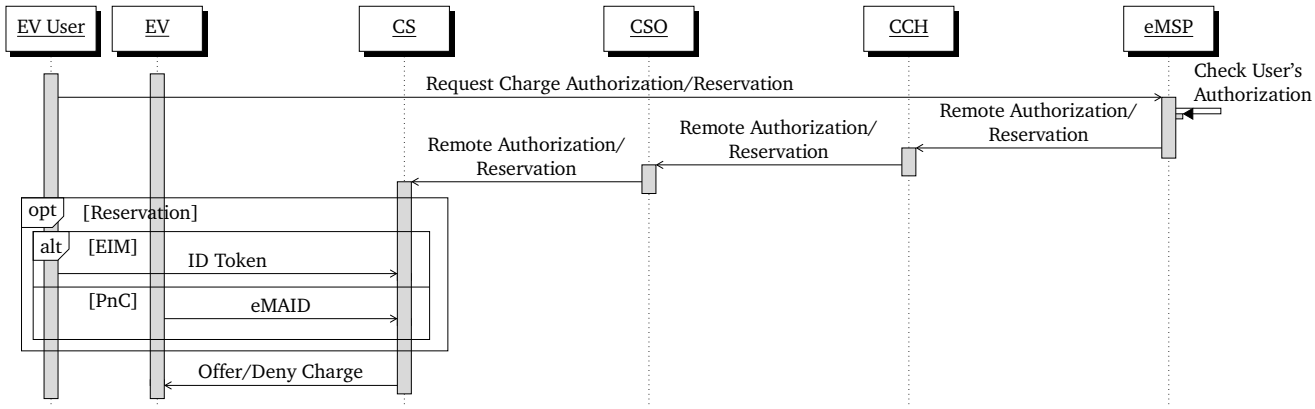


Figure 4.4.: Remote EIM-Based Authorization

certificates; cf. Fig. 2.10; not shown in Fig. 4.5), to the SECC. The SECC requests the revocation status of the certificate (chain) from their CSO. This request contains the previously received eMAID as well as the data required for OCSP requests for the EVCC's certificate chain (i.e., for each certificate a hash over the issuer's distinguished name, a hash over the issuer's public key, the certificate's serial number, and the URL of the OCSP responder). Note that the issuer's distinguished name in combination with certificate's serial number uniquely identifies a certificate (cf. [35], Section 4.1.2.2). If the CSO does not have the certificate status cached, they send OCSP requests to the corresponding (identified via a certificate extension) OCSP responders (e.g., the eMSP for the contract certificate). The CSO responds to the SECC with the status of the certificate chain (accepted, expired, revoked, etc.) and with the full ID Token information. The SECC uses the received revocation status information, the EVCC's contract certificate chain, and a locally installed eMSP root CA certificate as input to the certificate path validation. Alternatively, the CS could request a full certificate path validation for the EVCC's contract certificate chain from their CSO, i.e., instead of only requesting the revocation information (not shown in Fig. 4.5).

After the EVCC's certificate chain is validated and the user's authorization is confirmed, the SECC sends a 128 bit random nonce to the EVCC. The EVCC responds with a signature over the nonce (created with the private contract key) and the CS verifies this signature using the public key from the EVCC's contract certificate.

Depending on the agreement between CSO and eMSP, a successful authentication of the EVCC could be used as implicit authorization for the user to charge (i.e., if every user with a valid contract certificate under this eMSP's root CA is authorized to charge at every CS of this CSO; cf. [96], Use case D1 and [148], Requirement C07.FR.12). Alternatively, an explicit authorization of the contract certificate's eMAID might be required (i.e., using the eMAID as ID Token for the process in Fig. 4.3; not shown in Fig. 4.5; cf. [96], Use case D2). Additionally, using the contract certificate's eMAID as an ID Token for charge authorization is also required for PnC in case of a reservation (cf. Fig. 4.4) or if the CS is currently offline (i.e., if the SECC cannot request the certificate's revocation status then OCPP's offline authorization mechanisms are used as a replacement; cf. [148], Requirement C07.FR.09). Furthermore, the ID Token may have to be matched with the eMSP's ID Token list (at the CS or using a request to the backend as part of the "Contract Certificate" message in Fig. 4.3) because information like the user's language and charge setting are shared via this list.

Personal Data: Local PnC-Based Authorization generally involves the same personal data as the Local EIM-Based Authorization case, i.e., the same considerations apply. The main difference is that

PnC-based authorization adds a contract certificate chain and signature from the EVCC. The contract certificate uniquely identifies the EV user's contract with their eMSP, i.e., the contract certificate, or more specifically, any of its unique values like the eMAID in the subject name or the contained public contract key, can be seen as pseudonyms of the user and thus clearly constitute personal data. Additionally, the contract certificate's validity dates are personal data if they reflect a date that is directly related to the user, e.g., if they correspond to the start and the end of the user's e-mobility contract. The corresponding Sub-CA certificates identify the eMSP, i.e., reveal that the user has a contract with the eMSP, which is arguably too general to constitute personal data.

The OCSP request for the contract certificate includes personal data in the form of a hash over the issuer's distinguished name in combination with the certificate's serial number, which uniquely identifies the certificate and with that also the EVCC. Assuming that the issuer does not use its key pair for multiple CA certificates with different distinguished names, the combination of the hash over the issuer's public key and the certificate's serial number also uniquely identifies the contract certificate and can be considered personal data. The requestor name and the signature in the OCSP request identify the CSO, i.e., at most reveal that a user is charging at any CS of this CSO, which is arguably too general to constitute personal data. The OCSP response contains the same certificate identifying information as the request. Additionally, the response contains a timestamp identifying when it was generated, which can be directly linked to the certificate identifying information and with that to the EVCC. This timestamp personal data since it can be used to derive the approximate starting time of a user's charging session. The responder ID and the signature in the OCSP response identify the eMSP, i.e., reveal that the user has a contract with the eMSP, which is arguably too general to constitute personal data. The certificate status in the OCSP response is also considered to be too general[10] to constitute personal data. OCSP extensions are assumed to not include further personal data.

The PnC-based authentication for ISO 15118 includes a timestamp from the SECC, which can be considered personal data since, as previously mentioned, any timestamps in an ISO 15118 session can be linked to the EV (user) and reveal the time and duration of the users' charging sessions. Additionally, the EVCC sends an ECDSA signature to the SECC. The EVCC's signatures can be considered personal data since, in combination with the contract certificate, they can be linked to the signing entity.

Table 4.2.: Data for Charge Authorization

Process	Data	EV (User)	CS	CSO	CCH	eMSP
Local EIM-Based Authorization						
Local (e.g., RFID Card)	**RFID UID**	O	R			
Over OCPP (cf. Fig. 2.14)	**ID Token (RFID UID)**		O	R		
Over OCPI (cf. [142], Section 12)	**ID Token (RFID UID), Location ID, EVSE ID(s)**			O	R	R

personal data in **bold** font, O = originator of data, R = recipient of data

[10] Possible values of the certificate status are good, revoked, and unknown.

Table 4.2.: Data for Charge Authorization *(cont.)*

Process	Data	EV (User)	CS	CSO	CCH	eMSP
	Authorization**, ID Token (**incl. eMSP ID**, Token UID, eMAID, Token Group ID,** Validity, Language, Charging Settings, **Energy Contract ID, Update Time), Location ID, EVSE ID(s)**			R	R	O
Over OCPP (cf. Fig. 2.14)	**ID Token Info (incl. Token Group ID,** Validity, Language, Priority, **EVSE ID, Cache Expiry)**		R	O		
Remote EIM-Based Authorization						
Proprietary Authorization	**EV User ID, Target Location and EVSE**	O				R
OCPI Authorization (cf. [142], Section 13)	**ID Token (**incl. eMSP ID, **Token UID, eMAID, Token Group ID,** Validity, Language, Charging Settings, **Energy Contract ID, Update Time), Location ID, EVSE ID**			R	R	O
OCPP Authorization (cf. Fig. 2.16)	**ID Token (Token UID, Group ID), EVSE ID, Charging Profile**		R	O		
Proprietary Reservation	**EV User ID, Target Location and/or EVSE, Reservation Parameters (Time of Arrival**, Needed Connector Type)	O				R
OCPI Reservation (cf. [142], Section 13)	**ID Token (**incl. eMSP ID, **Token UID, eMAID, Token Group ID,** Validity, Language, Charging Settings, **Energy Contract ID, Update Time), Location ID, EVSE ID, Expiry Date**			R	R	O
OCPP Reservation (cf. [148], Section H.2)	**ID Token (Token UID, Group ID), EVSE ID, Expiry Date,** Connector Type		R	O		
Reservation Consumption	**ID Token (cf. Local EIM-/PnC-Based Authorization)**	O	R			
Local PnC-Based Authorization						
Over ISO 15118 (cf. Fig. 2.6)	**eMAID, Contract Certificate Chain(Contract Certificate,** Sub-CA Certificates**)**	O	R			
Over OCPP (cf. [148], Section C.2.2)	**Contract Certificate Chain, ID Token (eMAID), OCSP Data (Hash Over the Issuer's Distinguished Name, Hash Over the Issuer's Public Key, Certificate's Serial Number, and URL of OCSP Responder)**		O	R		

personal data in **bold** font, O = originator of data, R = recipient of data

Table 4.2.: Data for Charge Authorization *(cont.)*

Process	Data	EV (User)	CS	CSO	CCH	eMSP
OCSP Query (cf. [168])	OCSP Request (Requestor Name, Certificate Requests(**Hash Over Issuer's Distinguished Name, Hash Over Issuer's Public Key, Certificate's Serial Number**, Optional Extensions), Optional Extensions, Optional Signature)			O		R[11]
	OCSP Response (Responder ID, Certificate Responses(**Hash Over Issuer's Distinguished Name, Hash Over Issuer's Public Key, Certificate's Serial Number**, Certificate Status, Optional Extensions), **Response Timestamp**, Optional Extension, Signature)			R		O
Over OCPI (cf. [142], Section 12)	**ID Token (eMAID), Location ID, EVSE ID(s)**			O	R	R
	Authorization, **ID Token (**incl. eMSP ID, **Token UID, eMAID, Token Group ID,** Validity, Language, Charging Settings, **Energy Contract ID, Update Time), Location ID, EVSE ID(s)**			R	R	O
Over OCPP (cf. [148], Section C.2.2)	Certificate Status, **ID Token Info (incl. Token Group ID,** Validity, Language, Priority, **EVSE ID, Cache Expiry)**		R	O		
Over ISO 15118 (cf. Fig. 2.6)	Challenge, **Timestamp**	R	O			
	Challenge, **Signature**	O	R			

personal data in **bold** font, O = originator of data, R = recipient of data

4.2.3. Charge Session

After user authorization, the CS registers a transaction with its CSO who forwards this information over the CCH to the eMSP. This transaction is used to report relevant information throughout the charging session (consumed energy, accumulating cost, etc.). Additionally, a charging profile is negotiated between EV and CS, with optional input from CSO and eMSP. Finally, the charging loop is entered whereby the CS periodically sends meter values to the EV and to the CSO (as updates to the transaction) who again forwards this information over the CCH to the eMSP.

These processes of a charging session, i.e., transaction management, profile negotiation, and charge loop, are shown in Fig. 4.6 and detailed in the following paragraphs. The data that is exchanged during the charge session processes is shown in Table 4.3. Notation and formatting of the table correspond to that of Table 4.1.

Start New Transaction: Once the EV user is authorized to charge at the CS, the CS informs its CSO about the start of a new transaction. The CS' transaction message includes a timestamp, the reason for the start (e.g., due to user authorization), information on the transaction (ID, charging state, and total charging time), the ID Token of the user (if the message was sent due to user authorization), the EVSE ID, the

[11] The OCSP responder for contract certificates is assumed to be the eMSP.

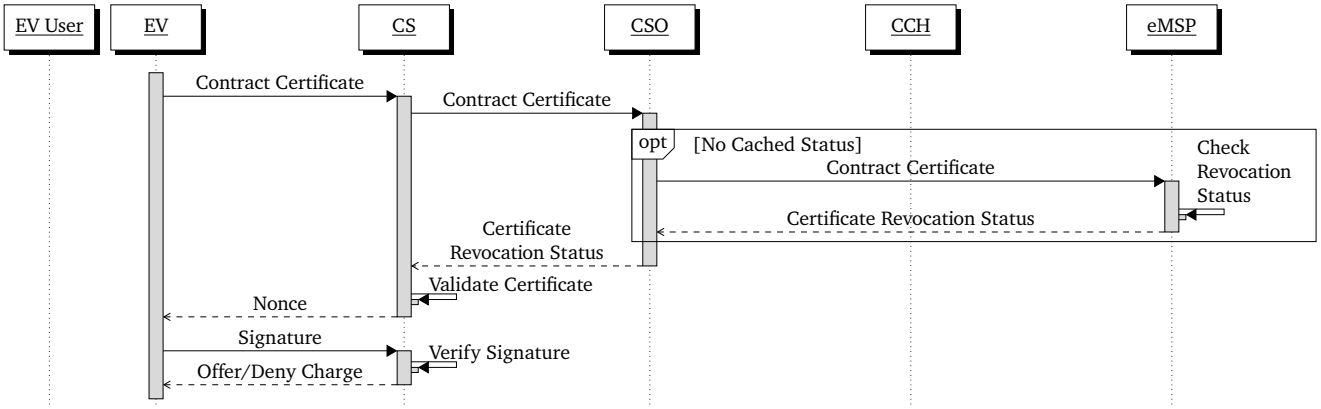


Figure 4.5.: Local PnC-Based Authorization

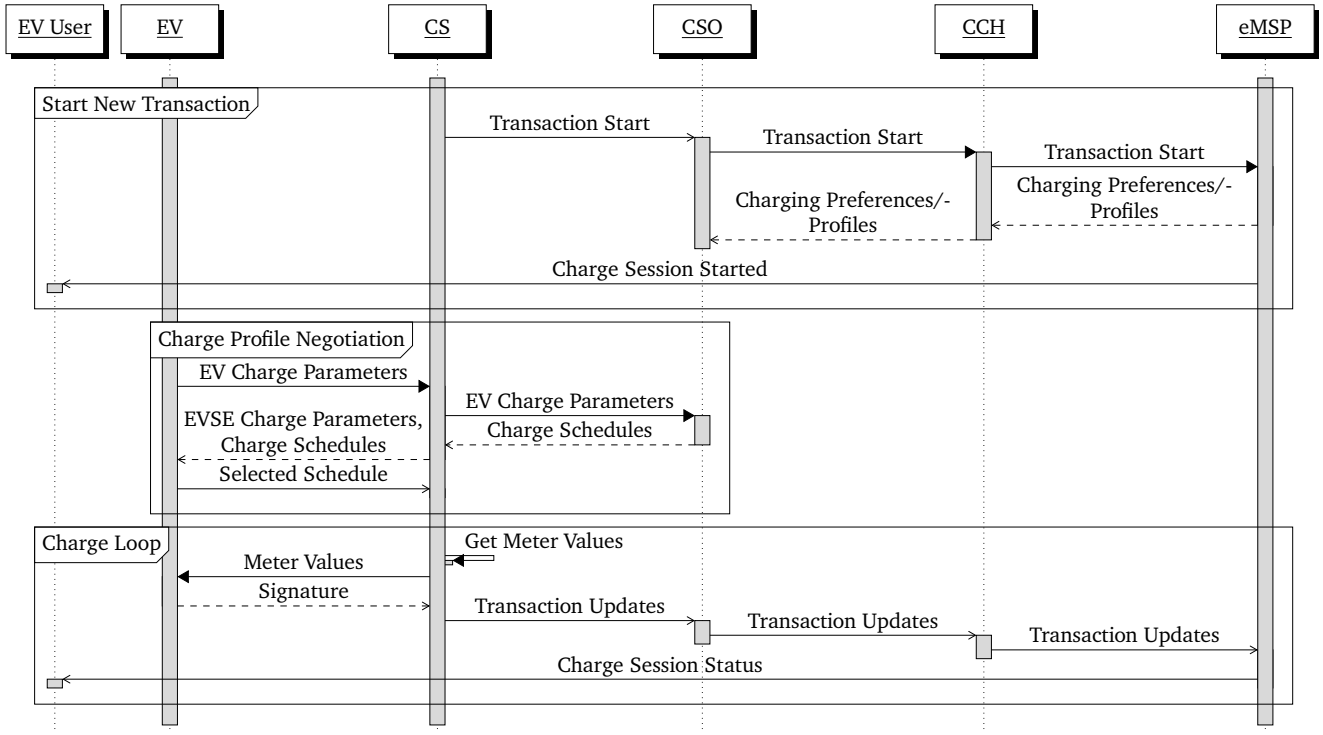


Figure 4.6.: Charge Session

connector ID, and the current (optionally signed) meter values. The CSO's response to the CS contains the full ID Token information and optionally a charging priority.

Information about the started transaction is sent from the CSO over the CCH to the respective eMSP. The corresponding OCPI messages include the CSO's ID, a transaction ID, a timestamp, the user's ID Token, the CS location with EVSE and connector, the total kWh and cost so far, and information on the individual charging periods including relevant information to derive the total cost (i.e., a starting timestamp, the applicable tariff, and values like the consumed energy or parking time). After the eMSP was informed about the started transaction, they may send the user's charging preferences (including the charging settings, the departure time, the needed energy, and whether V2G power transfer is allowed) and/or a charging profile (including the charging limits over time) to the CSO. Additionally, the eMSP may inform the EV user via their smartphone app about the started charging session and provide updated information on, e.g., the location, accumulated consumption, tariffs, and estimated cost.

Personal Data: Since a transaction is linkable to an individual user, any transferred IDs of the EV (user), location, CS, and EVSE as well as any timestamps are considered personal data. The transaction ID is a "transaction pseudonym" of the user and thus considered personal data, even though it involves the lowest privacy-risk of all pseudonym types (cf. the previously mentioned definitions from Pfitzmann and Hanse [161]). Since the energy meter is uniquely linked to a specific EVSE/location and the meter's identifiers (ID and public key) is sent in a specific transaction (i.e., linkable to an individual user), the meter's identifiers are considered personal data. If meter values are linked to a transaction, they can also be linked to an individual user. Meter values reveal the consumed energy of the session and can be linked to the corresponding charge parameters, -profiles, and -states of the transaction. This combination can be used to derive the elapsed time of a charging session. Hence, the meter values reveal session-specific charging parameters and are considered to be personal data. Similarly, information about the applicable tariff in combination with the charging cost can be used to derive the elapsed time is considered to be personal data. Additionally, even the charging cost on its own as well as meter values in combination with tariff information are considered personal data as they reveal information about the financial transactions of a user. The transaction trigger (reason for the start) and whether V2G power transfer is allowed are both considered to be too general to constitute personal data. The CSO ID, the charging settings (fast, cheap, etc.), and the ID Token's charging priority, validity status, and language are not considered personal data as previously discussed.

Charge Profile Negotiation: Before the EV can start charging, a charging profile must be negotiated with the CS. For this, the EV sends its charging parameters to the CS, which forwards them to the CSO. The CSO responds with a charging profile including multiple schedules that define the maximum allowed power and possible sales tariffs. Note that OCPI currently does not provide messages to send the required tariff information from eMSP to CSO. The CS forwards this charging profile along with its own charge parameters to the EV. Finally, the EV selects a schedule and may indicate its intended consumption (e.g., if the EV intends to consume less power than is allowed by the schedule; not shown in Fig. 4.6). If the EV sent its intended consumption, the CS forwards this information to the CSO.

Personal Data: Since charge profile negotiation is conducted for a specific transaction, all transferred data can be linked to the EV user that is associated with this transaction. Hence, EVSE IDs, intended departure time, needed energy, as well as charge profiles, -schedules, and -intent (all three containing timestamps and being linkable to a sessions meter values, tariffs, and costs) are considered personal data according to previous discussions. The EV's energy transfer mode (AC/DC)

is too general to constitute personal data. The remaining EV charge parameters (maximum voltage, maximum current, minimum current) reflect physical limits and are assumed to be too broad and unspecific (at most identifying the EV model) to be considered personal data. Similarly, the EVSE's charge parameters reflect physical limits (at most identifying the CS model) and thus not considered personal data.

Charge Loop: While the EV is charging, the CS periodically queries its electricity meter. The resulting meter information (including an ID of the meter, the measured value, a timestamp, and possibly a signature by the meter) as well as other charging data (e.g., maximum allowed current) are periodically sent to the EV. The CS optionally requests a signature over the meter values from the EV (i.e., a signature from the EVCC with its private contract key; called *metering receipt* in ISO 15118). Additionally, the CS sends the meter values to the CSO using a transaction update with the same data as for the transaction start except that the EVCC's signature may be included and the ID Token is omitted. The CSO sends the transaction update to the eMSP over the CCH using the same message as for the transaction start except that only the changed data is included. This data may be used by the eMSP to update the user on the progress of the charging session, e.g., via the user's smartphone app using a proprietary protocol (cf. "Charge Session Status" in Fig. 4.6).

During the charging loop, both the EV and the CS can re-initiate the negotiation for a charge profile (not shown in Fig. 4.6). Additionally, the CSO can send a new schedule to the CS to trigger a renegotiation (same message and data as in the Charge Profile Negotiation process) and the eMSP can send new charging preferences/-profiles to the CSO (same messages and data as in the Start New Transaction process).

Personal Data: Any information that is sent during the charge loop can be linked to a specific session and with that to an EV user. Hence, the EVSE IDs, charge schedules/-states/-periods, meter IDs, meter values, timestamps, session-/transaction IDs, costs, and tariffs are considered personal data according to previous discussions. Similarly, the CSO ID is not considered personal data according to previous discussions. The transaction information trigger, indicating the reason for the CS' transaction message (e.g., due to a periodic measurement), is considered to be too general to constitute personal data. Additionally, the indication of the SECC that a signature by the EVCC over the current meter values is required is not personal data.

Table 4.3.: Data for Charge Session

Process	Data	EV (User)	CS	CSO	CCH	eMSP
Start New Transaction						
Over OCPP (cf. Fig. 2.14)	**Timestamp,** Trigger, **Transaction Info (ID, Charging State, Total Time), ID Token (RFID UID or eMAID), EVSE, Starting Meter Value (incl. Timestamp, Value, Signature, Public Key)**		O	R		
Over OCPI (cf. [142], Sections 9&14)	CSO ID, **Transaction ID, Timestamp, ID Token (RFID UID, eMAID), Location, EVSE, Connector, Meter ID, Total kWh, Total Cost, Charging Periods (Start Date, Value, Tariff ID)**			O	R	R

personal data in **bold** font, O = originator of data, R = recipient of data

Table 4.3.: Data for Charge Session *(cont.)*

Process	Data	EV (User)	CS	CSO	CCH	eMSP
	Transaction ID, Charging Preferences (Charging Settings**, Departure Time, Needed Energy,** V2G Power Transfer Allowed**)**			R	R	O
	Transaction ID, Charging Profile (Start Date, Duration, Charging Periods with Rate Limits)			R	R	O
Over OCPP (cf. Fig. 2.14)	Charging Priority, **ID Token Info (incl. Token Group ID,** Validity, Language, Priority, **EVSE, Cache Expiry)**		R	O		
Proprietary Protocol	**Transaction ID, Location, EVSE, Connector, Timestamp, Total kWh, Total Cost, Total Time, Tariff**	R				O
Charge Profile Negotiation						
Over ISO 15118 (cf. Fig. 2.7)	EV Charge Parameters (Energy Transfer Mode, **Departure Time, Needed Energy**, Max Voltage, Max Current, Min Current)	O	R			
Over OCPP (cf. [148], Section K.5)	**EVSE ID,** EV Charge Parameters (Energy Transfer Mode, **Departure Time, Needed Energy**, Max Voltage, Max Current, Min Current)		O	R		
	EVSE ID, Charging Profile (ID, From Date, To Date, Transaction ID, Schedules (Start Date, Duration, Max Power Over Time, eMSP Tariffs Over Time)		R	O		
Over ISO 15118 (cf. Fig. 2.7)	EVSE Charge Parameters (Nominal Voltage, Max Current), **Charge Schedules (Max Power Over Time, eMSP Tariffs Over Time)**	R	O			
	Selected Charge Schedule, Charging Intent (Intended Max Power Consumption Over Time)	O	R			
Over OCPP (cf. [148], Section K.5)	**EVSE ID, EV's Charging Intent (Start Date, Duration, Max Power Over Time)**		O	R		
Charge Loop						
Over ISO 15118 (cf. Fig. 2.8)	**EVSE ID, Current Charge Schedule ID, Meter Info (Meter ID, Meter Reading, Signature By Meter, Timestamp),** Signature By EVCC Required	R	O			
	Session ID, Current Charge Schedule ID, Meter Info (Meter ID, Meter Reading, Signature By Meter, Timestamp), Signature By EVCC	O	R			
Over OCPP (cf. Fig. 2.15)	**Timestamp,** Trigger**, Transaction Info (ID, Charging State, Total Time), EVSE, Meter Values (incl. Timestamp, Value, Signature, Public Key)**		O	R		

personal data in **bold** font, O = originator of data, R = recipient of data

Table 4.3.: Data for Charge Session *(cont.)*

Process	Data	EV (User)	CS	CSO	CCH	eMSP
Over OCPI (cf. [142], Sections 9)	CSO ID, **Transaction ID, Timestamp, Total kWh, Total Cost, Charging Periods (Start Date, Value, Tariff ID)**			O	R	R
Proprietary Protocol	**Transaction ID, Timestamp, Total kWh, Total Cost, Total Time, Tariff**	R				O

personal data in **bold** font, O = originator of data, R = recipient of data

4.2.4. End of Charge Session and Billing

After the EV is done charging, the session needs to be stopped. This process can be initiated locally (by the EV user or the EVCC) or remotely (by the eMSP at the request of the user). Once the session is stopped, the CSO needs to be informed about the end of the corresponding transaction along with the respective meter values. This information may be sent to the eMSP, not for billing purposes (only the CDR is used for billing) but, e.g., to update the user on their charging session via user's app. Additionally, once the EV is unplugged from the CS, the CSO, CCH, and eMSP are informed about the live CS status change. Finally, all billing relevant data needs to be sent from the CSO to the eMSP in the form of a CDR.

The processes for the end and billing of a charge session (i.e., local-/remote session stop, transaction end, CS status update, and sharing billing data) are shown in Fig. 4.7 and detailed in the following paragraphs. The data that is exchanged during the charging session processes is shown in Table 4.4. Notation and formatting of the table correspond to that of Table 4.1.

Local Stop: The session is locally stopped by the EV or the user. For the user to stop the session locally, they first need to be authorized at the CS by presenting their ID Token (Fig. 4.7 assumes that the EV user could be authorized based on the CS' local data; cf. Fig. 4.3 for the full local EIM-based authorization process). Only the same ID Token that was used to start the session (or an ID Token within the same group) is allowed to stop the session. Otherwise, the process of verifying the user's authorization to stop the session is the same as for the start of the session (cf. Local EIM-Based Authorization in Section 4.2.2). For the EV (i.e., the EVCC over ISO 15118) to stop the session, it only informs the CS that the session is to be stopped. Authorization is provided implicitly since the EV maintains an authenticated communication channel to the CS throughout the session.

Personal Data: IDs of the user, location and EVSE as well as ID Token and authorization information are classified as personal- or non-personal data according to previous discussions. The message to stop the charging session over ISO 15118 only indicates whether the session is to be terminated or paused, i.e., contains no persona data.

Remote Stop: Alternatively, the session is stopped remotely by the eMSP at the request of the user. The process is generally the same as for the initial start of the session (cf. Remote EIM-Based Authorization in Section 4.2.2), except that less data is exchanged (only the transaction ID).

Personal Data: IDs of the user and transaction are classified as personal data according to previous discussions.

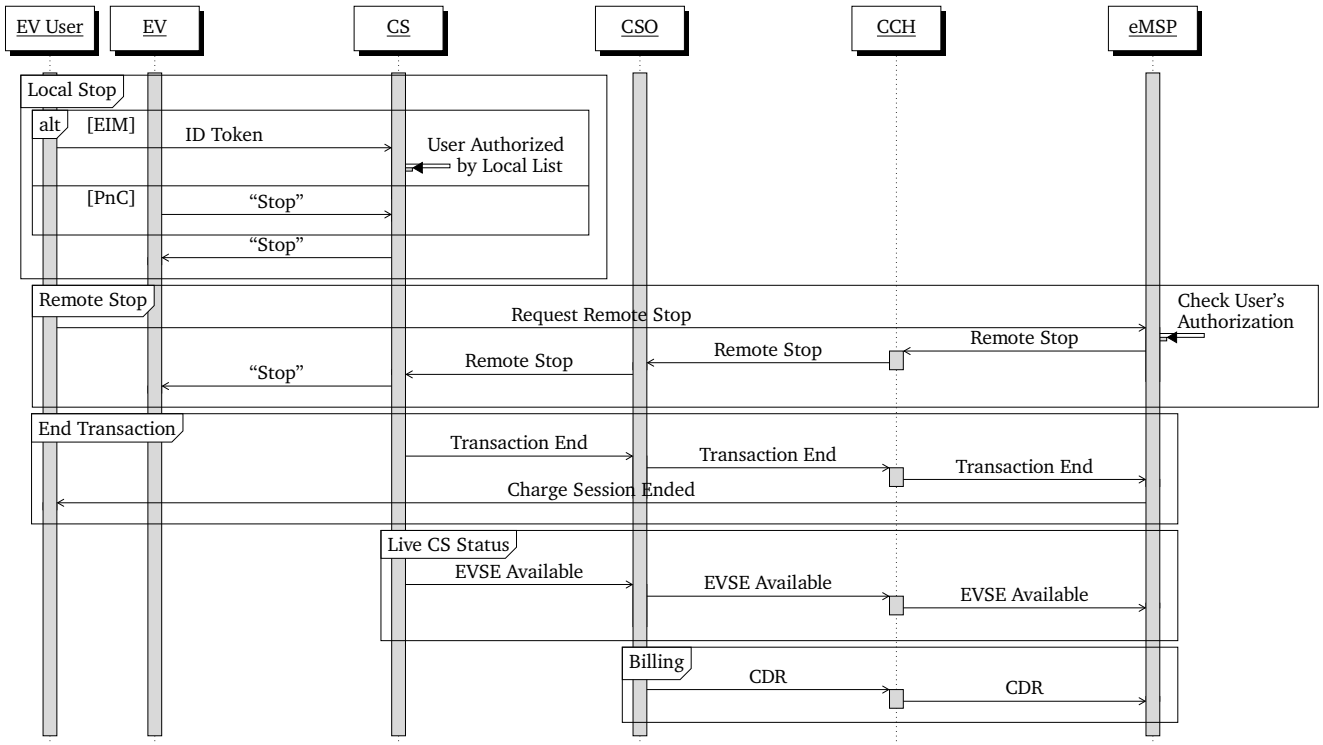


Figure 4.7.: End of Charge Session and Billing

End Transaction: After the charging session is ended, the CS sends an update for the corresponding transaction to the CSO and the CSO forwards this to the eMSP over the CCH. This process generally uses the same messages and data as the transaction updates during the charging loop (cf. Section 4.2.3). The only differences are that ID Token information may be exchanged between CS and CSO (in the case of a local EIM-based stop by the user) and that the CSO sends the total cost of the session to the CS.

Personal Data: All transaction data is classified as personal- or non-personal data according to previous discussions.

Live CS Status: Once the EV is unplugged from the CS, the CS sends a live status update to its CSO in order to inform them that the corresponding EVSE is now available. The CSO distributes this information over the CCH to associated eMSPs. The process and data is the same as for the initial connection of the EV (cf. Section 4.2.1).

Personal Data: The live CS status data is classified as personal data according to the previous discussion.

Billing: For the billing of charging sessions, the CSO generates a CDR for the session based on the transaction information it received from its CS. The CSO sends this CDR over the CCH to the eMSP. The CDR contains all billing relevant information, including, start and end date of the session, the user's ID Token, location information (coordinates of CS, EVSE ID, connector ID, etc.), applicable tariffs, consumption information and optional signatures.

Personal Data: A CDR directly relates to a specific transaction, i.e., it can also be uniquely linked to a specific EV user. Thus, the CDR ID and invoice ID are considered personal data. Additionally, the contained ID Token, transaction ID, meter ID, location information, charging periods (containing meter values and timestamps), total energy, and total time are considered personal data according

to previous discussions. Similarly, the tariff information in combination with the total cost is considered personal data according to previous discussions. The CSO ID is not considered personal data according to previous discussions.

Table 4.4.: Data for End of Charge Session and Billing

Process	Data	EV (User)	CS	CSO	CCH	eMSP
Local Stop						
Local (e.g., RFID Card)	**RFID UID**	O	R			
Over OCPP (cf. Fig. 2.15)	**ID Token (RFID UID)**		O	R		
Over OCPI (cf. [142], Section 12)	**ID Token (RFID UID), Location ID, EVSE ID(s)**			O	R	R
	Authorization**, ID Token (**incl. eMSP ID, **Token UID, eMAID, Token Group ID,** Validity, Language, Charging Settings, **Energy Contract ID, Update Time), Location ID, EVSE ID(s)**			R	R	O
Over OCPP (cf. Fig. 2.15)	**ID Token Info (incl. Token Group ID,** Validity, Language, Priority, **EVSE ID, Cache Expiry)**		R	O		
Over ISO 15118 (cf. Fig. 2.5b)	Stop(Type)	O	R			
	"Stop"	R	O			
Remote Stop						
Proprietary Protocol	**EV User ID, Transaction ID**	O				R
Over OCPI (cf. [142], Section 13)	**Transaction ID**			R	R	O
Over OCPP (cf. Fig. 2.16)	**Transaction ID**		R	O		
End Transaction						
Over OCPP (cf. Fig. 2.15)	**Timestamp,** Trigger, **Transaction Info (ID, Charging State, Total Time), ID Token (RFID UID or eMAID), EVSE, Meter Values**		O	R		
	Total Cost, ID Token Info (incl. Token Group ID, Validity, Language, Priority, **EVSE, Cache Expiry)**		R	O		
Over OCPI (cf. [142], Sections 9)	CSO ID, **Transaction ID, Timestamp, Total kWh, Total Cost, Charging Periods (Start Date, Value, Tariff ID)**			O	R	R
Proprietary Protocol	**Transaction ID, Timestamp, Total kWh, Total Cost, Total Time, Tariff**	R				O
Live CS Status						
OCPP Update (cf. Fig. 2.15)	**EVSE ID, Connector ID, Status, Timestamp**		O	R		
OCPI Update (cf. [142], Section 8)	**Location ID, EVSE ID, Connector ID, Status, Timestamp**			O	R	R

personal data in **bold** font, O = originator of data, R = recipient of data

Table 4.4.: Data for End of Charge Session and Billing *(cont.)*

Process	Data	EV (User)	CS	CSO	CCH	eMSP
Billing						
OCPI CDR (cf. [142], Section 10)	CSO ID, **CDR ID, Start-/End Date, Transaction ID, ID Token (UID, eMAID), Location (incl. Name, Address, City, Coordinates, EVSE ID, Connector ID), Meter ID, Tariffs (incl. Tariff ID, Price Components, Start-/End Date), Charging Periods (Start Date, Value, Tariff ID), Signed Data (Public Key, Plaintext, Signature), Total Cost, Total Energy, Total Time, Total Parking Time, Invoice ID**			O	R	R

personal data in **bold** font, O = originator of data, R = recipient of data

4.2.5. User Credential Provisioning

In order to implement the authentication of EV users at a CS, the provisioning of respective credentials is required. The provisioning of EIM credentials is assumed to use a secure out-of-band mechanism (e.g., sending an RFID card via postal mail) and not further considered. The processes for the provisioning of an EV user's contract credentials for PnC-based charge authorization is shown in Fig. 4.8 and detailed in the following paragraphs. Note that currently none of the surveyed roaming protocols support the exchange of contract credential provisioning messages for ISO 15118. For this reason, we assume that the corresponding ISO 15118 messages are simply forwarded over OCPI in their binary format (as is the case with OCPP 2.0.1; cf. [148], Section M.4).

The data that is exchanged during the provisioning of contract credentials is shown in Table 4.4. Note that all of the messages include personal data in the form of an EVCC's certificates, which uniquely identify the EV user either over their EV (via the PCID from the provisioning certificate) or over their charging contract (via the eMAID from the contract certificate).

Provisioning of Contract Credentials: In order to enable the installation of new contract credentials over ISO 15118, each EVCC is equipped with provisioning credentials from its OEM (cf. Section 4.1.1). The provisioning credentials are installed in a secure environment some time before the delivery of the vehicle and are uniquely identified by their PCID (included in the provisioning certificate). At the conclusion of a contract between EV user and eMSP, if their contract includes the PnC service, the EV user provides their PCID to the eMSP. Afterwards, we consider two possible procedures for contract credential provisioning: (i) without certificate pools and (ii) with certificate pools.

Provisioning of Contract Credentials without Pools: During the first charging session (before charge authorization), the EVCC sends a request for contract credentials to the SECC. This request includes the provisioning certificate and is signed with the corresponding private provisioning key. The request is forwarded from the CS to the CSO and from the CSO to the CPS (we assume that the CCH implements the CPS). The CPS/CCH verifies that the provisioning certificate as well as the signature are valid and forwards the request to the eMSP. Note that, in order to forward the request from CPS/CCH to the eMSP, the CPS/CCH either needs to broadcast the request (or at

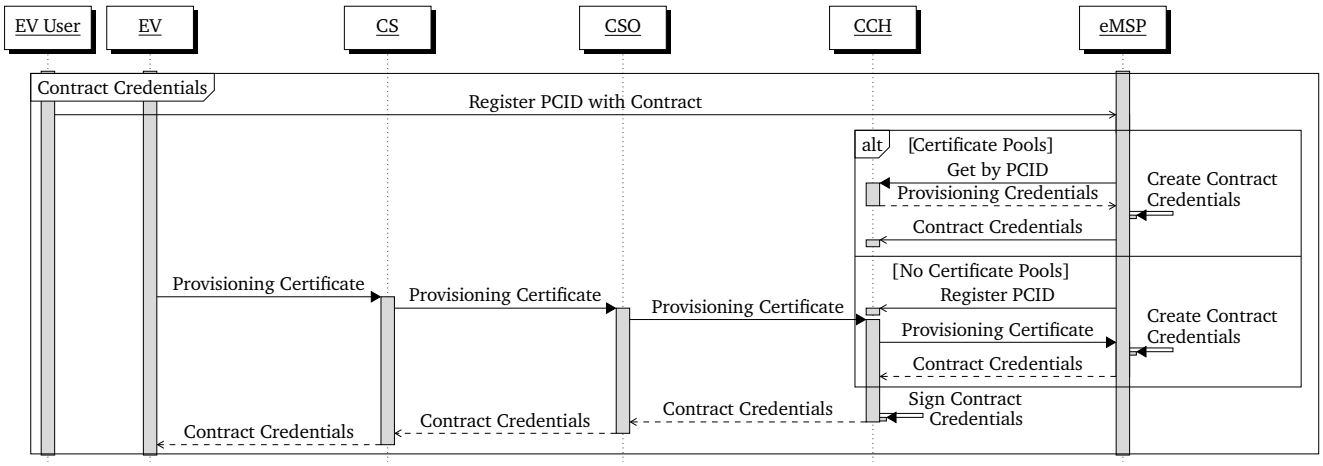


Figure 4.8.: User Credential Provisioning

least the PCID) to all possible eMSPs or needs to know which PCIDs are registered with which eMSPs; we assume the latter. If the PCID of the provisioning certificate was previously registered by an EV user, the eMSP generates new contract credentials and encrypts the corresponding private contract key with the public provisioning key from the provisioning certificate. More precisely, the private contract key is symmetrically encrypted with a session key that is generated based on ephemeral-static ECDH with a newly generated ephemeral ECDH private key and the static public provisioning key.

The eMSP sends the contract certificate along with the encrypted private contract key and the ephemeral ECDH public key to the CPS (i.e., the CCH). The CPS/CCH asserts the correctness and authenticity of this data and creates a signature over it. The signed contract credential data is forwarded over the CSO and the CS to the EVCC. The EVCC verifies the signature over the contract credentials and saves them for later use.

Provisioning of Contract Credentials with Pools: Since the exact interactions of backend systems are not defined in ISO 15118, different methods of implementing the contract credential provisioning process are possible in the backend. One such method is based on certificate pools and presented in the DKE application guideline VDE-AR-E 2802-100-1 [202]. This method is, for instance, implemented by Hubject [84] (the most established CA for EV charging [17] and operator of a CCH) and, while VDE-AR-E 2802-100-1 only has formal status in Germany, a similar method (also using certificate pools) is recommended by ElaadNL (a knowledge and innovation center for the smart charging field in the Netherlands.) in [51]. Furthermore, a recent position paper by CharIN (an international alliance of EV charging industry representatives) describing an ISO 15118 PKI certificate policy [29] also recommends the use of certificate pools as described in [202]. The main advantage of the certificate pool method is that it provides the temporal decoupling of the generation of contract credentials and the request for contract credentials as described in the following.

VDE-AR-E 2802-100-1 introduces two certificate pools: a provisioning certificate pool and a contract certificate pool; we assume both pools are implemented by the CPS/CCH. The provisioning certificate pool is used to store provisioning certificates, which are uploaded by the OEMs some time before the delivery of the corresponding vehicle (not shown in Fig. 4.8). The eMSPs have access to the provisioning certificate pool, which allows them to download a provisioning certificate

directly after an EV user registers their PCID with a charging contract. Hence, eMSPs can also generate contract credentials (with encrypted private key) for a user independently of the EVCC's contract credential request. Afterwards, contract credentials (together with the corresponding PCID) are uploaded into the contract certificate pool.

Finally, during the first charging session, the EVCC sends a request for contract credentials to the SECC, which forwards the request over the CSO to the CPS/CCH. The CPS/CCH retrieves the user's contract credentials from the contract certificate pool, signs them, and sends them back to the EVCC over the CSO and CS.

Personal Data: The PCID as well as the OEM provisioning certificate uniquely identify the user's EV, i.e., clearly constitute personal data. The contract certificate as well as the EVCC's signature are argued to be personal data in accordance with previous discussions. The list of the EVCC's installed V2G root certificates as well as the used ISO 15118 XML schema version are too general to constitute personal data. Since the private contract key is clearly personal data as it is used by the EVCC to authenticate itself. However, the involved privacy-risk is minimal since the key is encrypted, which arguably removes its legal status of personal data. Since the encrypted private contract key can be decrypted by anyone in possession of the private provisioning key in combination with the ephemeral ECDH public key, we consider the respective values in the OCPI, OCPP, and ISO 15188 messages, i.e., the encrypted private contract key in combination with the ephemeral ECDH public key, to be personal data. Finally, the CPS certificate chain and signature only identify the CPS are not considered personal data.

Table 4.5.: Data for User Credential Provisioning

Process	Data	EV (User)	CS	CSO	CCH	eMSP
Contract Credential Provisioning						
Out-of-band	**PCID**	O				R
Over OCPI (currently not supported)	**PCID**				R	O
Over ISO 15118 (cf. Fig. 2.9)	**EVCC's Signature, OEM Provisioning Certificate**, List of Installed V2G Root Certificates	O	R			
Over OCPP (cf. [148], Section M.4)	**EVCC's Signature, OEM Provisioning Certificate**, List of Installed V2G Root Certificates, ISO 15118 XML Schema Version		O	R		
Over OCPI (currently not supported)	**EVCC's Signature, OEM Provisioning Certificate**, List of Installed V2G Root Certificates, ISO 15118 XML Schema Version			O	R	
	OEM Provisioning Certificate				O	R
	Contract Certificate Chain, Encrypted Private Contract Key, Ephemeral ECDH Public Key				R	O
	Contract Certificate Chain, Encrypted Private Contract Key, Ephemeral ECDH Public Key, CPS Certificate Chain, CPS' Signature			R	O	

personal data in **bold** font, O = originator of data, R = recipient of data

Table 4.5.: Data for User Credential Provisioning *(cont.)*

Process	Data	EV (User)	CS	CSO	CCH	eMSP
Over OCPP (cf. [148], Section M.4)	**Contract Certificate Chain, Encrypted Private Contract Key, Ephemeral ECDH Public Key**, CPS Certificate Chain, CPS' Signature		R	O		
Over ISO 15118 (cf. Fig. 2.9)	**Contract Certificate Chain, Encrypted Private Contract Key, Ephemeral ECDH Public Key**, CPS Certificate Chain, CPS' Signature	R	O			

personal data in **bold** font, O = originator of data, R = recipient of data

4.3. Summary and Goals

As shown in the previous sections, a considerable amount of personal data is shared between the different actors of the EV charging infrastructure. Table 4.6 summarizes the personal data that is exchanged between these different actors. Most importantly, it shows that every actor learns pseudonyms of the EVs/EV users as well as their charge locations including precise timestamps and a variety of session-specific charge parameters (consumption, cost, etc.). These types of information have been shown to be privacy-sensitive (cf. [39, 40, 184, 185] as discussed in Section 2.1.3) and some of it, i.e., location data, is even considered especially sensitive data under the GDPR (cf. [19] as mentioned in Section 2.2.1). Since the pseudonyms of EVs/EV users remain the same for long periods of time (throughout the EV's lifetime/for the duration of the user's e-mobility contract) and considering the high resolution of the data (e.g., OCPI [142] timestamps are reported to the second, whereas the results of [39] were achieved with a temporal resolution of one hour) as well as the expectantly high frequency of charging sessions (possibly more than 3 times a day [176]), these circumstances enable actors in the EV charging infrastructure to create detailed movement profiles of users and infer habitual patterns (cf. Section 2.1.3), posing a high privacy risk. Because not all of the data is required by all actors, this situation is arguably not in line with the provisions of the GDPR.

For the protection of the user's privacy and in light of the GDPR's provisions (cf. Section 2.2), the dissemination of personal data should be minimized such that, e.g., no actor receives enough data to build detailed movement profiles of users. Regarding data minimization, we assume that CSOs must know the charge session's information (location, duration, consumption, etc.) in order to manage their CSs. Additionally, CSOs need to know to which eMSP a user has a contract in order to enable the CSO to eMSP billing. An eMSP on the other hand needs to know a charge session's information without the precise location but they must be able to identify the specific EV user in order to enable the eMSP to user billing. The CCH only needs to handle personal data in the process of contract credential installation since, in their role as CPS, they must be able find the user's credentials in the contract certificate pool and guarantee the authenticity of this data.

The goal is the development of a privacy-preserving solution for charge authorization and billing based on a privacy-by-architecture approach (cf. [182]; also refereed to as hard privacy [43]), i.e., enhancing privacy by architectural means like data minimization or de-identification. Privacy-by-policy measures (cf. [182]; also refereed to as soft privacy [43]), e.g., notice and choice mechanisms regarding the processing of personal data or mechanisms providing user awareness and ensuring policy compliance, are out of scope.

A privacy-preserving solution should enable the minimization of personal data, prevent actors from building movement profiles of users, and support existing authorization concepts. That is, local EIM- and PnC-based authorization methods should be supported via both whitelist-based offline authorization and online authorization requests that can be answered by any backend actor. EIM-based authorization should not put additional requirements on the EV. Additionally, the remote authorization and -reservation methods should be supported. A solution should only require minimal cryptographic operations on RFID cards and should not require hardware-supported security features or trusted hardware on CSs in order to avoid expensive hardware upgrades. However, the compromise of one CS (e.g., a leaked authorization whitelist) should not be able to be expanded to other CSs. On the other hand, an EV can use a TPM as a root of trust for PnC-based authorization (cf. Section 2.5.5).

For contract credential installation, in terms of usability, an EV user should not need to provide more than the PCID to an eMSP for PnC registration and an EV should not have to know the target eMSP for a *CertificateInstallationReq* message. The CCH, in its role as CPS, should be able to find the response(s) to a *CertificateInstallationReq* message without interaction with an eMSP, but eMSPs should still be able to generate responses to *CertificateInstallationReq* messages on demand, i.e., both methods described in Section 4.2.5 should be supported.

Based on the PET taxonomy from [80] (cf. Section 2.1.4) the goals for a solution can be summarized as follows:

Scenario: The involved actors in the scenario are EV user, EV, CS, CSO, CCH, and eMSP. The EV user, EV, and CS are not trusted. The backend actors are trusted from a security perspective but not from a privacy perspective.[12]

Aspect: Regarding the aspect dimension, a solution should hide the behavior of EV users in order to avoid the possibility of building movement profiles. Additionally, a solution should conceal the identities of EV users to anyone but their eMSP.

Aim: Regarding the aim dimension, a solution should provide for the confidentiality of personal data to an extent that adheres to the data minimization principle. Additionally, a solution should provide for the unlinkability of charging events in order to avoid the possibility of building movement profiles.

Foundation: A solution should be based on the commonly used computational security model [80]. In particular, due to the low computational power of RFID cards, an EIM-based method should only require symmetrical cryptography while a PnC-based solution may involve asymmetric means.

Data: Regarding the data dimension, a solution should address the types of data as defined by the presented protocols (cf. Section 4.2).

Trusted Third Party: No additional trusted third parties should be required. That is, trusted third parties should be restricted to the ones mentioned in the scenario and their involvement, regarding frequency, phase, and tasks, should be limited to their current involvement in EV charging process (e.g., regarding contract credential installation or offline-/online charge authorization; cf. Section 4.2).

Reversibility: The unlinkability of charging events should not be reversible and the concealment of EV user identities should only be reversible by their eMSP.

[12] The scenario is reflected by the adversary model (cf. Section 5.1).

Table 4.6.: Personal Data in the EV Charging Infrastructure

Process/Data (Personal Data Type)	EV (User)	CS	CSO	CCH	eMSP
Initialization					
ID Token Details List (EV User-/Group Pseudonym, User-Related Date)		R	R	R	O
EVCC ID (EV Identifier)	O	R			
SECC IP, CS ID, EVSE ID, Timestamp (Charge Location/-Time)	R	O			
Live CS Status (Charge Location/-Time)		O	R	R	R
EV User ID (EV User Pseudonym)	O				R
Local Charge Authorization					
If EIM: RFID UID as ID Token (EV User Pseudonym)	O	R	R		
If PnC: eMAID as ID Token, Contract Certificate/OCSP Data (EV User Pseudonym, User-Related Date)	O	R	R		
If PnC: OCSP Request Data (EV User Pseudonym)			O		R
If PnC: OCSP Response Data (EV User Pseudonym, Charge Time)			R		O
ID Token, Location-/EVSE ID (EV User Pseudonym, Charge Location)			O	R	R
ID Token Details, Location-/EVSE ID (EV User-/Group Pseudonym, User-Related Date, Charge Location)			R	R	O
ID Token Info, EVSE ID (Group Pseudonym, Charge Location/-Time)		R	O		
If PnC: Timestamp (Charge Time)	R	O			
If PnC: EVCC Signature (EV Identifier)	O	R			
Remote Charge Authorization (+*Reservation in Italic*)					
EV User ID, Target Location*/-Time* (EV User Pseudonym, Charge Location*/-Time*)	O				R
ID Token Details, Location-/EVSE ID, *Expiry Time* (EV User-/Group Pseudonym, User-Related Date, Charge Location*/-Time*)			R	R	O
ID Token Details, EVSE ID, Charging Profile *or Expiry Time* (EV User-/Group Pseudonym, Charge Location/-Parameters/-Time)		R	O		
ID Token (EV User Pseudonym)	O	R			
Start New Transaction					
ID Token, Location-/CS IDs, Timestamps, Transaction Info (EV User Pseudonym, Charge Location/-Time/-Parameters/-Cost)		O	R	R	R
Transaction ID, Charging Preferences-/Profile (EV User Pseudonym, Charge Time/-Parameters)		R	R	R	O
ID Token Info, EVSE ID (Group Pseudonym, Charge Location/-Time)		R	O		
Transaction ID, Location-/CS IDs, Timestamps, Transaction Info (EV User Pseudonym, Charge Location/-Time/-Parameters/-Cost)	R				O
Charge Profile Negotiation					
Departure Time, Needed Energy (Charge Time/-Parameters)	O	R	R		
EVSE ID, Charge Profile/-Schedules, Transaction ID (Charge Location/-Time/-Parameters, EV User Pseudonym)		R	O		
Charge Schedules (Charge Time/-Parameters)	R	O			
Selected Charge Schedule, Charge Intent (Charge Time/-Parameters)	O	R			
EVSE ID, Charge Intent (Charge Location/-Time/-Parameters)		O	R		
Charge Loop					
EVSE ID, Selected Charge Schedule, Meter Info (Charge Location/-Time/-Parameters)	R	O			
Session ID, EVCC Signature, Selected Charge Schedule, Meter Info (EV User Pseudonym, Charge Location/-Time/-Parameters)	O	R			

O = originator of data, R = recipient of data

Table 4.6.: Personal Data in the EV Charging Infrastructure *(cont.)*

Process/Data (Personal Data Type)	EV (User)	CS	CSO	CCH	eMSP
Transaction ID, EVSE ID, Timestamps, Transaction Info, Meter Values (EV User Pseudonym, Charge Location/-Time/-Parameters/-Cost)		O	R		
Transaction ID, Timestamps, Transaction Info, Meter Values (EV User Pseudonym, Charge Time/-Parameters/-Cost)			O	R	R
Transaction ID, Timestamps, Transaction Info (EV User Pseudonym, Charge Time/-Parameters/-Cost)	R				O
Local Stop (Only EIM)					
RFID UID as ID Token (EV User Pseudonym)	O	R	R		
ID Token, Location-/EVSE ID (EV User Pseudonym, Charge Location)			O	R	R
ID Token Details, Location-/EVSE ID (EV User-/Group Pseudonym, User-Related Date, Charge Location)			R	R	O
ID Token Info, EVSE ID (Group Pseudonym, Charge Location/-Time)		R	O		
Remote Stop					
EV User ID, Transaction ID (EV User Pseudonym)	O				R
Transaction ID (EV User Pseudonym)		R	R	R	O
End Transaction					
ID Token, CS ID, Timestamps, Transaction Info, Meter Values (EV User Pseudonym, Charge Location/-Time/-Parameters/-Cost)		O	R		
ID Token Info, Total Cost (Group Pseudonym, Charge Location/-Time/-Cost)		R	O		
Transaction ID, Timestamps, Total Cost/-kWh, Charging Periods (EV User Pseudonym, Charge Time/-Parameters/-Cost)			O	R	R
Transaction ID, Timestamps, Total Cost/-kWh/-Time (EV User Pseudonym, Charge Time/-Parameters/-Cost)	R				O
CS Status and Billing					
Live CS Status (Charge Location/-Time)		O	R	R	R
CDR (EV User Pseudonym, Charge Location/-Time/-Parameters/-Cost)			O	R	R
User Credential Provisioning					
PCID (EV Identifier)	O			R	R
OEM Provisioning Certificate with PCID, EVCC Signature (EV Identifier)	O	R	R	R	R
Contract Certificate with eMAID, Private Contract Key (EV User Pseudonym)	R	R	R	R	O

O = originator of data, R = recipient of data

5. Threat Analysis

This section presents the main part of the STRIDE- and LINDDUN-based threat analysis, i.e., *Step 5* to *Step 7* from the process described in Section 2.3.2, of the EV charging system model from Section 4. We describe the considered adversary model, build DFDs of the system model, and identify threats to the system based on the STRIDE and LINDDUN methodologies.

5.1. Adversary Model

The adversary model is based on the framework for the classification of Internet of Things (IoT) adversaries from [46] (vehicular use cases are considered as part of IoT in [46]). The framework defines the possible classes of adversaries as active and passive. Active adversaries can further be divided into remote- and physical adversaries; whereby, remote adversaries could be Internet- or intranet-based. In the EV charging use case it is important to consider multiple types of adversaries, covering different classes from [46] as follows:

Network Hacker: The Network Hacker is an active remote adversary targeting any used communication channel and is based on the Dolev-Yao model (cf. [47]). The adversary can act as a MitM on the channel, i.e., intercept, modify, or drop any messages; however, they cannot break any of the deployed cryptography. The adversary has two main goals. For one, they want to collect any personal data of EV users in order to build detailed movement profiles of the users, which, e.g., could be used to rob a user's home in their absence [65, 89]. Secondly, they want to charge their EV on the cost of another user, e.g., by using replay attacks.

Local Adversary: As often pointed out (e.g., [160, 1, 115, 133, 46]), a purely remote adversary is generally too weak for a vehicular/IoT setting as devices are commonly left unattended in public areas resulting in a high risk of physical compromise. Hence, we also consider the Local Adversary as an active physical adversary. The adversary has physical access to any unattended devices in public areas, i.e., EVs/EVCCs and CSs/EVSEs/SECCs. They can physically tamper with, add, or remove any respective components. Furthermore, they can extract or modify any data stored on the devices (e.g., steal private keys or install a malicious firmware image) unless this data is tamper protected (e.g., by a TPM). The adversary has the same goals as the Network Hacker, i.e., collect any personal data of EV users and/or charge their EV on another users cost.

Curious Operator: The potential of ill-intentioned operators raises serious privacy concerns [125]. However, modeling these operators as active adversaries (e.g., following the Dolev-Yao model) is generally too strong as they are restricted by regulations, audits, and the desire to maintain reputation [158]. Hence, we consider the Curious Operator as an honest-but-curious operator of any backend system, i.e., a passive adversary. Following the definition of an honest-but-curious adversary from [158], the Curious Operator is a legitimate operator of a backend system who does not deviate from the defined protocol

but attempts to learn as much information as possible from legitimately received messages. For example, operators could try to build movement profiles of EV users and/or sell information to other companies for targeted advertising [119].

The assumptions made during threat analysis are listed in the following:

- Backend actors (including their processes and data stores) are assumed to be secure (but not trusted with all personal data; cf. the Curious Operator).
- Systems are assumed to be secure against run-time attacks (e.g., against buffer overflow attacks [36]).
- Non-network data flows, i.e., between a process and a data store or between two sub-processes on the same system, are assumed to be secure.
- TLS-protected data flows are assumed to be secure.
- The used cryptographic primitives are assumed to be secure.
- The authentication of EV users to eMSP via the proprietary protocol (e.g., smartphone app-based) is assumed to be secure.
- The OCSP application layer authenticity protection mechanism is assumed to be secure.
- The initial provisioning of actors – including providing provisioning credentials to EVs, providing EIM credentials to EV users, providing OCPI credential tokens, and providing TLS credentials – is assumed to be secure.
- We assume that lower-layer (network and below) protocols have privacy-preserving measures in place, e.g., using appropriate MAC address randomization [128] and IP address randomization [138] techniques in order to prevent the linking of an EV's charging sessions based on this information.
- We assume that the minimum required data of backend actors (cf. Section 4.3) does not allow them to build movement profiles of EV users. Note that CSOs receive load management data and charge session locations as part of their minimum required data (cf. Section 4.3), which might enable them to link charging events of the same EV (user) (cf. [185] as discussed in Section 2.1.3). Hence, in practice, data generalization techniques for load management data might be required for this assumption to hold [185].
- We assume that the risk of linking an individual's movement profile to the individual's identity (e.g., by using public information such as geo-localized tweets or -pictures; cf. Section 2.1.3) is sufficiently high.
- We assume that live CS status information is publicly accessible and updated in near-real time (e.g., via a map that show EV users the occupation status of CSs).
- Data stores and processes of an EV user's EIM devices is assumed to be secure (e.g., devices with appropriate common criteria certification [108]).

5.2. Security and Privacy Threats

Based on the system model from Section 5.1 and the adversary model from Section 4, we show the high-level DFD of the EV charging infrastructure in Fig. 5.1.[1] The breakdown into more detailed sub-DFDs is shown in Appendix A.

The EV user's devices for EIM authorization (e.g., smartphone or RFID card) and the interactions between user and device (non-network data flows; cf. assumptions in Section 5.1) are assumed to be secure. Hence, for simplicity, the respective data flows can be modeled as directly originating from and/or terminating at the EV user. Similarly, the interactions between user and EV are assumed to be secure and, hence, we model the respective data (e.g., the user's charge preferences or tariff selections) as directly originating from the EV.

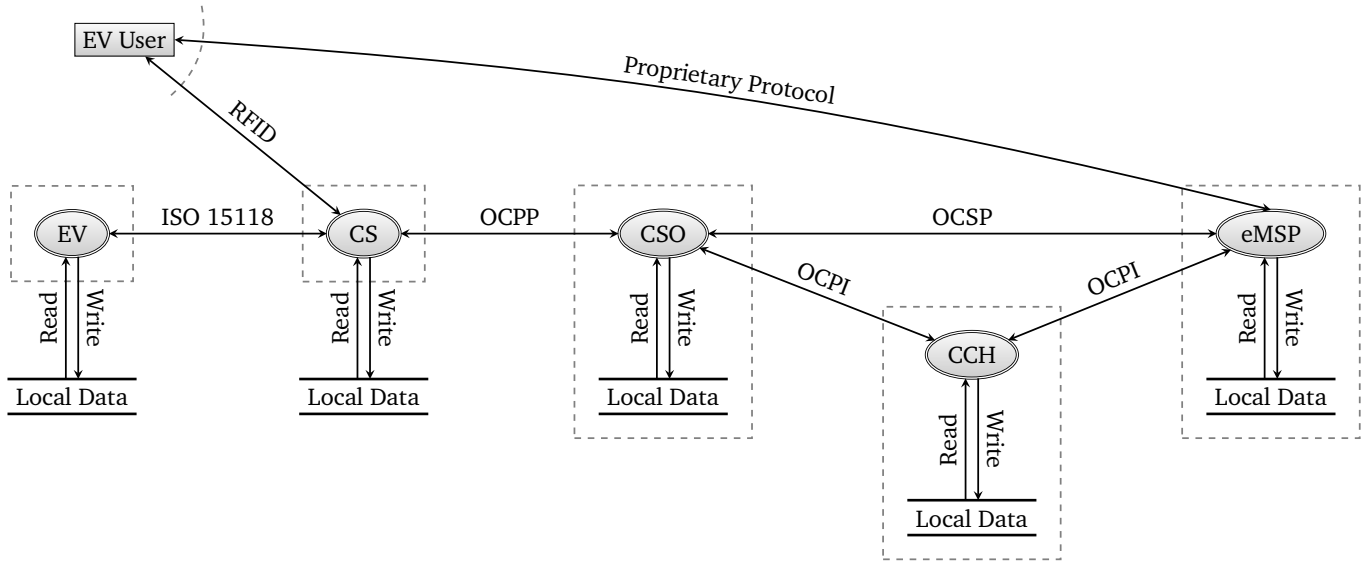


Figure 5.1.: High-Level EV Charging DFD

The trust boundaries around EV and CS indicate that the Local Adversary could extract or modify any stored data and that run-time attacks are out-of-scope. The trust boundaries around CSO, CCH, and eMSP indicate that the respective processes and data stores are secure (cf. assumptions in Section 5.1). Note, however, that these actors are not necessarily trusted to handle personal data due to the potential misuse by a Curious Operator. Additionally, all trust boundaries indicate that the Network Hacker has access to all network data flows. Note that this excludes the data flows between processes and data stores (non-network data flows).

5.2.1. Security Threats

This section describes the existing security threats (ST_x) in the EV charging infrastructure, resulting from our STRIDE-based threat analysis. Threats are documented in the form of misuse cases as recommended by LINDDUN (cf. Section 2.3.2). Note that for simplicity comparable DFD elements were grouped during analysis (cf. the concept of *reduction* as recommended in [82], Chapter 9) such that if the same threat applies

[1] In Fig. 5.1, the name of an entity is used as a simplified representation of the complex processes of this entity)

to multiple elements it only has to be analyzed once. Since addressing denial of service threats is not one of the goals of this thesis (cf. Section 4.3), denial of service threats are not analyzed.

Spoofing of an External Entity or Process: As listed in Section 5.1, we assume TLS authentication to be secure and any credentials of backend systems to be stored securely. Additionally, the authentication of the EV user to the eMSP via a proprietary protocol (e.g., smartphone app-based) is assumed to be secure. Furthermore, OCSP traffic is protected from spoofing by its application layer signatures. Hence, we only consider the following threats:

ST_1 *Spoofing of a CS:* If an adversary would obtain the TLS credentials from a CS (e.g., a Local Adversary could read them from the CS' storage), they would be able to spoof this CS towards the CSO and/or the EV (user). Spoofing the CS towards the CSO would allow the adversary to receive a list of valid ID Tokens (i.e., the Local Authorization List), which could be used to spoof affected EV users via the local EIM-based authorization mechanism. Additionally, it would allow the adversary to capture privacy-sensitive information like the reservations of EV users. Spoofing the CS towards the EV (user) would allow the adversary to receive privacy-sensitive information (e.g., IDs of EV and user) and security-sensitive information (EIM ID Tokens).

Assets, Stakeholders, and Threats: The authenticity of all CS processes towards EVs, EV users, and the CSO is threatened.
Primary Misactor: Local Adversary
Basic Flow: The misactor obtains the TLS credentials from a CS, allowing them to spoof this CS towards the CSO and EVs.
Alternative Flow: The misactor tampers with the firmware of a CS, allowing them to control and effectively spoof this CS towards the CSO, EVs, and EV users.
Trigger: CS connects to CSO (controlled by misactor; can always happen) or EV user tries to charge at a CS that is controlled by misactor.
Preconditions: For basic flow: ST_8 (alternative flow 2) or ST_9 (alternative flow 2 or alternative flow 6).
For alternative flow: ST_4 (basic flow).

ST_2 *Spoofing of an EV:* A Local Adversary could obtain the ISO 15118 credentials, i.e., contract and/or provisioning credentials, of an EV from its local storage. This would allow the adversary to spoof this EV towards CSs and towards backend actors. Hence, the adversary could receive new contract credentials and/or charge on the cost of the legitimate user.

Assets, Stakeholders, and Threats: The authenticity of all EV processes towards CSs is threatened. Since charge authorization and billing are based on this authenticity, this threat also affects the respective backend actors. Additionally, the authenticity of the EV towards the CCH/CPS or eMSP during th certificate installation process is threatened.
Primary Misactor: Local Adversary
Basic Flow: The misactor obtains the contract credentials of an EV, allowing them to spoof this EV to a CS.
Alternative Flow 1: The misactor tampers with the firmware of an EV, allowing them to control and effectively spoof this EV.
Alternative Flow 2: The misactor obtains the provisioning credentials of an EV, allowing them to spoof this EV to a CCH/CPS or eMSP. The misactor receives contract credentials enabling the basic flow.
Trigger: Controlled by misactor; can always happen.

Preconditions: For basic flow: ST_2 (alternative flow 2) or ST_{13} (alternative flow) or ST_{14} (alternative flow 1 or alternative flow 4).
For alternative flow 1: ST_5 (basic flow).
For alternative flow 2: ST_{13} (basic flow) or ST_{14} (basic flow or alternative flow 3).

ST_3 *Spoofing of a local EV user:* A Local Adversary could obtain an EV user's EIM ID Token from a CS' local storage (authorization whitelist/-cache). Additionally, ST_1 allows an adversary to receive an EV user's ID Token in messages from the CSO backend as well as directly from the user (during local EIM-based authorization). Furthermore, a Network Hacker may capture the user's ID Token during transit for local EIM-based authorization. Any of these cases would allow the adversary to spoof this EV user towards CSs and charge other EVs on the legitimate user's cost.

Assets, Stakeholders, and Threats: The authenticity of the EV user towards CSs is threatened. Since charge authorization and billing are based on this authenticity, this threat also affects the respective backend actors.
Primary Misactor: Local Adversary or Network Hacker
Basic Flow: The misactor obtains a user's EIM ID Token from a CS' local storage, allowing them to spoof this user to any CS by presenting the user's ID Token.
Alternative Flow 1: The misactor spoofs a CS to EV users. A user tries to authorize a charging session at this CS via local EIM-based authorization. The misactor's spoofed CS receives a user's ID Token, allowing the misactor to spoof this user to any CS.
Alternative Flow 2: The misactor spoofs a CS to a CSO. The CSO sends an authorization whitelist. The EIM ID Tokens from the whitelist can be used by the misactor to spoof all included users to any CS.
Alternative Flow 3: An EV user authorizes a charging session via local EIM-based authorization. The misactor captures the user's ID Token during transit, allowing them to spoof this user to any CS.
Alternative Flow 4: The misactor guesses a valid EIM ID Token, allowing them to spoof the corresponding user at any CS.
Trigger: Controlled by misactor after they obtain a user's ID Token; can always happen.
Preconditions: For basic flow: ST_8 (basic flow or alternative flow 1) or ST_9 (basic flow or alternative flow 1).
For alternative flow 1: ST_9 (alternative flow 5).
For alternative flow 2: ST_9 (alternative flow 3).
For alternative flow 3: ST_{11} (any flow).
For alternative flow 4: ST_{10} (any flow).

Tampering with a Data Flow, Data Store, or Process: As listed in Section 5.1, we assume that any backend processes and data stores are secure. Additionally, we assume that any non-network data flows and data flows using TLS are secure and run-time attacks are out of scope. OCSP traffic is protected from tampering by its application layer signatures and certificate root stores are considered to be part of a system's firmware. Hence, we only consider the following threats:

ST_4 *Tampering with the CS:* A Local Adversary could manipulate any of the CS' data stores. The adversary may tamper with the CS' firmware in order to arbitrarily change its behavior, i.e., any of its processes. Note that this may lead to ST_1 and enables the adversary to tamper with any message that is sent by the CS. Additionally, it allows the adversary to charge for free at the affected CS.

Assets, Stakeholders, and Threats: The integrity of the CS' data stores is threatened. Since the integrity of the CS' processes relies on the integrity of the CS' firmware data store, it too is

threatened.

Primary Misactor: Local Adversary

Basic Flow: The misactor tampers with a CS' firmware data store, allowing the misactor to arbitrarily change the CS' behavior.

Alternative Flow: The misactor tampers with a CS' ID Token whitelist and/or -cache data stores, allowing the misactor to charge for free at the affected CS.

Trigger: Controlled by misactor; can always happen.

Preconditions: The misactor has access to the CS.

ST_5 *Tampering with the EV:* A Local Adversary could manipulate any of the EV's data stores. The adversary may tamper with an EV's firmware in order to arbitrarily change its behavior. Note that tampering with an EV's firmware may also enable ST_2. The adversary could tamper with the EV of a legitimate user in order to have them sign ISO 15118 certificate installation/-update requests and/or authorization requests, thus, enabling the adversary to receive new valid contract credentials and/or charge on this user's cost.

Assets, Stakeholders, and Threats: The integrity of the EV's data stores is threatened. Since the integrity of the EV's processes relies on the integrity of the EV's firmware data store, it too is threatened.

Primary Misactor: Local Adversary

Basic Flow: The misactor tampers with an EV's firmware data store, allowing the misactor to arbitrarily change the EV's behavior.

Trigger: Controlled by misactor; can always happen.

Preconditions: The misactor has access to the EV.

Repudiation of an External Entity, Data Store, or Process: Since backend actors are modeled as honest-but-curious (cf. Section 5.1), they do not try to deny any of their actions. Hence, we only consider repudiation threats originating from the EV user, namely:

ST_6 *Repudiation of charge authorization:* Due to the existence of ST_2 and ST_3, a user could deny that they requested a charge authorization, i.e., claim that someone else used their credentials to charge on their account. Additionally, since an OCPP authorization request only contains the EV user's ID Token (and possibly contract certificate), it does not provide the CSO with the necessary information (e.g., a signature) to prove that the request was initiated by the respective EV user. Hence, ST_1 also provides ground for repudiation.

Assets, Stakeholders, and Threats: The non-reputability of charge authorizations is threatened. Since the billing of charging sessions is based on the information from the charge authorization (to identify which user is billed), this threat also affects the billing process.

Primary Misactor: Local Adversary

Basic Flow: The misactor spoofs an EV in order to charge on a legitimate user's account. The charge is repudiated by the user.

Alternative Flow 1: The misactor spoofs an EV user in order to charge on the legitimate user's account. The charge is repudiated by the user.

Alternative Flow 2: The misactor spoofs a CS and reports forged transactions (with ID Tokens of different users) to the CSO. The charge is repudiated by the users.

Trigger: Controlled by misactor; can always happen.

Preconditions: For basic flow: ST_2 (basic flow or alternative flow 1).
For alternative flow 1: ST_3 (any flow).
For alternative flow 2: ST_1 (any flow).

ST_7 *Repudiation of charge session data:* Since billing-relevant charge session data is not end-to-end authenticity protected (e.g., in the case of a user's charge parameter/tariff selection) or only optionally end-to-end authenticity protected (e.g., in the case of meter values), they are open to the threat of repudiation. Additionally, similar to ST_6, even if this data is protected, the existing spoofing and tampering threats provide ground for repudiation.

Assets, Stakeholders, and Threats: The non-reputability of charge session data is threatened. Since the billing of charging sessions is based on the session data (for calculating the cost), this threat also affects the billing process.
Primary Misactor: Local Adversary
Basic Flow: The misactor spoofs a CS and reports manipulated session data to the CSO. The session data is repudiated by the users.
Alternative Flow: All flows of ST_6 also lead to a repudiation of the session data.
Preconditions: For basic flow: ST_1 (any flow).
For alternative flow: ST_6 (any flow).

Information Disclosure of a Data Store, Data Flow, or Process: As listed in Section 5.1, we assume that TLS protects from information disclosure of a data flow. As lower-layer protocols are assumed to be privacy-preserving, the information disclosure threat does not apply SDP data (i.e., the IP and port of the SECC). Furthermore, information disclosure of backend processes and data stores is out of scope as backend actors are assumed to be secure. Hence, we only consider the following threats:

ST_8 *Information disclosure of the CS' data stores:* A Local Adversary could extract security-relevant data from the CS' stores. This security-relevant data includes ID Tokens, which would allow the adversary to impersonate legitimate users (cf. ST_3), and the CS' credentials, which would allow the adversary to impersonate this CS (cf. ST_1). The adversary could also extract privacy-sensitive information from the CS' stores. They could extract ID Tokens of users that have charged at this CS (from the authorization cache), threatening the affected users' location privacy if it can be linked with other data. Additionally, they could extract any auxiliary personal information about ID Tokens, i.e., the associated group ID and potentially the cache expiry time (cf. Section 4.2).

Assets, Stakeholders, and Threats: The confidentiality of the CS's data stores is threatened, including security-relevant data and personal data.
Primary Misactor: Local Adversary
Basic Flow: The misactor extracts ID Tokens from the CS' authorization whitelist.
Alternative Flow 1: The misactor extracts ID Tokens from the CS' authorization cache.
Alternative Flow 2: The misactor extracts the CS' credentials from the CS' data stores.
Trigger: Controlled by misactor; can always happen.
Preconditions: The misactor has access to the CS.

ST_9 *Information disclosure of the CS' processes:* By corrupting a CS' processes (cf. ST_4), an adversary could extract all of its local data (same data as in ST_8). Additionally, based on ST_1, the adversary could also extract any data that is not persistently stored at the CS but that is received or can be requested over OCPP and ISO 15118. This data includes privacy-sensitive information in the form of charge reservations, charge profile/schedule selections, charge session data (timestamps, meter values, etc.), and pseudonyms of EV users (the EVCC IDs and ID Tokens even if they are not cached). Additionally, side channel attacks (e.g., power-trace- or timing-based) could be used in order to extract a CS' private keys during their usage, enabling ST_1.

Assets, Stakeholders, and Threats: The confidentiality of the CS's data stores and transient data is threatened, including security-relevant data and personal data.
Primary Misactor: Local Adversary
Basic Flow: The misactor tampers with a CS' processes, allowing them to extract ID Tokens from the CS' authorization whitelist.
Alternative Flow 1: The misactor tampers with a CS' processes, allowing them to extract ID Tokens from the CS' authorization cache.
Alternative Flow 2: The misactor tampers with a CS' processes, allowing them to extract the CS' credentials from the CS' data stores.
Alternative Flow 3: The misactor spoofs a CS to the CSO, allowing them to receive transient data over OCPP.
Alternative Flow 4: The misactor spoofs a CS to an EV, allowing them to receive transient data over ISO 15118.
Alternative Flow 5: The misactor spoofs a CS to EV users. A user tries to authorize a charging session at this CS via local EIM-based authorization. The misactor's spoofed CS receives a user's ID Token.
Alternative Flow 6: The misactor extracts a CS' private key via side channel attacks.
Trigger: Controlled by misactor; can always happen.
Preconditions: For basic flow and alternative flow 1 and 2: ST_4 (basic flow).
For alternative flow 3 and 4: ST_1 (any flow).
For alternative flow 5: ST_1 (alternative flow).
For alternative flow 6: The misactor has access to the CS.

ST_{10} *Information disclosure at the local EIM authorization process:* A brute force attack on RFID UIDs as shown in [38] could be used to enumerate all valid UIDs, i.e., valid EIM ID Tokens. This brute force attack could also reveal which ID Tokens were used to charge at the attacked CS based on the timing difference between authorization via the CS' cache and via an OCPP request to the backend.

Assets, Stakeholders, and Threats: The confidentiality of ID Tokens and, potentially, of the CS' authorization cache are threatened.
Primary Misactor: Local Adversary
Basic Flow: The misactor conducts a brute force attack on RFID UIDs at a CS, allowing them to guess valid ID Tokens.
Alternative Flow: The misactor conducts a brute force attack on RFID UIDs at a CS and tries to validate, based on the response timing, whether tokens were authorized based on the CS' cache and via an OCPP request.
Trigger: Controlled by misactor; can always happen.
Preconditions: The misactor has access to the CS.

ST_{11} *Information disclosure of the local EIM authorization data flow:* A Network Hacker can obverse any messages that are not protected. Hence, the adversary could eavesdrop on the local EIM-based authentication mechanism in order to extract a user's ID Token, which enables ST_3. Additionally, RFID tags usually send their UID without any validations after being powered by a reader [114], which could be abused by an adversary in close enough proximity[2] in order to extract a user's ID Token.

Assets, Stakeholders, and Threats: The confidentiality of an EV user's ID Token is threatened.
Primary Misactor: Network Hacker

[2]The read range could be as high as 30 meters depending on the used technology [120].

Basic Flow: The misactor eavesdrops on the RFID traffic between the EV user's tag/card and a CS, which allows them to obtain the user's ID Token.
Alternative Flow: The misactor powers the EV user's tag/card, which allows them to obtain the user's ID Token.
Trigger: For the basic flow: Controlled by misactor; can happen when an EV user uses the local EIM-based authentication mechanism.
For the alternative flow: Controlled by misactor; can always happen.
Preconditions: For the basic flow: The misactor is close enough to eavesdrop on the RFID traffic.
For the alternative flow: The misactor is close enough to power the EV user's tag/card.

ST_{12} *Information disclosure of the OCSP data flow:* OCSP usually does not use TLS in order to avoid the possibility of circular dependencies (cf. [35], Section 8) and the EV charging protocols do not require a secure transport for OCSP messages. Hence, OCSP messages are most likely transferred over HTTP[3] [116, 219] and a Network Hacker can obverse the OCSP messages for PnC-based authentication between the CSO and the eMSP. Since the OCSP request uniquely identifies the contract certificate it also serves as a pseudonym of the EV user. Thus, this threat can cause privacy issues.

Assets, Stakeholders, and Threats: The confidentiality of an EV user's pseudonym is threatened. Also, the approximate start time of a charging session is revealed.
Primary Misactor: Network Hacker
Basic Flow: The misactor eavesdrops on the OCSP traffic between CSO and eMSP.
Trigger: Controlled by misactor; can happen when OCSP messages are sent.
Preconditions: The OCSP traffic is not confidentiality protected.

ST_{13} *Information disclosure at the EV's data stores:* Similar to ST_8, a Local Adversary could extract security-relevant data from the EV's stores, namely, its provisioning and contract credentials. These credentials would allow the adversary to impersonate the legitimate EV, i.e., ST_2.

Assets, Stakeholders, and Threats: The confidentiality of the EV's data stores is threatened, including security-relevant data.
Primary Misactor: Local Adversary
Basic Flow: The misactor extracts an EV's provisioning credentials.
Alternative Flow: The misactor extracts an EV's contract credentials.
Trigger: Controlled by misactor; can always happen.
Preconditions: For the basic flow: The misactor has access to the EV.
For the alternative flow: The misactor has access to the EV and the EV has contract credentials installed.

ST_{14} *Information disclosure at the EV's processes:* Similar to ST_9, corrupting an EV's processes (cf. ST_5) would allow the adversary to extract all of its data (same data as in ST_{13}). Additionally, based on ST_2, the adversary could also extract any data that is received or can be requested over ISO 15118, i.e., privacy-sensitive information in the form of charge profiles, -schedules, and -tariffs. Additionally, side channel attacks (e.g., power-trace- or timing-based) could be used in order to extract an EV's private keys during their usage.

Assets, Stakeholders, and Threats: The confidentiality of the EV's data stores is threatened, including security-relevant data.

[3] For example, Hubject lists their OCSP server as `http://ocsp.hubject.com:8080` [85], indicating that OCSP traffic uses no transport security.

Primary Misactor: Local Adversary
Basic Flow: The misactor tampers with an EV's processes, allowing them to extract an EV's provisioning credentials.
Alternative Flow 1: The misactor tampers with an EV's processes, allowing them to extract an EV's contract credentials.
Alternative Flow 2: The misactor spoofs an EV to a CS, allowing them to receive transient data over ISO 15118.
Alternative Flow 3: The misactor extracts an EV's private provisioning key via side channel attacks.
Alternative Flow 4: The misactor extracts an EV's private contract key via side channel attacks.
Trigger: Controlled by misactor; can always happen.
Preconditions: For the basic flow: ST_5 (basic flow).
For alternative flow 1: ST_5 (basic flow) and the EV has contract credentials installed.
For alternative flow 2: ST_2 (basic flow or alternative flow 1).
For alternative flow 3: The misactor has access to the EV.
For alternative flow 4: The misactor has access to the EV and the EV has contract credentials installed.

ST_{15} *Information disclosure to the backend:* Any personal data of an EV user that is sent to a backend system, while not being required for the correct operation of the system, is considered a privacy threat due to the potential misuse of this data by a Curious Operator (e.g., de-anonymizing a user based on collected location data; cf. Section 2.1.3).

Assets, Stakeholders, and Threats: The confidentiality of the EV user's personal data is threatened.
Primary Misactor: Curious Operator
Basic Flow: The misactor gains access to personal data, which is not required for the correct operation of their system.
Trigger: Controlled by misactor; can always happen.
Preconditions: Insufficient data minimization.

Denial of Service Against a Data Store, Data Flow, or Process: Denial of service attacks are out of scope.

Elevation of Privilege of Process: As listed in Section 5.1, backend processes are assumed to be secure and run-time attacks are out of scope. Hence, the only elevation of privilege threats result from spoofing as follows:

ST_{16} *Elevation of privilege at a CS:* Based on ST_2 or ST_3 an adversary could receive the privileges of the respective victim at a CS.

Assets, Stakeholders, and Threats: The authorization of EV users is threatened.
Primary Misactor: Local Adversary
Basic Flow: The misactor spoofs an EV, which results in the misactor receiving the authorization status of the corresponding EV user.
Alternative Flow: The misactor spoofs an EV user at a CS, which results in the misactor receiving the authorization status of this EV user.
Trigger: Controlled by misactor; can always happen.
Preconditions: For the basic flow: ST_2 (any flow).
For the alternative flow: ST_3 (any flow).

ST_{17} *Elevation of privilege at a CSO:* Based on ST_1 an adversary could receive the privileges of a CS at the CSO.

Assets, Stakeholders, and Threats: The authorization of a CS is threatened.
Primary Misactor: Local Adversary
Basic Flow: The misactor spoofs a CS, which results in the misactor receiving the authorization of this CS.
Trigger: Controlled by misactor; can always happen.
Preconditions: ST_1 (any flow).

5.2.2. Privacy Threats

This section describes the existing privacy threats (PT_x) in the EV charging infrastructure, resulting from our LINDDUN-based threat analysis. Threats are documented in the form of misuse cases as recommended by LINDDUN (cf. Section 2.3.2). Threats to soft privacy, i.e., the content unawareness and policy/consent non-compliance threats, are not addressed since soft privacy is not in line with the goals of this thesis (cf. Section 4.3).

Linkability of an External Entity, Data Store, Data Flow, or Process: As listed in Section 5.1, we only consider linkability threats at the application layer, i.e., linkability at lower layers (e.g., by IP address) is out-of-scope. Additionally, we assume that live CS status information is publicly accessible and updated in near-real time. Hence, we consider the following threats:

PT_1 *Linkability of multiple EV user authorizations:* Since ID Tokens are static, a Network Hacker that observes the local EIM-based authorization channel (cf. ST_{11}) can link EV user authorizations across all observed charging sessions/locations. Similarly, based on ST_8, ST_9, or ST_{10}, an adversary could link authorizations of the same EV user across all affected CSs, regardless of the authorization mechanism. Additionally, any Curious Operator (CSO, CCH, or eMSP) can link EV user authorizations across different charging sessions/locations as they all receive the corresponding ID Tokens (assuming no whitelist-based authentication was used; cf. ST_{15}). Linking EV users across charging sessions/locations enables an adversary to build detailed movement profiles for the affected users with the potential threat of re-identification (cf. Section 2.1.3).

Assets, Stakeholders, and Threats: The pseudonyms of EV users can be linked across different charging sessions/locations. A misactor is able to build movement profiles if linking across locations is possible, which may be identifying (cf. Section 2.1.3).
Primary Misactor: Network Hacker or Local Adversary or Curious Operator
Basic Flow: The misactor (Network Hacker or Local Adversary) obtains the authorization data (ID Tokens) of EV users. The misactor can link the authorization data of the same user based on static pseudonyms (RFID UID or eMAID).
Alternative Flow 1: The misactor (Local Adversary) spoofs a CS to EVs. The EV authenticates itself using PnC credentials. The misactor can link the authentication data of the same user, i.e., the credentials, since linkable signatures are used.
Alternative Flow 2: The misactor (Curious Operator) receives EV user authorizations during normal operation. The misactor can link the authorization data of the same user based on static pseudonyms (RFID UID or eMAID).
Trigger: Controlled by misactor; can always happen.
Preconditions: For basic flow: Insufficient de-identification and the misactor obtains authorization data, i.e., ST_8 (basic flow or alternative flow 1) or ST_9 (basic flow or alternative flow 1 or alternative flow 3 or alternative flow 4 or alternative flow 5) or ST_{10} (any flow) or ST_{11} (any flow).

For alternative flow 1: Insufficient de-identification, the misactor can link multiple signatures from the same private key, the misactor receives transient data from an EV over ISO 15118, i.e., ST_9 (alternative flow 4).
For alternative flow 2: Insufficient de-identification and the misactor obtains authorization data, i.e., ST_{15} (basic flow).

PT_2 *Linkability of EV users and groups:* Since ID Token authorization data may include a group ID, any adversary with access to this information can link different users in the same group and possibly infer social relations between the users (e.g., that they are family members or co-workers). Any Curious Operator (cf. ST_{15}) as well as adversaries based on ST_8 and ST_9 have access to this group information.

Assets, Stakeholders, and Threats: The ID Token data of different EV users can be linked by the group ID. A misactor may infer social relations.
Primary Misactor: Local Adversary or Curious Operator
Basic Flow: The misactor (Local Adversary) obtains ID Token data from a CS' authorization whitelist/-cache. The misactor can link the ID Token data of different users based on the group ID.
Alternative Flow 1: The misactor (Local Adversary) spoofs a CS towards a CSO. The misactor uses the spoofed CS to receive an authorization whitelist (contains ID Token data with group IDs) from the CSO or to request the group ID of specific ID Tokens (via separate authorization requests; the response contains the respective group ID) from the CSO. The misactor can link the ID Token data of different users based on the group ID.
Alternative Flow 2: The misactor (Curious Operator) receives ID Token data during normal operation. The misactor can link the ID Token data of different users based on the group ID.
Trigger: Controlled by misactor; can always happen.
Preconditions: For basic flow: The ID Token data contains a group ID and the misactor obtains ID Token data, i.e., ST_8 (basic flow or alternative flow 1) or ST_9 (basic flow or alternative flow 1).
For alternative flow 1: The ID Token data contains a group ID and the misactor obtains ID Token data, i.e., ST_9 (alternative flow 3).
For alternative flow 2: The ID Token data contains a group ID and the misactor obtains ID Token data, i.e., ST_{15} (basic flow).

PT_3 *Linkability of EV users and EVs:* Based on ST_9, an adversary could link the ID Tokens of users to EVCC IDs. Additionally, during the process of user credential provisioning, any Curious Operator (cf. ST_{15}) as well as an adversary based on ST_9 can link an EV user's eMAID (from the contract certificate) to the corresponding EVCC's PCID (from the provisioning certificate). Note that a spoofed CS (ST_1) could always indicate that a contract certificate is invalid in order to force a new credential provisioning process and this way guarantee that eMAID and PCID can be linked. Similar to the linkability of users to groups (cf. PT_2), this threat could be used to infer social relation between users (RFID UIDs and/or eMAIDs) sharing the same EV (PCID).[4] Furthermore, this threat allows the likability of multiple EV user authorizations (cf. PT_1) across different contracts.

Assets, Stakeholders, and Threats: The ID Token(s) of an EV user (or of multiple users) can be linked to the same EV. A misactor may infer social relations. A misactor may link different ID Tokens (RFID UIDs and/or eMAIDs) of the same user via the EV.

[4]The simultaneous use of multiple different contract credentials on the same EV is only supported with ISO 15118-20 [98]. In ISO 15118-2 [97], linking multiple eMAIDs to the same PCID would still be possible if different, consecutive contracts are associated withe the same PCID (e.g., if a user switches eMSPs or the EV is sold).

Primary Misactor: Local Adversary or Curious Operator

Basic Flow: The misactor (Local Adversary) spoofs a CS to an EV. The EV sends its EVCC ID to the CS and the EV (user) authorizes a charging session at the CS. The misactor can link the EVCC ID to the ID Token used for authorization.

Alternative Flow 1: The misactor (Curious Operator) receives an EV's contract certificate installation request and response during normal operation. The misactor can link PCID and eMAID.

Alternative Flow 2: The misactor (Local Adversary) spoofs a CS to an EV and to the CSO. The CS indicates that the EV's contract certificate is invalid (e.g., by setting the timestamp in the `SessionSetupRes` message to 2 years[5] in the future). The EV sends a certificate installation request. The CS forwards the request to the backend and receives a response. The misactor can link PCID and eMAID.

Trigger: Controlled by misactor; can always happen.

Preconditions: For the basic flow: Insufficient de-identification and the misactor receives transient data from an EV over ISO 15118, i.e., ST_9 (alternative flow 4).
For alternative flow 1: Insufficient de-identification and the misactor receives certificate installation requests and responses, i.e., ST_{15} (basic flow).
For alternative flow 2: Insufficient de-identification and the misactor can spoof a CS to an EV and to the CSO in order to receive transient data, i.e., ST_9 (alternative flow 3 and alternative flow 4).

PT_4 *Linkability of multiple charge sessions:* There are several ways in which multiple charge sessions could be linked. First, an adversary could link sessions of the same user based on the user's authorization (cf. PT_1). Second, a Curious Operator could link sessions of the same user based on the user's ID Token in transaction- and CDR messages (even if no authorization request was received, i.e., if whitelist-based authorization was used). Third, based on ST_9, an adversary could link sessions to the same EV based on the EVCC ID from the V2G session setup. Finally, charge sessions could be linked based on session-specific charging parameters. Any Curious Operator is informed about a charge session's start-/end time, consumption, location, cost, chosen tariffs, and the user's charge preferences. Due to the abundance and high definition of this data, it may be used to link sessions of the same user even without direct identifiers like the ID Token (cf. Section 2.1.3, in particular [184, 185]).

Assets, Stakeholders, and Threats: The charge session data of EV users can be linked. A misactor may infer movement-/behavioral patterns.

Primary Misactor: Local Adversary or Curious Operator

Basic Flow: The misactor (Local Adversary) spoofs a CS to an EV and receives charge session data along with the user's authorization data. The session data of multiple charge sessions at spoofed CSs can be linked based on the user's authorization data.

Alternative Flow 1: The misactor (Curious Operator) receives charge session data along with the user's ID Token during normal operation. The session data of multiple charge sessions can be linked based on the user's ID Token.

Alternative Flow 2: The misactor (Local Adversary) spoofs a CS to an EV and receives charge session data along with the EVCC ID. The session data of multiple charge sessions at spoofed CSs can be linked based on the EVCC ID.

Alternative Flow 3: The misactor (Curious Operator) receives charge session data during normal operation. The session data of multiple charge sessions may be linked without additional data

[5] ISO 15118 suggests that EVs use the CS's timestamp in order to determine if a new contract certificate is required (cf. [97], Section 8.4.3.2.2). Contract certificates have a maximum validity of 2 years (cf. [97], Table F.4).

(cf. Section 2.1.3, in particular [184, 185]).[6]

Trigger: Controlled by misactor; can always happen.

Preconditions: For the basic flow: Insufficient de-identification and ST_9 (alternative flow 4).
For alternative flow 1: Insufficient de-identification and the misactor receives charge session data along with the user's ID Token, i.e., ST_{15} (basic flow).
For alternative flow 2: Insufficient de-identification and ST_9 (alternative flow 4).
For alternative flow 3: Insufficient de-identification and the misactor receives charge session data, i.e., ST_{15} (basic flow).

PT_5 *Linkability of EV users and charge sessions:* Based on ST_9, an adversary can link a user's ID Token to a charge session at the affected CS. Additionally, any Curious Operator can link users' pseudonyms and charge sessions since transaction messages and CDRs contain the user's ID Token as well as an ID of the session.

Assets, Stakeholders, and Threats: The charge session data can be linked to a pseudonym of the EV user. A misactor may infer movement-/behavioral patterns.

Primary Misactor: Local Adversary or Curious Operator

Basic Flow: The misactor (Local Adversary) spoofs a CS to an EV and receives charge session data along with the user's authorization data. The session data at spoofed CSs can be linked to the user's ID Token.

Alternative Flow: The misactor (Curious Operator) receives charge session data along with the user's ID Token during normal operation. The session data can be linked to the user's ID Token.

Trigger: Controlled by misactor; can always happen.

Preconditions: For the basic flow: Insufficient de-identification and ST_9 (alternative flow 4).
For the alternative flow: Insufficient de-identification and the misactor receives charge session data along with the user's ID Token, i.e., ST_{15} (basic flow).

PT_6 *Linkability of charge sessions and locations/CSs:* Based on ST_9, an adversary can link a charge session's data to the spoofed CS. Additionally, any Curious Operator can link sessions and locations since transaction messages and CDRs contain a session ID as well as an ID of the location/EVSE. Furthermore, since the live CS status information (whether a CS is available or occupied) is updated in (near) real-time, any adversary with knowledge about the start and/or end time of charging sessions and who receives session data (i.e., any Curious Operator) can link locations/CSs to the session data.

Assets, Stakeholders, and Threats: The charge session data can be linked to a location/CS. A misactor may infer movement-/behavioral patterns (cf. [184, 185]).

Primary Misactor: Local Adversary or Curious Operator

Basic Flow: The misactor (Local Adversary) spoofs a CS to an EV and receives charge session data. The session data at spoofed CSs can be linked to the CS.

Alternative Flow 1: The misactor (Curious Operator) receives charge session data during normal operation. The session data identifies the location/CS.

Alternative Flow 2: The misactor (Curious Operator) receives charge session data during normal operation. The session data may be linked to the location/CS based on live CS status information via the respective start and/or end time.

[6] As a Local Adversary who spoofs CSs to EVs only receives a very limited amount of session data, i.e., only the session data of spoofed CSs, the possibility of this misactor being able to link multiple charge sessions without additional data is not considered to be likely given that the results of [184, 185] consider the role of an aggregator, i.e., a central actor that receives the session data of many different locations (e.g., a CSO).

Trigger: Controlled by misactor; can always happen.
Preconditions: For the basic flow: ST_9 (alternative flow 4).
For alternative flow 1: Insufficient data minimization and the misactor receives charge session data, i.e., ST_{15} (basic flow).
For alternative flow 2: Insufficient data minimization, sufficiently timely live CS status updates to enable time-based linking, and the misactor receives charge session data, i.e., ST_{15} (basic flow).

PT_7 *Linkability of EV users and locations/CSs:* There are several ways in which a user's pseudonym could be linked to a location/CS that they charged at. First, based on ST_8, ST_9, ST_{10}, or ST_{11}, an adversary can link a user's ID Token to the affected CS. Second, any Curious Operator can link users and locations since authorization requests, reservation requests, transaction messages, and CDRs contain the user's ID Token as well as an ID of the location/EVSE. Third, as long as a Curious Operator knows the start and/or end time of a charging session and can link a user to the session (cf. PT_5), they can link the user to the location of the session based on live CS data (cf. PT_6; even if the session information does not directly identify the location). Finally, a Network Hacker can derive the time of a user's PnC-based authentication via the corresponding OCSP request (cf. ST_{12}) and link this information, i.e., a pseudonym of the EV user (the OCSP request uniquely identifies the user's contract certificate), to the charge location based on publicly available live CS stats data.

Assets, Stakeholders, and Threats: An EV user's pseudonyms can be linked to a location/CS. A misactor is able to build movement profiles, which is likely identifying (cf. Section 2.1.3).
Primary Misactor: Network Hacker or Local Adversary or Curious Operator
Basic Flow: The misactor (Local Adversary) obtains the authorization data (ID Tokens) of EV users from a CS' authorization cache. The misactor can link the authorization data to the affected location/CS.
Alternative Flow 1: The misactor (Local Adversary) spoofs a CS to an EV and obtains an EV user's pseudonyms during the authorization process. The misactor can link the pseudonyms to the affected location/CS.
Alternative Flow 2: The misactor (Network Hacker) obtains an EV user's RFID UID during the local EIM-based authorization process. The misactor can link the user's pseudonym to the affected location/CS.
Alternative Flow 3: The misactor (Curious Operator) receives charge session data during normal operation. The misactor can link the user's pseudonyms to the charge location/CS (both contained in the session data).
Alternative Flow 4: The misactor (Curious Operator) receives charge session data during normal operation. The session data may be linked to the location/CS based on live CS status information via the respective start and/or end time. The misactor can link the user's pseudonyms to the charge location/CS (even if the location is not contained in the session data).
Alternative Flow 5: The misactor (Network Hacker) derives the start time of a charging session based on the corresponding OCSP request. The start time may be linked to the location/CS based on publicly available live CS status information. The misactor can link the OCSP request (contains a user pseudonym as it uniquely identifies the contract certificate) to the charge location/CS.
Trigger: Controlled by misactor; can always happen.
Preconditions: For basic flow: Insufficient de-identification and the misactor can obtain authorization data from a CS' authorization cache, i.e., ST_8 (alternative flow 1) or ST_9 (alternative flow 1) or ST_{10} (alternative flow).

For alternative flow 1: Insufficient de-identification and the misactor can spoof a CS to EVs in order to receive transient ISO 15118 data, i.e., ST_9 (alternative flow 4).
For alternative flow 2: Insufficient de-identification and the misactor can receive the local EIM-based authorization data, i.e., ST_{11} (basic flow).
For alternative flow 3: Insufficient de-identification and the misactor obtains session data, i.e., ST_{15} (basic flow).
For alternative flow 4: Insufficient de-identification, sufficiently timely live CS status updates to enable time-based linking, and the misactor receives charge session data, i.e., ST_{15} (basic flow).
For alternative flow 5: Insufficient de-identification, sufficiently timely and publicly available live CS status updates to enable time-based linking, and the misactor can eavesdrop on OCSP traffic, i.e., ST_{12} (basic flow).

Identifiability of an External Entity, Data Store, Data Flow, or Process: As mentioned in the data minimization goals (cf. Section 4.3), we assume that eMSPs need to identify their customers for the fulfillment of their contract with the EV users (cf. Section 4.3), i.e., we do not consider it as a threat. Furthermore, as listed in Section 5.1, we assume that there is a sufficiently high risk of a user's mobility traces being linkable to their identity (cf. Section 5.1).

PT_8 *Identifiability of EV users based on movement profiles:* An EV user could be identified based on a movement profile of their charge sessions. Due to the uniqueness of movement profiles (cf. Section 2.1.3), they can likely be linked to public information in order to identify the corresponding EV user.

Assets, Stakeholders, and Threats: The link between an EV user's movement profile and identity is revealed. A misactor may infer additional personal information about the user.
Primary Misactor: Curious Operator or Local Adversary or Network Hacker
Basic Flow: The misactor (Curious Operator) receives enough authorization information during normal operation to build a movement profile. The misactor links the movement profile to an identity based on public information.
Alternative Flow 1: The misactor (Curious Operator) receives enough session information during normal operation to build a movement profile. The misactor links the movement profile to an identity based on public information.
Alternative Flow 2: The misactor (Local Adversary) receives enough authorization information by spoofing CSs to build a movement profile. The misactor links the movement profile to an identity based on public information.
Alternative Flow 3: The misactor (Local Adversary) receives enough session information by spoofing CSs to build a movement profile. The misactor links the movement profile to an identity based on public information.
Alternative Flow 4: The misactor (Local Adversary) receives enough information from CSs' authorization caches to build a movement profile. The misactor links the movement profile to an identity based on public information.
Alternative Flow 5: The misactor (Network Hacker) receives enough information by eavesdropping on local EIM authorizations to build a movement profile. The misactor links the movement profile to an identity based on public information.
Trigger: Controlled by misactor; can always happen.
Preconditions: For any flow: Public information is available that allows linking movement profiles to identities.

For the basic flow: The misactor is able to build movement profiles by linking multiple user authorizations and linking users to locations, i.e., PT_1 (alternative flow 2) and PT_7 (alternative flow 3 or alternative flow 4).
For alternative flow 1: The misactor is able to build movement profiles by linking multiple sessions and linking sessions to locations, i.e., PT_4 (alternative flow 1 or alternative flow 3) and PT_6 (alternative flow 1 or alternative flow 2).
For alternative flow 2: The misactor is able to build movement profiles by linking multiple user authorizations and linking users to locations, i.e., PT_1 (basic flow or alternative flow 1) and PT_7 (alternative flow 1).
For alternative flow 3: The misactor is able to build movement profiles by linking multiple sessions and linking sessions to locations, i.e., PT_4 (basic flow or alternative flow 2) and PT_6 (basic flow).
For alternative flow 4: The misactor is able to build movement profiles by linking multiple user authorizations and linking users to locations, i.e., PT_1 (basic flow) and PT_7 (basic flow).
For alternative flow 5: The misactor is able to build movement profiles by linking multiple user authorizations and linking users to locations, i.e., PT_1 (basic flow) and PT_7 (alternative flow 2).

Non-Repudiation of a Data Store, Data Flow, or Process: Since we are concerned with billing(-related) processes, we do not consider non-repudiation as a threat but instead as a security goal.

Detectability of a Data Store, Data Flow, or Process: Relevant detectability threats are already considered as part of previous threats, i.e., the possibility to detect if an ID Token is present in a CS' authorization cache (which we consider to mainly be an information disclosure threat since it also reveals the ID Token; cf. ST_{10}), the possibility to detect a charge authorization based on OCSP messages (which we consider to mainly be an information disclosure threat since it also reveals a pseudonym of the user; cf. ST_{12}), and the possibility to detect the start of a charging session based on live CS status information (which we only consider to be a threat in combination with the linkability to other data; cf. PT_6 and PT_7). Hence, we do not consider further detectability threats.

Disclosure of Information at a Data Store, Data Flow, or Process: Disclosure of information threats are covered by STRIDE.

Content Unawareness of an External Entity: Content unawareness threats are out of scope.

Policy/Consent Non-Compliance of a Data Store, Data Flow, or Process: Policy and consent non-compliance threats are out of scope.

5.2.3. Summary

The STRIDE- and LINDDUN-based threat analysis has identified a significant amount of existing security- and privacy threats in EV charging protocols. In particular, 17 security threats and 8 privacy threats (with multiple potential flows for most threats) were identified under consideration of the adversaries described in Section 5.1. In order to address these threats, Section 6 defines a list of requirements which a potential solution should meet such that a secure and privacy-preserving charging and billing of EVs can be achieved.

6. Requirement Analysis

The following sections list the security- and privacy requirements (Section 6.1 and Section 6.2) that a solution must fulfill in order to address the threats of Section 5.2 considering the goals of Section 4.3. In general, an EV user's payment credentials must be protected during their storage and usage. Additionally, the existing authentication and authorization methods of the current EV charging protocols must be adjusted to make them secure and protect the user's privacy under the considered adversary model (cf. Section 5.1). Furthermore, the security and privacy protections should hold even in the face of an adversarial CS in order to avoid expensive hardware upgrades to already deployed CSs.

Moreover, no backend actor should be able to link a long-term pseudonym of the EV user (or identifier of the EV) to the user's charging locations in order to prevent the possibility of building mobility traces. In particular, only the EV user's eMSP should be able to link different charging sessions of this user, i.e., by linking them to the user's real identity, for billing purposes. Additionally, a CSO must be able to identify the eMSP of an EV user charging at one of their CSs and must know the corresponding consumption for billing purposes. Only a CSO should be able to link a transaction to a location, i.e., know the current and precise charging parameters of specific CSs, in order to manage their CSs. While an eMSP must be informed of live CS status data in order to provide reservation services, this charge location data should not be linkable to session-specific charge parameters in order to lower the privacy risk resulting from the analysis in [184, 185].

In order to guarantee the usability of the solution and considering the goals of Section 4.3, we also address relevant functional requirements for a solution. Finally, Section 6.4 provides a mapping between threats and requirements in order to show how the presented requirements are necessary and sufficient to address the identified threats, given the considered adversary model (cf. Section 5.1) and goals (cf. Section 4.3).

6.1. Security Requirements

In order to address the security threats from Section 5.2.1, a solution must meet the following security requirements (SR_x):

SR_1 *Secure EV Credentials:* An EV provisioning- and contract credentials should be protected during their storage and usage. This requirement addresses the credential information disclosure threats at the EV as well as the resulting spoofing, repudiation, information disclosure, and elevation of privilege threats.

SR_2 *Secure Charge Authorization:* The charge authorization mechanism should be secure under the assumption of an adversary controlled CS. It should not be possible to guess valid authorization values. It should not be possible to authorize a charging session at any CS based on a disclosed CS authorization whitelist/-cache. It should not be possible to authorize a charging session at any CS by replaying previously transmitted authorization data (obtained by the adversary via a spoofed CS or information disclosure at the local EIM data flow). The authorization status of EV users should be revocable.

This requirement addresses EV user spoofing threats, authorization-related repudiation threats, and revocation-related information disclosure threats.

SR_3 *EV Integrity:* The access to an EV's private PnC keys should be prohibited if the EV's firmware was manipulated [66]. This requirement addresses the adverse effects of a successful tampering with an EV, i.e., the resulting spoofing and information disclosure threats.

SR_4 *End-to-End Protection of Charge Session Data:* In the PnC case, billing-relevant charge session data should be authenticity protected by the EV for the corresponding eMSP. The eMSP should be assured of the integrity of the EV. This requirement addresses the threat of repudiation of charge session data in the presence of a malicious CS. Since for the EIM case no additional requirements should be put on the EV (cf. Section 4.3), this requirement only applies to the PnC case.

6.2. Privacy Requirements

In order to address the privacy threats from Section 5.2.2, a solution must meet the following privacy requirements (PR_x):

PR_1 *Data Minimization:* Backend actors should only receive the minimum required personal data (cf. Section 4.3). This requirement addresses the general information disclosure threat to the backend and partially addresses the resulting linkability and identifiability threats.

PR_2 *Unlinkable EV User Authorizations:* Multiple authorizations of an EV user should not be linkable based on the authorization data by any actor but the eMSP. Multiple EV users should not be linkable to the same group by any actor but the eMSP.

PR_3 *Unlinkable PnC Authentications:* The PnC authentication mechanism should not be linkable across charging sessions.

PR_4 *Unlinkability of EV Users and EVs:* An EV user or their pseudonyms should not be linkable to their EV or IDs of their EV by anyone but the eMSP.

PR_5 *Unlinkability of Charge Sessions:* Multiple charge session of the same EV user should not be linkable by anyone but the eMSP. Only the eMSP may link charge session data to an EV user. Any actor may link charge session data to a transaction pseudonym. Only the CSO may link charge session data to a location.

PR_6 *Unlinkability of EV Users and Locations:* No actor should be able to link an EV user or their pseudonyms to a location. Only the CSO and CS may link at most transaction pseudonyms to a charge location.

PR_7 *Anonymity of EV Users:* No actor but the eMSP may identify an EV user.

6.3. Functional Requirements

In order to address the goals from Section 4.3 and provide an adequate level of usability, a solution must meet the following functional requirements (FR_x):

FR_1 *Session Stop Restrictions:* A solution should be able to offer the same session stop restrictions as the current mechanisms, i.e., for local EIM-based authorization, only the same token that was used to start a session or a token in the same group should be able to stop the session.

FR_2 *Support Existing Billing Relations:* A CSO should be able to bill the eMSP of a user. The eMSP should be able to bill the user.

FR_3 *Minimal Overhead:* The additional overhead of a solution should be as low as possible and cryptographic requirements for the local EIM-based charge authorization mechanism should be minimal.

FR_4 *Support for Offline Operation:* CSs should be able to operate in semi-online mode, i.e., CSs should be able to operate mostly offline with periodic synchronizations wit their CSO.

FR_5 *Support for Reservations:* The solution should support the ability of EV users to reserve CSs via their eMSP.

FR_6 *Support for Authorization Mechanisms:* The solution should support existing authorization methods (cf. Section 4.2.2), i.e., local EIM-based authorization, local PnC-based authorization, and remote authorization. EIM-based authorization should not put additional requirements on the EV (e.g., the EV should not have to authenticate itself during EIM-based authorization) in order to keep the changes to existing methods as well the requirements for additional system capabilities to a minimum.

FR_7 *Support Contract Credential Installation:* The solution should support the automatic installation of contract credentials. The usability of contract credential installation should not be impacted compared to the existing mechanism, i.e., the user registers their EV (via PCID) at the eMSP and installation is automatic during the first charging session.

FR_8 *Support Credential Pools:* The solution should support credential pools for the temporal decoupling of credential installation-related processes (cf. Section 4.2.5).

FR_9 *Minimal Changes:* It should be possible to integrate the solution into the existing charging protocols with only minimal changes in order to keep the cost of implementation at an appropriate level.

6.4. Mapping of Requirements to Threats

Table 6.1 shows a mapping between the defined security- and privacy requirements and threats (with flows and preconditions). An “X” is used to mark which threat flows (*Basic Flow* (*BF*) and/or *Alternative Flow* (*AF*)) a requirement addresses and “(X)” indicates that the threat flow is addressed indirectly via its preconditions.

The risk of some threats is accepted (marked with “A”; or “(A)” if it is accepted via its preconditions). In particular, it is accepted that the Local Adversary can always tamper with the EV, i.e., the EVCC is not required to be tamper-proof, however the adverse effects of a successful tampering are addressed. Additionally, considering the Local Adversary and the goal of not requiring trusted hardware on the CS (cf. Section 4.3), we accept the risks of tampering and information disclosure at the CS’ data stores as well as resulting risks that do not affect EV users (e.g., the resulting spoofing of a CS is accepted but using obtained ID Tokens to spoof an EV user is not). However, we accept that a Local Adversary who spoofs a CS (which is possible as the Local Adversary can, e.g., tamper with the CS’ firmware data store) can link charge session data at the spoofed CS to the spoofed CS (i.e., PT_6 *BF*) as the resulting privacy harm is very limited. Similarly, based on the goal of not requiring additional transport security for the local EIM-based authorization mechanism, we

accept the information disclosure risks regarding this data flow but address the resulting risks to EV users (e.g., the resulting linkability of eavesdropped traffic and the resulting spoofing of EV users).

Table 6.1.: Mapping of Requirements to Threats

Threats		Accepted	Requirements										
Flows	Preconditions		SR_1	SR_2	SR_3	SR_4	PR_1	PR_2	PR_3	PR_4	PR_5	PR_6	PR_7
ST_1 Spoofing of a CS:													
BF	ST_8 (*AF2*) or ST_9 (*AF2* or *AF6*)	(A)											
AF	ST_4 (*BF*)	(A)											
ST_2 Spoofing of an EV:													
BF	ST_2 (*AF2*) or ST_{13} (*AF*) or ST_{14} (*AF1* or *AF4*)	-	(X)		(X)								
AF1	ST_5 (*BF*)	-			X								
AF2	ST_{13} (*BF*) or ST_{14} (*BF* or *AF3*)	-	(X)		(X)								
ST_3 Spoofing of a local EV user:													
BF	ST_8 (*BF* or *AF1*) or ST_9 (*BF* or *AF1*)	-		X									
AF1	ST_9 (*AF5*)	-		X									
AF2	ST_9 (*AF3*)	-		X									
AF3	ST_{11} (any flow)	-		X									
AF4	ST_{10} (any flow)	-		(X)									
ST_4 Tampering with the CS:													
BF	-	A											
AF	-	A											
ST_5 Tampering with the EV:													
BF	-	A											
ST_6 Repudiation of charge authorization:													
BF	ST_2 (*BF* or *AF1*)	-	(X)		(X)								
AF1	ST_3 (any flow)	-		(X)									
AF2	ST_1 (any flow)	-		X									
ST_7 Repudiation of charge session data:													
BF	ST_1 (any flow)	-				X							
AF	ST_6 (any flow)	-	(X)	(X)	(X)								
ST_8 Information disclosure of the CS' data stores:													
BF	-	A											
AF1	-	A											
AF2	-	A											

BF = Basic Flow, *AFi = Alternative Flow i*, X = addressed, (X) = addressed via preconditions

A = accepted, (A) = accepted via preconditions

Table 6.1.: Mapping of Requirements to Threats *(cont.)*

Threats		Accepted	Requirements										
Flows	Preconditions		SR_1	SR_2	SR_3	SR_4	PR_1	PR_2	PR_3	PR_4	PR_5	PR_6	PR_7
ST_9 Information disclosure of the CS' processes:													
BF	*ST_4 (BF)*	(A)											
AF1	*ST_4 (BF)*	(A)											
AF2	*ST_4 (BF)*	(A)											
AF3	*ST_1* (any flow)	(A)											
AF4	*ST_1* (any flow)	(A)											
AF5	*ST_1 (AF)*	(A)											
AF6	-	A											
ST_{10} Information disclosure at the local EIM authorization process:													
BF	-	-		X									
AF	-	-		X									
ST_{11} Information disclosure of the local EIM authorization data flow:													
BF	-	A											
AF	-	A											
ST_{12} Information disclosure of the OCSP data flow:													
BF	-	-		X									
ST_{13} Information disclosure at the EV's data stores:													
BF	-	-	X										
AF	-	-	X										
ST_{14} Information disclosure at the EV's processes:													
BF	*ST_5 (BF)*	-	X		X								
AF1	*ST_5 (BF)*	-	X		X								
AF2	*ST_2 (BF* or *AF1)*	-	(X)		(X)								
AF3	-	-	X										
AF4	-	-	X										
ST_{15} Information disclosure to the backend:													
BF	-	-					X						
ST_{16} Elevation of privilege at a CS:													
BF	*ST_2* (any flow)	-	(X)		(X)								
AF	*ST_3* (any flow)	-		(X)									
ST_{17} Elevation of privilege at a CSO:													
BF	*ST_1* (any flow)	(A)											
PT_1 Linkability of multiple EV user authorizations:													
BF	*ST_8 (BF* or *AF1)* or *ST_9 (BF* or *AF1* or *AF3* or *AF4* or *AF5)* or *ST_{10}* (any flow) or *ST_{11}* (any flow)	-						X					
AF1	*ST_9 (AF4)*	-							X				

BF = Basic Flow, *AFi = Alternative Flow i*, X = addressed, (X) = addressed via preconditions
A = accepted, (A) = accepted via preconditions

Table 6.1.: Mapping of Requirements to Threats *(cont.)*

Threats		Accepted	Requirements										
Flows	Preconditions		SR_1	SR_2	SR_3	SR_4	PR_1	PR_2	PR_3	PR_4	PR_5	PR_6	PR_7
AF2	ST_{15} (*BF*)	-						X					
PT_2 *Linkability of EV users and groups:*													
BF	ST_8 (*BF* or *AF1*) or ST_9 (*BF* or *AF1*)	-						X					
AF1	ST_9 (*AF3*)	-						X					
AF2	ST_{15} (*BF*)	-						X					
PT_3 *Linkability of EV users and EVs:*													
BF	ST_9 (*AF4*)	-								X			
AF1	ST_{15} (*BF*)	-								X			
AF2	ST_9 (*AF3* and *AF4*)	-								X			
PT_4 *Linkability of multiple charge sessions:*													
BF	ST_9 (*AF4*)	-									X		
AF1	ST_{15} (*BF*)	-									X		
AF2	ST_9 (*AF4*)	-									X		
AF3	ST_{15} (*BF*)	-									X		
PT_5 *Linkability of EV users and charge sessions:*													
BF	ST_9 (*AF4*)	-									X		
AF	ST_{15} (*BF*)	-									X		
PT_6 *Linkability of charge sessions and locations/CSs:*													
BF	ST_9 (*AF4*)	(A)											
AF1	ST_{15} (*BF*)	-									X		
AF2	ST_{15} (*BF*)	-									X		
PT_7 *Linkability of EV users and locations/CSs:*													
BF	ST_8 (*AF1*) or ST_9 (*AF1*) or ST_{10} (*AF*)	-										X	
AF1	ST_9 (*AF4*)	-										X	
AF2	ST_{11} (*BF*)	-										X	
AF3	ST_{15} (*BF*)	-										X	
AF4	ST_{15} (*BF*)	-										X	
AF5	ST_{12} (*BF*)	-										X	
PT_8 *Identifiability of EV users based on movement profiles:*													
BF	PT_1 (*AF2*) and PT_7 (*AF3* or *AF4*)	-											X
AF1	PT_4 (*AF1* or *AF3*) and PT_6 (*AF1* or *AF2*)	-											X
AF2	PT_1 (*BF*) and PT_7 (*AF1*)	-											X

BF = Basic Flow, *AFi = Alternative Flow i*, X = addressed, (X) = addressed via preconditions
A = accepted, (A) = accepted via preconditions

Table 6.1.: Mapping of Requirements to Threats *(cont.)*

Threats		Accepted	Requirements										
Flows	Preconditions		SR_1	SR_2	SR_3	SR_4	PR_1	PR_2	PR_3	PR_4	PR_5	PR_6	PR_7
AF3	PT_4 (*BF* or *AF2*) and PT_6 (*BF*)	-											X
AF4	PT_1 (*BF*) and PT_7 (*BF*)	-											X
AF5	PT_1 (*BF*) and PT_7 (*AF2*)	-											X

BF = *Basic Flow*, *AFi* = *Alternative Flow i*, X = addressed, (X) = addressed via preconditions
A = accepted, (A) = accepted via preconditions

7. Privacy-Preserving Charge Authorization Concept

In order to satisfy the requirements from Section 6, we propose a method for privacy-preserving charge authorization as a replacement to the current ID Token based method (cf. Section 4). The charge authorization method supports both the EIM and the PnC case as well as both the offline and online authorization scenarios. Additionally, the authorization method enables the billing of eMSPs by CSOs and the user-specific charge session billing at eMSPs. Furthermore, the method protects the user's location privacy and prevent the linkability of different charging sessions by anyone except the user's eMSP.

The authorization method only relies on light-weight algorithms due to the low computational power of RFID EIM devices. In the PnC case, a higher level of security is provided by using DAA-based signatures for anonymous authentication (cf. Section 2.5.4) and a TPM (due to its applicability to the use case; cf. Section 2.5) to protect the EV's security-sensitive credentials and to provide trust in the EV's processes. Section 7.1 describes the authorization method based on the EIM case. Section 7.2 describes the integration of the DAA scheme from Section 2.5.4 for anonymous authentication into EV charging protocols and its extension with the presented authorization method for the privacy-preserving charge authorization and billing. Both methods are compatible with the general processes and communication flows of current EV charging protocols.

7.1. Privacy-Preserving EIM-Based Authorization

The general idea of the proposed method for local EIM-based authorization (e.g. RFID-based) in Section 7.1.2 is to use transaction pseudonyms (cf. Section 2.1.2) as ID Tokens. The pseudonyms are generated based on a shared secret between the EV user's EIM device and the eMSP such that the eMSP can re-identify the corresponding user, i.e., reverse the unlinkability of the pseudonyms, for the purpose of billing. The proposed method supports offline authorization at a CS as well as live authorization requests to the backend. A charging session can either be stopped by the same ID Token that was used to start it without requiring communication to the backend or by a different ID Token (e.g., to support ID Token groups; cf Section 4) with communication to the backend. Since the Authorization Cache method requires the linkability of two or more separate charge authorizations at the same CS, we do not support this feature in our solution. Section 7.1.3 further discusses consideration for remote EIM-based (e.g. smartphone-based) authorizations/-reservations.

7.1.1. Components

For the EIM-based authorization scenario, users are provisioned with EIM devices. An EIM device for local authorization (cf. Section 7.1.2) could be an RFID- or smart card, provided by the user's eMSP. As mentioned in Section 5.1, the data stores and processes of EIM devices are assumed to be secure. For remote authorization or reservations (cf. Section 7.1.3) the user's device could be a smartphone with an app that is provided by the eMSP. Since, for local authorization, the EIM device could have very limited computational resources (e.g., in

the case of an RFID card) we assume that the use of asymmetric cryptography is not possible in this scenario. This limitation does not apply to the remote EIM cases.

For local authorization, the EIM device (RFID card) stores a symmetric secret key ID_x, which uniquely identifies the user and is known to the eMSP. Additionally, the device is able to compute a cryptographically secure hash function mapping a bit string of arbitrary length to one of fixed length ($Hash(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^l$) as well as a secure keyed-hash $hmac_k(\cdot)$ indexed by key k (e.g., an HMAC, which requires two hashing and two XOR operations; cf. [118]). $Hash^n(\cdot)$ is used to represent recursive hashing, i.e., for example input m, $Hash^n(m) = Hash^{n-1}(Hash(m))$ and $Hash^1(m) = Hash(m)$. Furthermore, the EIM device maintains a counter i_x, which is used to generate different authorization values for every charge authorization, and the device stores an ID of the corresponding eMSP ID_{eMSP}.

7.1.2. Local EIM-Based Authorization Processes

The proposed method for local EIM-based authorization consists of three processes: (i) initialization of actors, (ii) charge authorization, and (iii) stopping a charge session. During the initialization, EIM devices (RFID cards) are prepared by the eMSP and delivered to their customers. Additionally, the ID Token whitelist is created and distributed to CSOs and CSs, whereby a CS' list only includes digests of the authorization values such that a leaked list (cf. ST_8) cannot directly be used for charge authorizations (cf. ST_3). For charge authorization, the EIM device sends two device-specific values to the CS: one identifies the current counter (without revealing it to anyone but the eMSP) and the other is used for authorization based on the device's secret key, counter, and a random nonce from the CS such that a captured response cannot be used on a different CS (cf. ST_{11} leading to ST_3). Note that the counter is different for every authorization in order to prevent linkability by anyone but the eMSP. Additionally, the device sends an identifier of the eMSP to the CS in order to support live authorization requests to the backend, the billing of charging sessions, and charging restrictions (e.g., location-, consumption-, or time-based) in accordance to the agreement between CSO and eMSP. The processes are detailed in the following sections. Additionally, the processes for updating an eMSP's whitelist and remote charge authorization/reservations are described.

Initialization

In order to prepare the privacy preserving EIM-based charge authorization, the eMSP changes the processes for initialization of EIM devices (RFID cards) and for the distribution of ID Token lists. The proposed method is shown in Fig. 7.1. EIM devices are initialized with a random secret key (ID_x for EV user X) and an ID of the eMSP ID_{eMSP}. The device's secret key ID_x is read-out by the eMSP some time before the device is provided to the user. Afterwards, the eMSP can prepare their ID Token list by computing their user's transaction pseudonyms $\langle id_x^i, auth_x^i \rangle$, whereby $id_x^i = Hash(hmac_{ID_x}(00||i))$ and $auth_x^i = hmac_{ID_x}(01||i)$, for each x in $[1,\ldots,m]$ and each i in $[1,\ldots,n]$, i.e., for m users with n authorizations each. New transaction pseudonyms need to be regularly generated and distributed (more details in a following section).

The ID Token list is distributed over the CCH to all applicable CSOs as usual with the exception that the eMSP shuffles the list before it is sent to the CCH such that entries cannot be linked based on their position (e.g., the first n entries all belong to the first user). Before the CSO forwards the list to a CS, they transform every entry's id_x^i into $idCS_x^i = Hash(id_x^i||CSID)$ and $auth_x^i$ into $authCS_x^i = Hash^3(auth_x^i||nonce_x^i)$, whereby $CSID$ uniquely identifies the target CS and $nonce_x^i$ is a random nonce and unique per pair of $auth_x^i$ and CS. Hence, all $idCS_x^i$ and $authCS_x^i$ values for a specific i and x are different on every CS such that the corruption

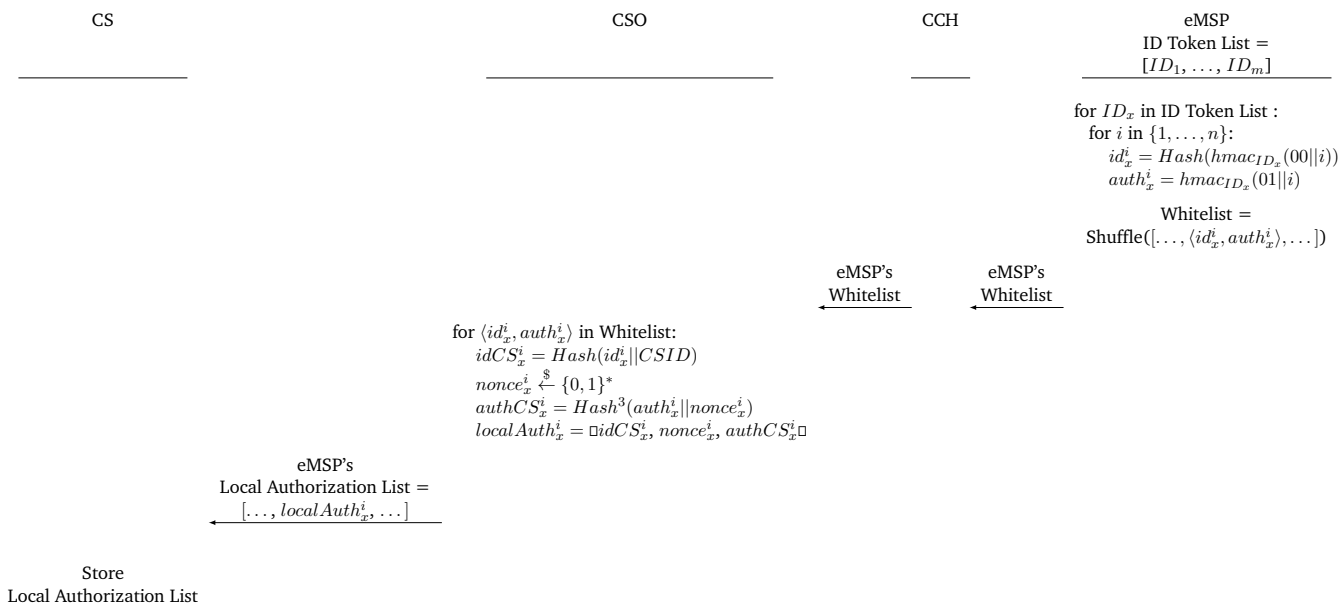


Figure 7.1.: Preparation for Offline Authorization

of one CS does not allow exploits on other CSs. The Local Authorization List that is sent from CSO to CS contains tuples in the form: $localAuth_x^i = \square idCS_x^i, nonce_x^i, authCS_x^i \square$.

Charge Authorization

Based on the Local Authorization List, a CS can authorize a charging session without requiring communication to the backend as shown in Fig. 7.2. After the communication between EIM device and CS is established, the EIM device increases its local counter i_x (cf. Section 7.1.1) and calculates $hmac_{ID_x}(00||i_x)$ with its secret ID_x. Afterwards, the EIM device sends $X1 = Hash(hmac_{ID_x}(00||i_x))$ and ID_{eMSP} back to the CS. The CS builds $idCS_x^i$ by hashing $X1$ from the EIM device's message concatenated with $CSID$ and uses this value to find the corresponding $localAuth_x^i$ entry in its Local Authorization List. If the entry is marked as used, the process is aborted. Otherwise the entry is marked as used and the process is continued. The $nonce_x^i$ value from $localAuth_x^i$ is sent to the EIM device, which calculates $temp_x^i = Hash(hmac_{ID_x}(01||i_x)||nonce_x^i)$ and stores this value together with the $nonce_x^i$. $X2 = Hash(temp_x^i)$ is sent to the CS as a response, which builds $Hash(X2)$, i.e., $Hash^3(hmac_{ID_x}(01||i_x)||nonce_x^i)$. If $Hash(X2)$ is equal to the $authCS_x^i$ value from the previously selected $localAuth_x^i$ entry in the Local Authorization List, then the entry is marked for deletion (it is still needed to stop the session) and the vehicle is cleared to charge. The CS should implement appropriate timeouts such that a blocked message (e.g., $X2$) cannot be replayed later for the same communication session.[1]

If $X1 = Hash(hmac_{ID_x}(00||i_x))$ from the EIM device's first message does not correspond to any $idCS_x^i$ value in the CS' Local Authorization List, then the CS has to send a live authorization request to its CSO. First, the CS generates a random nonce $nonce_x^i$ and sends it to the EIM device. The device responds with $X2 = Hash(temp_x^i)$ as usual. Afterwards, the CS forwards the user's authorization data (i.e., $X1$, ID_{eMSP}, $X2$, and $nonce_x^i$) to the CSO.

[1] Considering the physical overhead of EV charging (plugging in the EV etc.) and the small amount of time for generating/sending messages, such a timeout is argued to be realistic for the local EIM authorization use case.

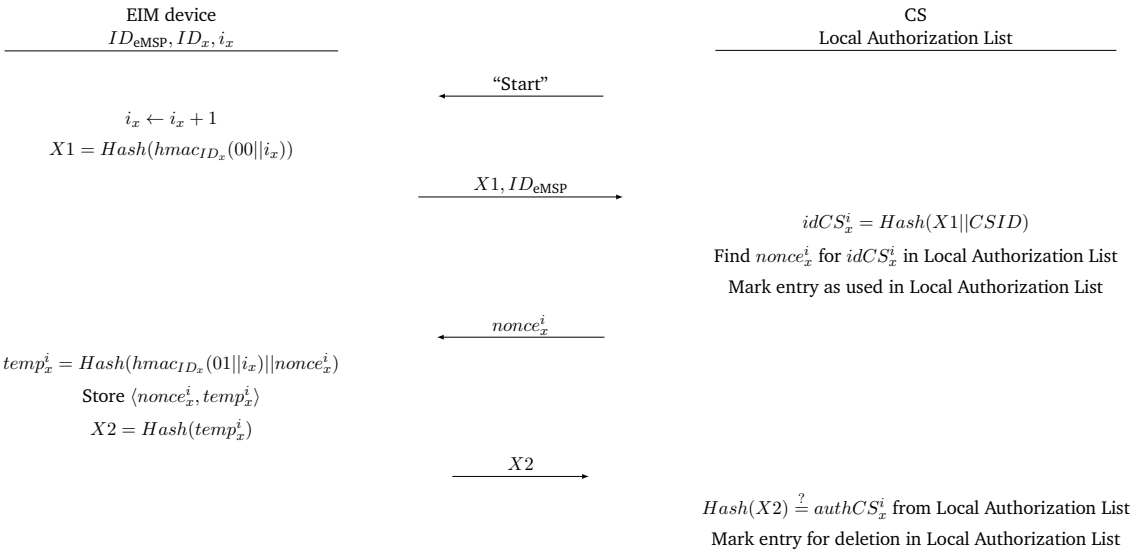


Figure 7.2.: Offline Charge Authorization

The CSO identifies the eMSP by ID_{eMSP} and checks if $X1$ corresponds to an id_x^i in this eMSP's whitelist. If id_x^i is present, then the CSO calculates $authCS_x^i = Hash^3(auth_x^i||nonce_x^i)$ using $nonce_x^i$ from the CS and $auth_x^i$ from the whitelist. Afterwards the CSO verifies that $Hash(X2)$ is equal to $authCS_x^i$. If it is equal, the CSO confirms the authorization request and otherwise denies it. If $X1$ is not present in this eMSP's whitelist (or no whitelist from the eMSP is available at the CSO), then the CSO forwards the request (with $X1$, ID_{eMSP}, $X2$, and $nonce_x^i$) to the CCH.

The CCH, similarly to the CSO, checks if $X1$ corresponds to an id_x^i in the eMSP's whitelist. If it is present, the CCH verifies the user's authorization ($Hash(X2) \stackrel{?}{=} authCS_x^i$) and responds to the CSO accordingly. If it is not present, the request is forwarded to the eMSP.

Finally, the eMSP checks if $X1$ belongs to one of its customers. If this is the case, they verify the user's authorization status, and if not, they deny the authorization request. We assume that eMSPs should always pre-generate more authorization values than they distribute in their whitelists such that they can authorize a user after all their current whitelist values are used up.

Stopping a Charge Session

The process for the EIM-based stopping of a charging session is shown in Fig. 7.3. The CS starts by sending the $nonce_x^i$ that was used during charge authorization to the EIM device. If $nonce_x^i$ is stored on the device it sends the corresponding $temp_x^i$ back to the CS and deletes $nonce_x^i$. The CS calculates $Hash^2(temp_x^i)$, verifies that the result is equal to the $auth_x^i$ that was used during charge authorization, and deletes the used entry from the list. This ensures that only the device that was used to start the session can also stop it.

If $nonce_x^i$ is not stored on the device (e.g., an error occurred or a different EIM device is used to stop the session), the device's i_x is incremented and new values $X1 = Hash(hmac_{ID_x}(00||i_x))$ and $X2 = Hash^2(hmac_{ID_x}(01||i)||nonce_x^i)$ are calculated and sent to the CS together with ID_{eMSP}. The CS sends a live stop session request, including the authorization data (i.e., $X1$, ID_{eMSP}, $X2$, and $nonce_x^i$) for both the start and the stop of the session, over the CSO and CCH to the eMSP. The eMSP can decide if the ID Token

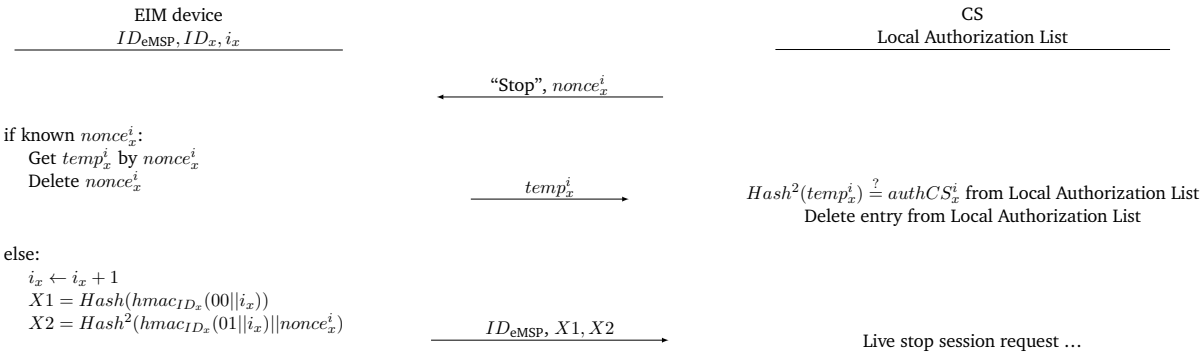


Figure 7.3.: Stopping a Session

is allowed to stop the charging session using the same steps as in the live authorization request with the additional validation of whether both EIM devices are the same or at least in the same group.

Whitelist Updates

Since each authorization value can only be used once, an eMSP's whitelist needs to be regularly updated. The general update procedure follows the process for preparation (cf. Fig. 7.1), i.e., the eMSP generates the next authorization values for their customers and distributes these values as whitelist. That is, the eMSP computes new transaction pseudonyms $\langle id_x^i, auth_x^i \rangle$[2] for their users $x \in [1, \ldots, m]$ and authorizations $i \in [n+1, \ldots, n']$ (with $n' > n+1$). This update procedure is executed periodically and can account for the number of charge authorizations per user (e.g., by only uploading new authorization values to replenish used ones). New whitelist can simply be added to the existing ones by the CPS, CSOs, and CSs, i.e., do not completely replace them.

In order to distribute authorization values for new customers, the eMSP could add these values to their periodic whitelist updates. Note that this requires online authorization requests for all new customers in the time between the conclusion of their contract and the whitelist update. Alternatively, the eMSP may send separate whitelist updates for new users. Note, however, that updates for single users would enable the trivial linkability of this users different authorization values. Hence, any update should include new authorization values for multiple users.

Since each authorization value can only be used once, all actors should delete used values out of their whitelists. For instance, eMSPs could request the deletion of used values in their periodic whitelist updates. Alternatively, after the end of a charging session, CSs can directly delete used values, CSOs can inform their other CSs, and the CPS can inform other CSOs.

7.1.3. Remote Authorization and Reservations

The method of remote charge authorization (e.g., smartphone-based) in the current EV charging protocols requires the EV user to disclose their desired charge location to the eMSP. Since this method clearly threatens

[2] Calcualted as before, i.e., $id_x^i = Hash(hmac_{ID_x}(00||i))$ and $auth_x^i = hmac_{ID_x}(01||i)$.

the user's location privacy, we propose the following change: Instead of only receiving the respective CSID, the EV user's device (e.g., their smartphone) additionally receives the certificate chain of the corresponding CSO. The certificate chain is received via the usual communication channel (e.g., via a web service or a QR code at the CS). After validating the received certificate chain, the user's device locally encrypts the CSID with the CSO's public key from the leaf certificate. The encrypted CSID is sent together with the CSO's ID to the eMSP. The eMSP verifies that the user is authorized to charge at CSs of the respective CSO and sends a remote authorization request to the CSO with a random ID Token. The CSO trusts the eMSP that the random ID Token is associated with a valid user and only uses this ID Token for billing purposes, i.e., the CSO later reports the random ID Token in the CDR to the eMSP. The eMSP stores the mapping between the random ID Token and the EV user in order to bill the correct user based on the CSO's CDR. Finally, the CSO can decrypt the CSID and otherwise handles the remote authorization request as usual.

In order to also support remote reservations further considerations are required since, opposed to the remote authorization case, an EV user needs to be authorized locally (e.g., via an RFID card) in order to ensure that a reservation is only consumed by the user who placed it. The method for CS reservations in the current EV charging protocols requires the eMSP to send the EV user's ID Token over CCH and CSO to the respective CS. This is no longer possible with the proposed method for privacy-preserving authorization since the eMSP cannot necessarily predict the next authorization value of the user's local authorization EIM device (e.g., an RFID card) as, for example, after an offline authorization the eMSP is not aware of the current value of i_x. For this reason, we introduce a second secret key for the local authorization EIM device ID_x^r and a second counter i_x^r (as an extension to the initialization of the device described in Section 7.1.2). Both ID_x^r and i_x^r are used exclusively for reservation and thus predictable by the eMSP.

After receiving a reservation request (with an encrypted CSID as used for remote authorization), the eMSP builds $id_x^r = Hash(hmac_{ID_x^r}(00||i_x^r + 1))$ and $auth_x^r = hmac_{ID_x^r}(01||i_x^r + 1)$ with the user ID_x^r and current i_x^r. The values id_x^r and $auth_x^r$ are sent over the CCH to the identified CSO. The CSO decrypts CSID and builds $idCS_x^r = Hash(id_x^r||CSID)$ as well as $authCS_x^r = Hash^3(auth_x^r||nonce_x^r)$ with a random $nonce_x^r$. Finally, $reservationAuth_x^r = \square idCS_x^r, nonce_x^r, authCS_x^r \square$ is sent to the CS.

The CS can use this data with a modified version of the proposed offline charge authorization method (cf. Fig. 7.2). The difference to Fig. 7.2 is that the first request of the CS indicates that a reservation is currently active (e.g., by replacing "Start" with "Reservation") and includes $idCS_x^r$ and $CSID$. The inclusion of $idCS_x^r$ and $CSID$ is required in order to prevent that a different EIM device (e.g., RFID card) updates its local state as follows: The EIM device first calculates $idCS' = Hash^2(hmac_{ID_x^r}(00||i_x^r + 1))$ and then verifies that $idCS'$ is equal to $idCS_x^r$. If it is equal, i_x^r is incremented and the usual protocol flow is followed (cf. Fig. 7.2). If it is not equal, an error is returned and communication is aborted.

7.2. Privacy-Preserving PnC-Based Authorization

The general idea of the the privacy-preserving method of PnC-based charge authorization is to use anonymous ECC-DAA-based signatures for the contract credential-based authentication of an EV towards a CS and combine this approach with the transaction pseudonym ID Tokens from Section 7.1 for authorization. The DAA scheme is based on the one proposed in [205] as described in Section 2.5.4 with the necessary changes to make it compatible with the EV charging infrastructure. All security-relevant data of the EV is protected by a TPM 2.0 to prevent its leakage while being stored and used even from the Local Adversary. The integration of the TPM-based DAA scheme uses components from [66, 67] as described in Section 2.5.5.

Section 7.2.1 describes the changes/additions to the TPM key profiles and certificate extension from [67] that are needed for the DAA and authorization schemes. Section 7.2.2 describes the necessary processes for the preparation of the proposed method. As per [66, 67], the EV's provisioning credentials are generated in its TPM and sealed to a trusted software state via an authorization policy. Additionally, the EV is provided with an endorsement key pair that is needed for the join protocol of [205], which is used for the provisioning of DAA contract credentials.

Section 7.2.3 describes the generation of contract credentials. Depending on whether or not certificate pools are used, this process may be done before (based on data from the pools) or during (based on data from the EV's request) a V2G communication session. In any case, the contract key pair is generated in the EV's TPM and the eMSP only generates the corresponding DAA credential for authentication, i.e., the eMSP is acting as issuer join protocol of [205], and an eMAID key for authorization based on the scheme from Section 7.1. Both of these values are sealed to the EV's TPM using the public endorsement key.

Section 7.2.4 describes the process for provisioning of contract credentials, including the generation and handling of contract credential requests and -responses. The provisioning of contract credentials uses a modified version of the join protocol from [205]. The join protocol needs to be changed for several reasons:

- The join protocol requires the signer to know the issuer's public key I_{pk}^{group}. As EVs do not know with which eMSP the user has a contract with, this is not possible.
- The join protocol requires two round trips. On the first round trip, the signer sends its public endorsement key and public DAA key to the issuer and the issuer responds with an encrypted nonce. On the second round trip, the signer sends the decrypted nonce along with a signature and the issuer responds with the decrypted DAA credential. A two round trip certificate installation, however, is argued to add too much overhead. For example, the installation of contact credentials using the current method with one round trip (cf. Section 4.2.5) may already take several minutes depending on the CS' uplink (cf. [98], Section 8.4.3.10).[3]
- Since a malicious CS might be able to trigger an EV to send a new contract installation request (PT_3), the requests should not leak a static identifier (the public endorsement key) of the EV.
- Support for the certificate pool method should be considered.

Finally, Section 7.2.5 describes the use of contract credentials using a modified version of the certify protocol from [205] that additionally binds a certified key to a specific charging session. The EV's DAA credentials are used to certify a session key pair for authentication during a V2G session and the eMAID key is used with the scheme from Section 7.1 for offline authorization.

7.2.1. Components

The following sections describe the components of the proposed solution. First, the used TPM key profiles (based on [66, 67] with changes/additions for DAA as per [205] and authorization as per Section 7.1), regulating how the respective keys can be used and how their usage can be authorized, are described. Additionally, the application of the provisioning certificate extension from [67], used to provide additional information about an EV from its OEM to eMSPs in an authentic manner, and an extension to eMSP certificates, allowing the integration of the issuer's public group key, are described.

[3] Note that the installation time of several minutes is an estimate in the draft of ISO 15118-20 [98], which allows the installation of multiple contract certificates, i.e., a CS might have to download multiple certificate installation responses for one request.

Table 7.1.: TPM 2.0 Key Templates

	EC	PC	CC^{DAA}	CC^{Sess}	SK_{eMAID}
type:	TPM2_ALG_ECC				TPM2_ALG_KEYEDHASH
nameAlg:	TPM2_ALG_SHA256				
object-Attributes:	fixedTPM, fixedParent, restricted, decrypt, adminWithPolicy, sensitiveDataOrigin	fixedTPM, fixedParent, sign, decrypt, sensitiveDataOrigin	fixedTPM, fixedParent, restricted, sign, sensitiveDataOrigin	fixedTPM, fixedParent, sign, userWithAuth, sensitiveDataOrigin	sign, userWithAuth
authPolicy:	TPM2_PolicySecret(TPM_RH_ENDORSEMENT)	Pol_{Auth} = TPM2_PolicyAuthorize(OEM_{pk})		$\perp$	
symmetric:	AES-128-CFB	n/a			
scheme:	n/a		TPM2_ALG_ECDAA TPM2_ALG_SHA256	$\perp$	TPM2_ALG_HMAC TPM2_ALG_SHA256
curveID:	TPM2_ECC_NIST_P256		TPM2_ECC_BN_P256	TPM2_ECC_NIST_P256	n/a

TPM 2.0 Key Profiles

Table 7.1 shows the different profiles of the used TPM keys. In general, the solution requires five different kinds of keys: (i) an endorsement credential EC key pair, (ii) a provisioning credential PC key pair, (iii) a DAA contract credential CC^{DAA} key pair, (iv) a session contract credential CC^{Sess} key pair (generated anew for every V2G communication session), and (v) a symmetric eMAID key SK_{eMAID}. The definition of the EC follows the TCG's profile for endorsement credentials [191], the definition of PC follows [66, 67] (cf. Table 2.5), the definition of CC^{DAA} follows that of the DAA key in [205] with the addition of an authorization policy, the definition of CC^{Sess} follows that of the ISO 15118 contract key in [67] except that its usage is not conditioned on a policy, and SK_{eMAID} is defined to enable authorization as per Section 7.1.

All key pairs are of type ECC whereas the eMAID key is of type keyed hash, i.e., for HMAC calculations. The fixedTPM and fixedParent attributes are set for all key pairs such that they cannot be exported out of the TPM or moved within its hierarchy (cf. [195], Section 25). Also the sensitiveDataOrigin attribute is set for all key pairs indicating that their private values were generated by the TPM. These three attributes are not set for the symmetric eMAID key since it is generated by the eMSP and only imported into the EV's TPM. Note, however, that SK_{eMAID} still cannot be exported out of the TPM since it has an empty ($\perp$) authPolicy and an export requires the key's authPolicy to be set and include a special authorization for the `TPM2_Duplicate` command (cf. [195], Section 23.3).

The restricted, decrypt, and sign attributes indicate the usages of the keys (cf. [195], Section 25). The EC private key is a restricted decryption key, meaning that its private key can only be used with certain commands and on specifically formatted objects and that its private key can only be used for decryption. The restrictions of the EC key pair are used in order to protect contract credentials (the DAA credential as per [205] and SK_{eMAID} for authorization) during transit from eMSP to EV and to guarantee that contract credentials can only be decrypted (and thus used) with the intended TPM. The PC private key is a general purpose key for signing and decryption, which is required for its normal ISO 15118 uses. The CC^{DAA} private key is a restricted DAA signing key, i.e., it can only be used to sign specifically formatted structures. This restriction is needed since the CC^{DAA} private key is used to certify (using the `TPM2_Certify` command) CC^{Sess} session key pairs for use during charging sessions and the output of a `TPM2_Certify` can only be trusted if it was generated by a restricted signing key. The CC^{Sess} private key itself is a regular signing key. The SK_{eMAID} key is a regular HMAC key, i.e., a key of type keyed hash with the sign attribute set.

The adminWithPolicy and userWithAuth attributes determine how key usage can be authorized (cf. [195], Section 25). The EC key pair has adminWithPolicy set and userWithAuth clear, meaning that any kind of use requires a valid policy session. Since the EC key pair's authPolicy is set in accordance with [191], authorization to use this key is bound to the endorsement hierarchy authorization. The PC and CC^{DAA} key pairs both have adminWithPolicy and userWithAuth cleared and are both sealed to the same authPolicy value (a `TPM2_PolicyAuthorize` with the OEM's public key; cf. 2.5.2) such that the processes for credential provisioning and charge authorization both require an assertion of an OEM-signed policy digest.[4] The CC^{Sess} key pair and SK_{eMAID} have adminWithPolicy clear and userWithAuth set, meaning that any authorization method can be used.

The symmetric field of the EC key pair indicates the "companion" symmetric en-/decryption algorithm to be used with this key, or more precisely, with a symmetric key that was derived based on the EC key pair (e.g., for credential protection; cf. Section 2.5.3). This field is only used for restricted decryption keys and thus does not apply to any other key pairs. The scheme field is required for restricted signing keys and indicates the used signing scheme, i.e., ECC-DAA with SHA256 for the CC^{DAA} key pair. Additionally, the scheme field is required for HMAC keys, i.e., for SK_{eMAID}, and indicates the used HMAC scheme. The curveID field only applies to ECC key pairs and indicates their respective curve, i.e., BN_P256 for the CC^{DAA} key pair and NIST_P256 for the others. Note that the BN_P256 provides less than 128-bit security [11], i.e., for an implementation to be considered computationally secure by current standards (e.g., [12]) a more secure curve would have to be used for the DAA key. While the TPM specification includes the BN_P638 curve [196] (providing more than 128-bit security [167]), most TPM implementations currently do not support this curve (cf., e.g., [205]). For this reasons, the BN_P256 curve is still used in this thesis.

Certificate Extensions

Since for [205], the security of DAA credentials is provided by the endorsement key pair (EC_{pk}, EC_{sk}) in the signer's (the EV's) TPM, eMSPs (as issuers) need to trust in the authenticity of the respective public key EC_{pk} when they issue DAA contract credentials. Similar to [66, 67], this trust can be provided by an inclusion of EC_{pk} in the EV's provisioning certificate PC_{cert}, which is signed by the OEM and verifiable by eMSPs, via a non-critical SIA extension as shown in Fig. 7.4a. Additionally, eMSPs need to receive the intended authPolicy of a CC^{DAA} key pair in a trustworthy manner for the credential protection mechanism (cf. Section 2.5.3). As done in [66, 67], the intended authPolicy is also included in the EV's provisioning certificate PC_{cert}. Hence, the only difference in the use of the provisioning certificate extension from [66, 67] is the inclusion of EC_{pk} (used in [205] for encryption in the credential protection mechanism) instead of the generic public storage key.

While [205] assumes that the issuer's group public key is known to verifiers, a similar assumption is not justified in the EV charging use case (CSs would have to install the certificates of every eMSP instead of only the root). In order to provide an eMSP's public group key $eMSP_{pk}^{group}$ to verifiers in an authentic manner, we include this key in the eMSP's certificate. The key is included via an extension since, to the best of our knowledge, no standardized method of including an ECC-DAA issuer's public group key in an X.509 certificate exists and since an eMSP's normal ECDSA public key ($eMSPSubCA2_{pk}$) is still required such that an EV can verify tariff signatures (cf. Section 4.1.4). Similar to [66, 67], the extension is implemented as a non-critical

[4] Since PC and CC^{DAA} are sealed to the same policy, setting the adminWithPolicy for the CC^{DAA} key pair would require two assertions of the same policy during credential provisioning as this process requires admin role authorization for the CC^{DAA} key pair. None of the proposed processes require admin role authorization for the PC key pair, hence, adminWithPolicy does not make a difference here.

OEM Provisioning Certificate (PC_{cert})		
Version:		X.509v3 (0x2)
Serial Number:		12345 (0x3039)
Signature Algorithm:		ecdsa-with-SHA256
Issuer:		CN=OEMSubCA2, O=ISO
Validity	Not Before:	May 7 08:40:32 2020 GMT
	Not After:	May 6 08:40:32 2050 GMT
Subject:		DC="OEM", CN=WDKA2B3C4D5E6F7G84, O=ISO
Subject Public Key Info	Public Key:	OCTET STRING (PC_{pk})
	Algorithm:	id-ecPublicKey
	Parameters:	namedCurve secp256r1
X509v3 Extensions	Basic Constraints:[c]	CA:FALSE
	Key Usage:[c]	Digital Signature, Key Agreement
	Subject Key Identifier:[nc]	64 bit ID of PC_{pk}
	Authority Key Identifier:[nc]	64 bit ID of $OEMSubCA2_{pk}$
	CRLDistributionPoints:[nc]	URI:http://oem.com/oem.crl
	Authority Information Access (OCSP):[nc]	URI:http://ocsp.oem.com/
	Subject Information Access:[nc]	**OID:1.0.20.4** **TPM 2.0 EC Public Endorsement Key (EC_{pk})** **512 bit OCTET STRING in Base64** **OID:1.0.20.5** **TPM 2.0 SHA256 Policy Digest (Pol_{Auth})** **256 bit OCTET STRING in Base64**
Signature	Algorithm:	ecdsa-with-SHA256
	Value:	OCTET STRING (Signature with $OEMSubCA2_{sk}$)

(a) Provisioning Certificate with SIA Extension (cf. [67])

eMSP Certificate ($eMSP_{cert}^{group}$)		
Version:		X.509v3 (0x2)
Serial Number:		12345 (0x3039)
Signature Algorithm:		ecdsa-with-SHA256
Issuer:		CN=eMSPSubCA1, O=ISO
Validity	Not Before:	May 7 08:40:32 2020 GMT
	Not After:	May 6 08:40:32 2050 GMT
Subject:		CN=eMSPSubCA2, O=ISO
Subject Public Key Info	Public Key:	OCTET STRING ($eMSPSubCA2_{pk}$)
	Algorithm:	id-ecPublicKey
	Parameters:	namedCurve secp256r1
X509v3 Extensions	Basic Constraints:[c]	CA:TRUE pathLenConstraint=0
	Key Usage:[c]	Digital Signature, Non-Repudiation
	Subject Key Identifier:[nc]	64 bit ID of $eMSPSubCA2_{pk}$
	Authority Key Identifier:[nc]	64 bit ID of $eMSPSubCA1_{pk}$
	CRLDistributionPoints:[nc]	URI:http://emsp.com/emsp.crl
	Authority Information Access (OCSP):[nc]	URI:http://ocsp.emsp.com/
	Subject Information Access:[nc]	**OID:1.0.20.6** **ECC-DAA Public Group Key ($eMSP_{pk}^{group}$)** **2048 bit OCTET STRING in Base64**
Signature	Algorithm:	ecdsa-with-SHA256
	Value:	OCTET STRING (Signature with $eMSPSubCA1_{sk}$)

(b) eMSP Certificate with SIA Extension

Figure 7.4.: Extended ISO 15118 Certificate Profiles

SIA extension as shown in Fig. 7.4b. The extension consists of a single `AccessDescription` element with the OID 1.0.20.6 indicating that the content is a public ECC-DAA group key.

7.2.2. Preparations

In order to setup the proposed method, certain preparations are required. Namely, this includes the initialization of the EV and its TPM with secure provisioning credentials based on [66, 67], the initialization of CSs, CSOs, the CPS, and eMSPs for the handling of certificate installation requests, as well as the setup of the used DAA scheme based on [205]. The following sections describe the preparatory processes in detail.

Initialization of the EV

In order to prepare an EV for the privacy-preserving method of PnC-based charge authorization, it is equipped with a TPM. As per [66, 67], in order to defend against the Local Adversary, the TPM is used to protect all of the EV's sensitive data, to provide a protected environment for cryptographic operations, and to measure the EV's software state during a measured boot. The measurements are recorded in the TPM's PCRs and used to seal the EV's credentials to a trusted software state, i.e., to the expected PCR values. All following key generations use the key profiles as described in Section 7.2.1.

As per [66, 67], an EV and its TPM are initialized by the OEM some time before the initial delivery. The only difference to the initialization process from [66, 67] (cf. Section 2.5.5) is that the endorsement key pair (EC_{pk}, EC_{sk}) replaces the storage root key pair (SRK_{pk}, SRK_{sk}). That is, the OEM instructs the EV's TPM to generate the endorsement key pair (EC_{pk}, EC_{sk}) and the provisioning key pair (PC_{pk}, PC_{sk}), reads out the respective public keys (EC_{pk} and PC_{pk}), and generates the provisioning certificate PC_{cert} for PC_{pk} including

EC_{pk} and Pol_{Auth} (i.e., a `TPM2_PolicyAuthorize` for the OEM's public key OEM_{pk}; cf. Section 7.2.1) via the certificate extension as described in Section 7.2.1. The provisioning certificate is provided to the EV and can be used to indicate to other actors that the provisioning and endorsement key pairs were securely generated in a trustworthy TPM.

If certificate pools are used, the OEM further instructs the EV's TPM to generate a contract key pair as per [67]. The only difference is that a DAA contract key pair ($CC_{pk}^{DAA}, CC_{sk}^{DAA}$) is generated. The corresponding public key CC_{pk}^{DAA} is read out by the OEM and uploaded together with PC_{cert} and the corresponding PCID into the provisioning certificate pool.

Similar to [66, 67], usage of the PC and CC^{DAA} key pairs is conditioned on a `TPM2_PolicyAuthorize` policy, meaning that in order to enable the EV to use its PC and CC^{DAA} key pairs, the OEM provides a policy digest Pol_{PCR} together with a signature $\sigma_{Pol_{PCR}}$ over this digest to the EV. The signature $\sigma_{Pol_{PCR}}$ is created with the OEM's private key OEM_{sk} corresponding to the OEM_{pk} that was used for the `TPM2_PolicyAuthorize`. The policy digest Pol_{PCR} includes a single policy command, namely a `TPM2_PolicyPCR` in order to seal the PC and CC^{DAA} key pairs to a trusted software state while allowing firmware upgrades (cf. Section 2.5.2).

The EV can use Pol_{PCR} and $\sigma_{Pol_{PCR}}$ to build a policy session in its TPM that authorizes the usage of PC_{sk} or CC_{sk}^{DAA} as shown in Fig. 7.5. First, the signature $\sigma_{Pol_{PCR}}$ needs to be verified with OEM_{pk} using the `TPM2_VerifySignature` command. The result is a signature verification ticket $TK_{validation}$, which is calculated by the TPM as an HMAC over Pol_{PCR} and the *name* of OEM_{pk} (i.e., a hash over the public area of OEM_{pk}). The HMAC is calculated based on a $proof$ value, i.e., a TPM-internal secret key (cf. [195], Section 14.4). Afterwards, the EV starts a policy session in its TPM and uses the `TPM2_PolicyPCR` command to update the session digest with a hash over a selection of PCRs. The selected PCRs correspond to the ones used during the measured boot. Finally, the EV calls the `TPM2_PolicyAuthorize` command, which validates that the current session digest is equal to Pol_{PCR} and verifies the signature verification ticket $TK_{validation}$. If both checks succeed, then the TPM resets the current session digest and updates it with the *name* of OEM_{pk}. If the resulting session digest is equal to the one provided during the creation of the PC and CC^{DAA} key pairs, then the session can be used to authorize key usage.

Initialization of Other Actors

In order to handle certificate installation requests in the backend, the mapping between PCIDs and eMSP needs to be setup at the CPS. For this, the eMSP informs the CPS about the newly registered PCID, whenever an EV user has registered their PCID with them. The CPS stores all PCID to eMSP mappings in order to forward an EV's certificate installation requests. If certificate pools are used the CPS uses the PCID to find an EV's credential in the pool. Additionally, in order to protect the personal information in certificate installation requests, the requests must be encrypted by the EV for the CPS. This requires that an EV can request the CPS' public key over ISO 15118 at a CS. For this, the CSO requests the public key certificate chains (up to the V2G root) of relevant CPSs and distributes them to its CSs.

Initialization of the DAA Scheme

To use the DAA scheme, its general parameters (security parameter, bilinear groups, and generators) need to be setup accordingly (cf. the setup process in [205] as described in Section 2.5.4). These parameters, e.g., could be defined in the ISO 15118 standard. Additionally, the issuers (eMSPs) must generate their key pairs ($eMSP_{pk}^{group}, eMSP_{sk}^{group}$). The respective public keys are included in an eMSP's certificate $eMSP_{cert}^{group}$

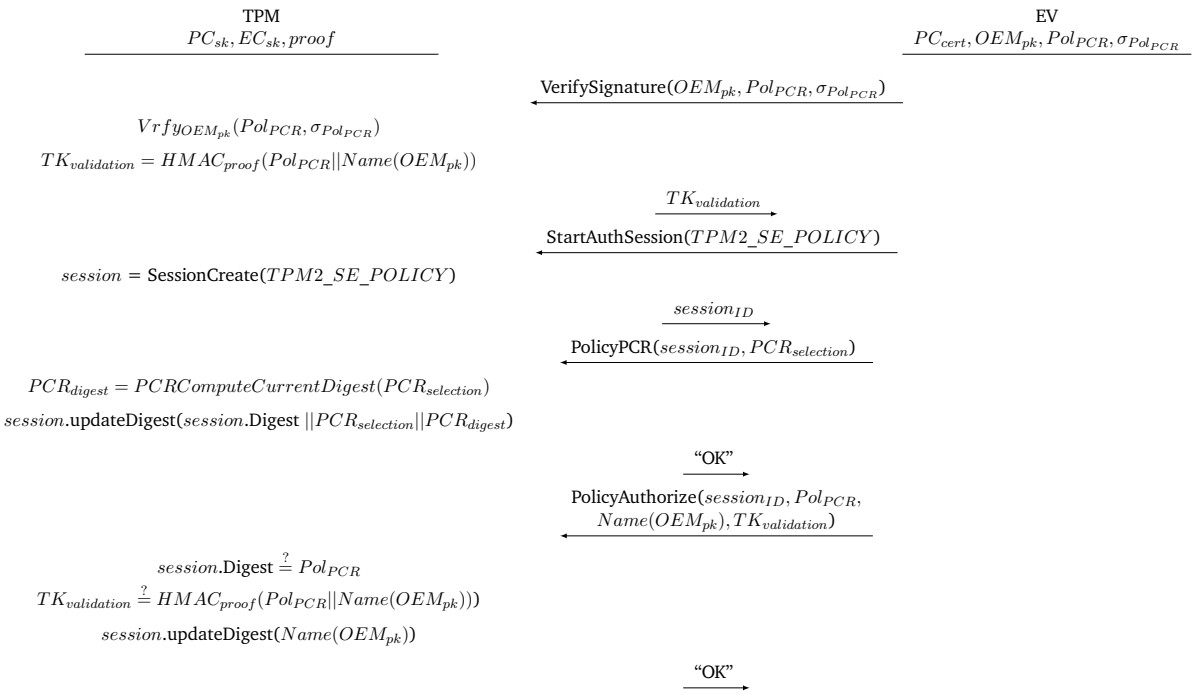


Figure 7.5.: Building a Policy Session to Use the EV's PnC Credentials

(cf. Fig. 7.4b) such that they can be validated by CSs, CSOs, and the CPS based on a eMSP certificate root $eMSP_{Root}$.

7.2.3. Contract Credential Generation

The process of contract credential generation is changed by replacing the usual ECDSA contract key pair with a privacy-preserving DAA key pair, which is generated in the EV's TPM. Additionally, the usual contract certificate is split into a DAA credential for the DAA key pair and a symmetric eMAID key, which is sealed to the same TPM as the DAA key pair.

Since the contract key pair is generated in the EV's TPM, the respective public key CC_{pk}^{DAA} needs to be sent to the eMSP. If certificate pools are not used, then this key is received via the EV's certificate request. If certificate pools are used, then the eMSP downloads an EV's provisioning data (including CC_{pk}^{DAA} and PC_{cert}) from the provisioning certificate pool based on the PCID. The eMSP proceeds to generate DAA contract credentials as shown in Fig. 7.6.

The eMSPs generates the DAA credential CC_{cred}^{DAA} for the EV's public DAA contract key CC_{pk}^{DAA} as described in [205] (cf. Section 2.5.4). Afterwards, the CC_{cred}^{DAA} is signed by the eMSP using its private key $eMSP_{sk}^{group}$ and the CC_{cred}^{DAA} concatenated with the signature σ_{cred} are encrypted with a newly generated symmetric key SK_{CC}, resulting the ciphertext C_{CC} (cf. [205]). In order to inform an EV about the validity period of its credentials, CC_{cred}^{DAA} could additionally be appended with a start and expiration date before encryption such that this information is not revealed to anyone but the correct EV (not shown in Fig. 7.6). Afterwards, the eMSP encrypts SK_{CC} with the public endorsement key EC_{pk} from the extension in the EV's certificate PC_{cert} using the TPM credential protection mechanism (cf. Section 2.5.3), resulting the ciphertext $C_{SK_{CC}}$. Since the credential protection encryption mechanism also includes the public data of the EV's DAA contract key and

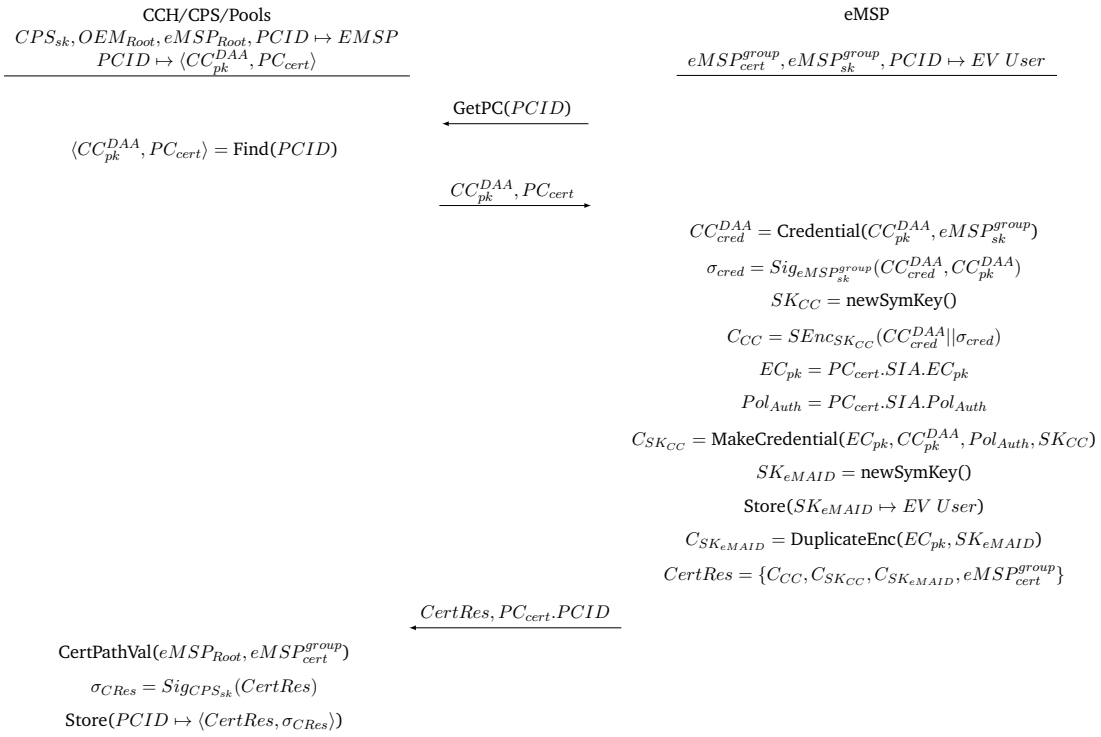


Figure 7.6.: Contract Credential Generation with Certificate Pools

the contract key's intended policy (from the PC_{cert} extension), SK_{CC} can only be decrypted if CC_{sk}^{DAA} was generated with the correct attributes (cf. Table 7.1) and in the same TPM as EC_{sk} (cf. Section 2.5.3). Since SK_{CC} is needed for the decryption of CC_{cred}^{DAA} and CC_{cred}^{DAA} is needed for the DAA-based authentication with CC_{sk}^{DAA}, this means that the EV can only authenticate itself as a member of the eMSP's DAA group if CC_{sk}^{DAA} was generated with the correct attributes and in the same TPM as EC_{sk}.

Additionally, the eMSP generates another symmetric key SK_{eMAID}, which uniquely identifies the EV user's account with the eMSP and is used to replace the eMAID. The eMSP stores the mapping between SK_{eMAID} and EV user and encrypts SK_{eMAID} with EC_{pk} resulting in $C_{SK_{eMAID}}$. The encryption of SK_{eMAID} uses the TPM's object duplication procedure (cf. [195], Section 23.3), such that $C_{SK_{eMAID}}$ can be imported into the key hierarchy of the EV's TPM (cf. Section 2.5.1).

Finally, the eMSP builds the certificate installation response $CertRes$ with the values C_{CC}, $C_{SK_{CC}}$, and $C_{SK_{eMAID}}$ as well as its own certificate $eMSP_{cert}^{group}$. $CertRes$ is sent with the corresponding PCID to the CPS, which validates the eMSP's certificate $eMSP_{cert}^{group}$ and signs $CertRes$ with its private key CPS_{sk}. Afterwards, the CPS stores $CertRes$ together with the newly created signature σ_{CRes} in the contract certificate pool. The contract certificate pool data is indexed via the PCID.

7.2.4. Contract Credential Provisioning

The process for installing new privacy-preserving contract credentials for ISO 15118 PnC is based on the ECC-DAA join protocol using a TPM 2.0 from [205] (cf. Section 2.5.4). For compatibility with the EV charging infrastructure, the join protocol from [205] is adjusted such that signers (EVs) do not need to know their

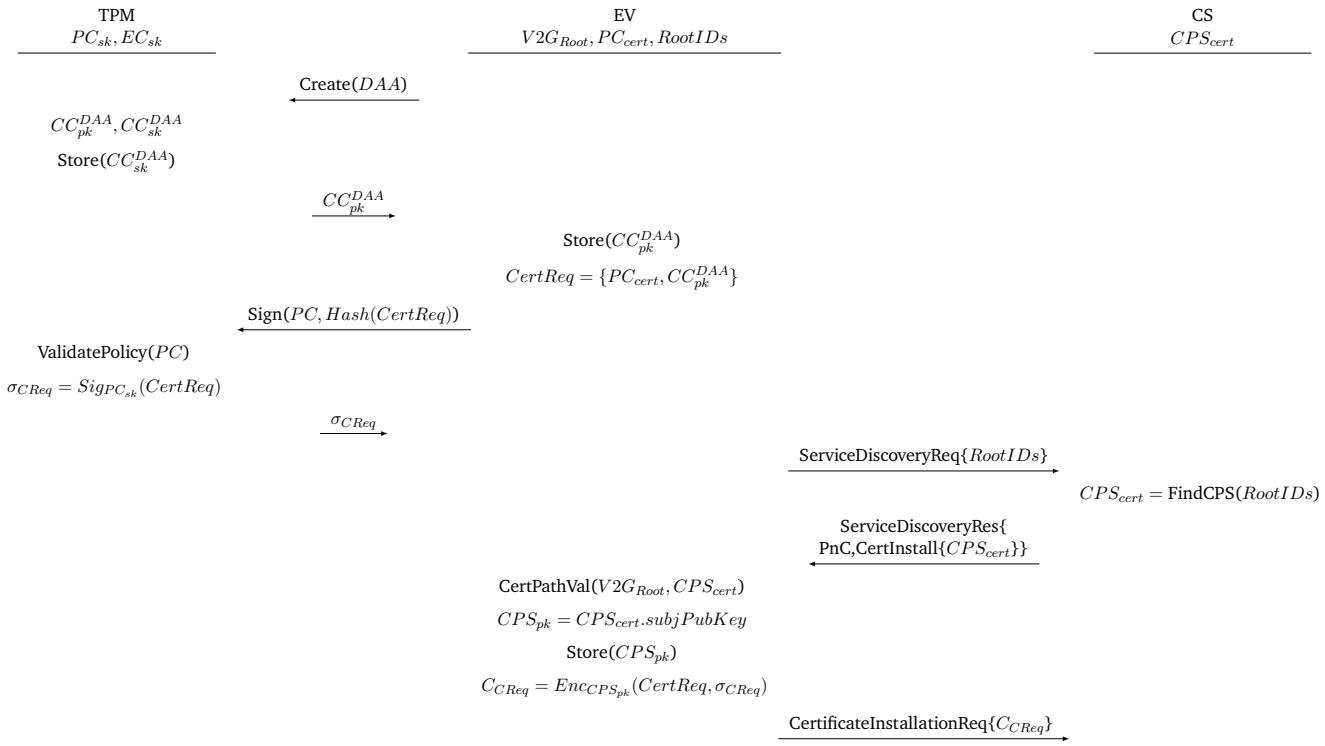


Figure 7.7.: Generating and Sending the Contract Credential Installation Request

issuers (eMSPs) beforehand, such that it only requires one round trip between signer and issuer, and such that personal data is protected from a malicious CS. The modified join protocol should still provide the same agreement property as [205], as authentication during credential installation. Additionally, the process includes the provisioning of a TPM-bound eMAID key such that DAA contract credentials and eMAID are guaranteed to be protected by the same TPM and also are only usable by this TPM.

Contract Credential Request Generation and Sending

The provisioning process starts with the generation of a certificate installation request as shown in Fig. 7.7. The EV starts by instructing its TPM to generate a DAA contract key pair ($CC_{pk}^{DAA}, CC_{sk}^{DAA}$). The new public key CC_{pk}^{DAA} is returned to the EV and the private key CC_{sk}^{DAA} is stored in the TPM.[5] Afterwards, the EV can use its new CC_{pk}^{DAA} to build the certificate request data $CertReq$ including its provisioning certificate PC_{cert} and can instruct its TPM to sign this data with the private provisioning key PC_{sk}. Creating a signature with the PC_{sk} requires a correct assertion of its policy (cf. Fig. 7.5), i.e., it requires that the EV booted into a trusted software state (cf. Section 2.5.2). Note that all previous steps can be performed independently from an ISO 15118 communication session.

When sending the request data over ISO 15118, the data needs to be encrypted for its destination in order to prevent the unnecessary exposure of personal information as shown in Fig. 7.7. After the usual ISO 15118

[5] As described in Section 2.5.1, private keys are not directly stored by the TPM. Instead, they are encrypted based on their parent key and the resulting ciphertext is returned to the host (i.e., the EV). Afterwards, in order to use the private key, it needs to be loaded into to the TPM where it is decryted based on its parent key. For simplicity, we model this process by storing private keys directly in the TPM.

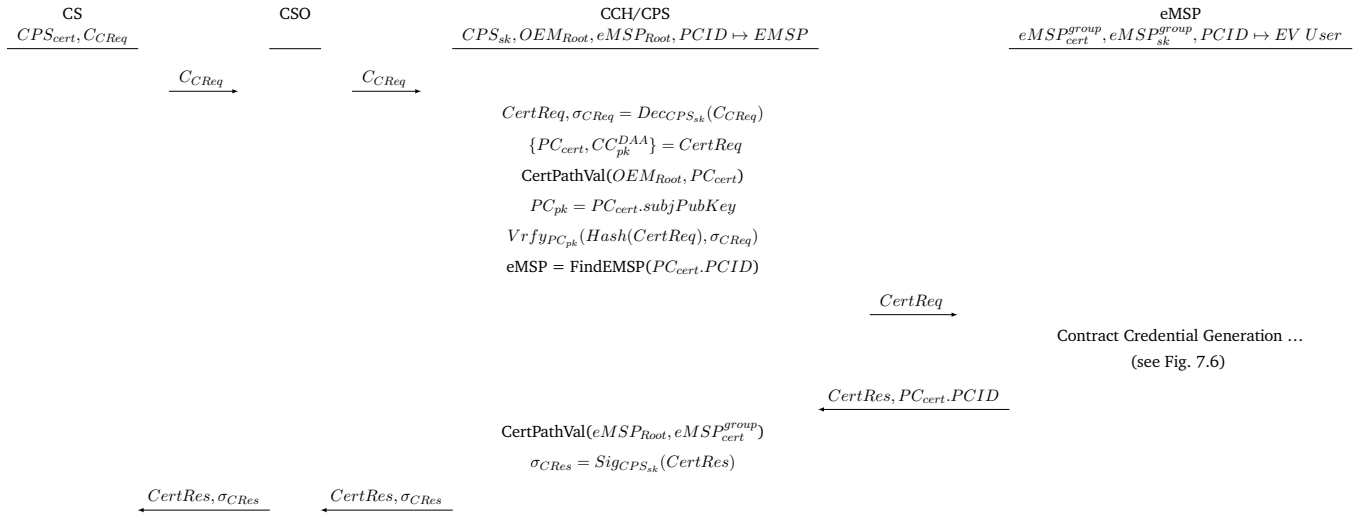


Figure 7.8.: Contract Credential Installation Request Handling in the Backend

communication setup (not shown), the EV requests the CS' available services with a *ServiceDiscoveryReq*. The *ServiceDiscoveryReq* already identifies the EV's installed V2G certificate roots. The CS uses this list of roots to find an appropriate CPS certificate chain, i.e., a chain that can be validated based on one of the EV's installed roots, and includes this chain (CPS_{cert}) in the *ServiceDiscoveryRes* message. Afterwards, the EV validates the received chain based on its V2G roots (CertPathVal) and extracts the public key CPS_{pk} from the CPS' certificate. CPS_{pk} is used to encrypt $CertReq$ with the corresponding signature σ_{CReq}. The resulting ciphertext C_{CReq} is sent to the CS in a *CertificateInstallationReq* message, whereby the usual ISO 15118 application layer signature over the message is omitted as it is replaced by the signature over $CertReq$. Note that if a CS or CSO where to receive this request's data in plaintext, it would allow them to link the contained PCID (which is assumed to uniquely identify the EV user) to the charge location, threatening the user's location privacy.

Contract Credential Request Handling and Response Generation

C_{CReq} is forwarded from the CS to its CSO and from the CSO to the CPS as shown in Fig. 7.8. The CPS decrypts C_{CReq} using its private key CPS_{sk} resulting in $CertReq$ and σ_{CReq}. Afterwards, the CPS validates PC_{cert} from $CertReq$ based on its available OEM certificate roots and extracts the public provisioning key PC_{pk} from PC_{cert}. The CPS uses PC_{pk} to verify the signature over $CertReq$. If certificate pools are used (not shown), the CPS loads all certificate installation responses (including $CertRes$ and σ_{CRes}) for the requestor's PCID from the contract certificate pool and sends these (without the PCID) over the CSO to the CS.

If certificate pools are not used, the CPS finds all eMSPs that have registered a contract for the PCID from PC_{cert}, and forwards the certificate request to these eMSPs. The eMSPs can now generate the DAA credential CC_{cred}^{DAA} for the EV's public DAA contract key CC_{pk}^{DAA} and generate the certificate installation response $CertRes$ as described in Section 7.2.3 and shown in Fig. 7.6. $CertRes$ is sent to the CPS, which validates the eMSP's certificate and signs $CertRes$ with its private key CPS_{sk}, resulting in the signature σ_{CRes}. Finally, CPS forwards $CertRes$ with the signature σ_{CRes} over the CSO to the CS.

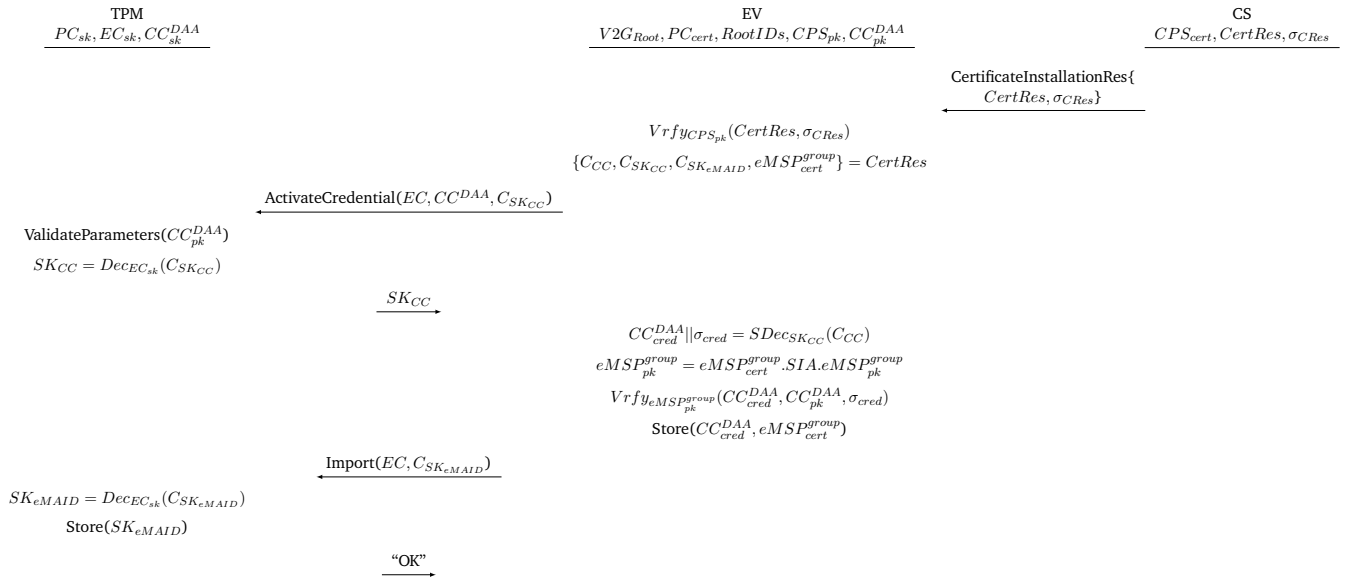


Figure 7.9.: Contract Credential Installation Response Handling

Contract Credential Response Handling

After the CS receives $CertRes$ and σ_{CRes} from its CSO, it forwards this data to the EV in a *CertificateInstallationRes* message as shown in Fig. 7.9. The EV starts by verifying the signature over $CertRes$ with the previously received CPS's public key CPS_{pk}. Afterwards, the EV uses its TPM to decrypt $C_{SK_{CC}}$ with EC_{sk}, resulting in SK_{CC}. This decryption uses the `TPM2_ActivateCredential` command, which requires the DAA contract key pair to be loaded in the TPM and verifies that this key pair has the expected parameters and is sealed to the intended policy. Note that `TPM2_ActivateCredential` requires admin role authorization for the DAA contract key pair, i.e., it does not require an assertion of its policy (cf. Section 7.2.1).

SK_{CC} is used by the EV to decrypt C_{CC}, resulting in CC_{cred}^{DAA} and σ_{cred}. Afterwards, the EV uses the eMSP's public key from $eMSP_{cert}^{group}$ to verify the signature σ_{cred} over CC_{cred}^{DAA} and stores its new credential CC_{cred}^{DAA} together with the eMSP's certificate $eMSP_{cert}^{group}$. Note that in the ISO 15118 PKI the EV cannot validate eMSP's certificate itself since it does not possess the necessary roots. Instead, the EV has to trust the CPS' signature over $CertRes$. These trust relations are not changed with the proposed method.

Finally, the EV triggers its TPM to decrypt the encrypted eMAID key $C_{SK_{eMAID}}$ using the `TPM2_Import` command. The resulting SK_{eMAID} is imported into the TPM's key hierarchy below the endorsement key.

7.2.5. Credential Usage

The process for using the privacy-preserving contract credentials for ISO 15118 PnC is based on the ECC-DAA certify protocol using a TPM 2.0 from [205] (cf. Section 2.5.4). First, in the initialization phase, the EV generates a new session key pair $(CC_{pk}^{Sess}, CC_{sk}^{Sess})$ in its TPM. Afterwards, the EV's DAA contract private key CC_{sk}^{DAA} is used to certify the previously generated session key pair. This certification uses the TPM's `TPM2_Certify` function, guaranteeing that the session key pair was generated with the claimed attributes and in the same TPM as the DAA key pair. We extend the protocol from [205] for the EV charging use case by

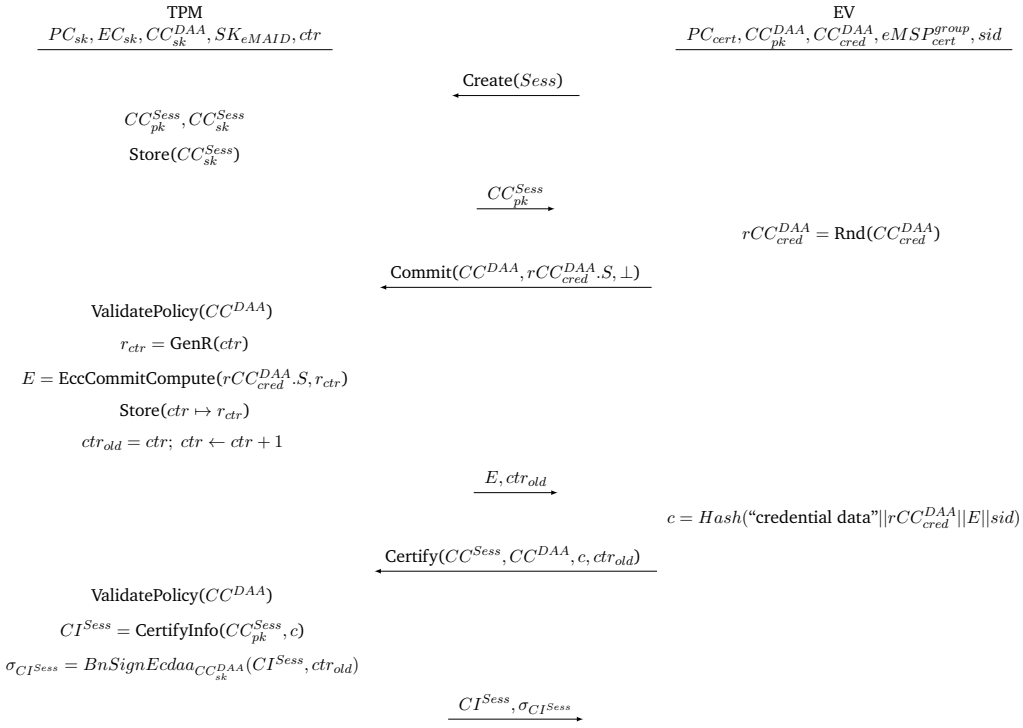


Figure 7.10.: Contract Credential Usage - Session Credential Certification

binding the certification to the current V2G communication session by including the ISO 15118 session ID in the qualifying data.

After the initialization phase, the EV sends its public session key CC^{Sess}_{pk} together with the data that is required to verify the authenticity of CC^{Sess}_{pk} to the CS. The CS either verifies the authenticity of CC^{Sess}_{pk} locally or via a request to its CSO (e.g., if the required eMSP root is not installed or if the CS does not support DAA). Afterwards, the EV can use CC^{Sess}_{sk} via its TPM for all ISO 15118 processes that require authenticity protection, i.e., for PnC charge authorization and for meter value signing. Furthermore, CC^{Sess}_{sk} can be used to sign billing-relevant charge session data in order to address ST_7. Additionally, SK_{eMAID} is used to generate the required tokens for the offline charge authorization method of Section 7.1.

Initialization

The process for preparing an EV's V2G session credentials is shown in Fig. 7.10. The EV starts by instructing its TPM to create the new session key pair ($CC^{Sess}_{pk}, CC^{Sess}_{sk}$). Afterwards, the EV randomizes its DAA contract credentials CC^{DAA}_{cred} resulting in a unique rCC^{DAA}_{cred} for the same DAA key, such that the randomized credential rCC^{DAA}_{cred} is still verifiable using the eMSP's public key $eMSP^{group}_{pk}$ (cf. Section 2.5.4). The S value[6] of rCC^{DAA}_{cred} is provided to the TPM with an unset ($\perp$) basename in the `TPM2_Commit` command as the first step of the DAA signing/certify process. The TPM generates a random value r_{ctr} for its current commit counter ctr and uses S and r_{ctr} to generate the commit value E. The TPM stores the mapping between ctr and r_{ctr}, increments ctr, and returns E together with the old value of the counter ctr_{old}.

[6] See [205] for details on the elements of a DAA credential.

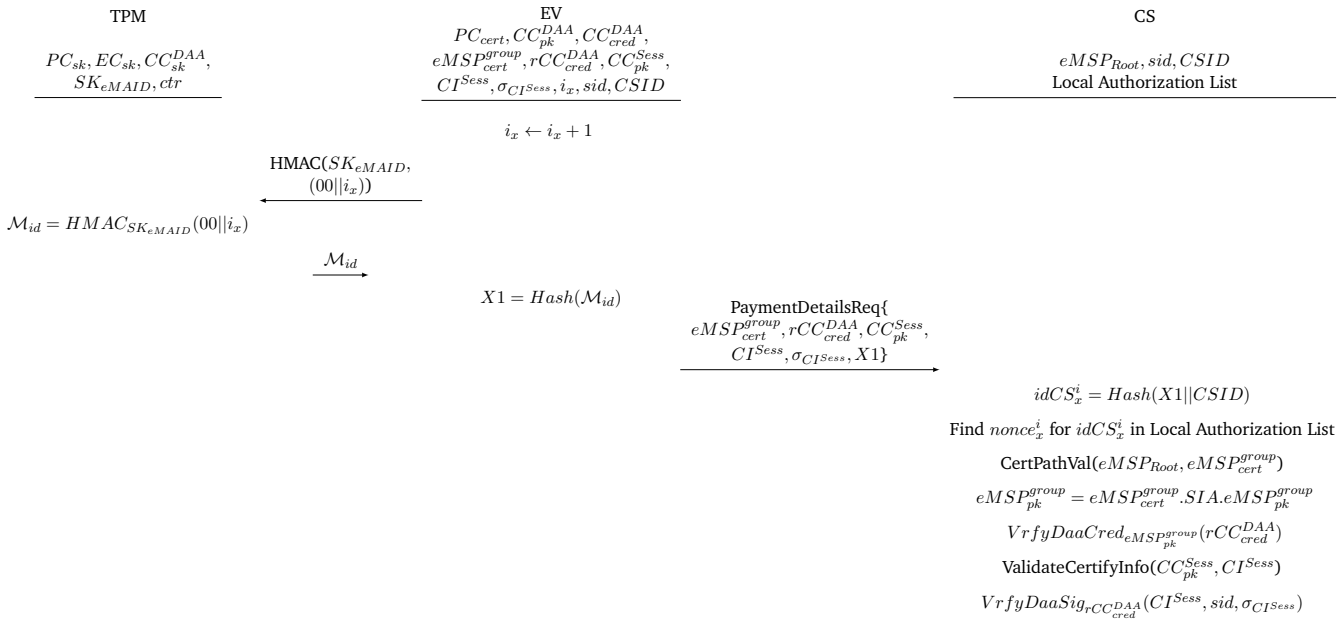


Figure 7.11.: Contract Credential Usage - DAA Authentication

For the second part of the DAA-based signing/certify process, the EV generates the qualifying data c as described in [205] with the addition of the current ISO 15118 V2G session ID sid. Since the integrity of the EV's software state is only validated before a charge authorization via the authorization policy of the CC^{DAA} key pair, an old CC^{Sess} with an appropriate certification data structure could be reused in a future ISO 15118 session in order to charge with a compromised EV (cf. ST_2 *AF1*) if certifications were not bound to a specific session.

Afterwards the `TPM2_Certify` command is called to certify CC^{Sess}_{pk} with CC^{DAA}_{sk}. Note that this command requires a correct assertion of the CC^{DAA} key pair's policy; hence, the EV can only get a certified session key pair if it booted into a trusted software state. In a first step, the TPM generates the certification data structure CI^{Sess} for CC^{Sess}_{pk}, including the qualifying data c and the *name* of the session key pair, i.e., a hash over its public area (with the attributes from Table 7.1 and the value of CC^{Sess}_{pk}). In a second step, the TPM uses CC^{DAA}_{sk} to sign CI^{Sess}. The signing process uses the same random value r_{ctr} as during the commit, which is identified based on the previous counter value ctr_{old}. CI^{Sess} and the generated signature $\sigma_{CI^{Sess}}$ are returned to the EV.

Authentication and Charge Authorization

The process of charge authorization is shown in Fig. 7.11. The EV starts by generating the $X1$ value as described in Section 7.1 whereby $hmac_{ID_x}(\cdot)$ is instantiated as an `TPM2_HMAC`, calculated by the TPM with SK_{eMAID} as ID_x. A manipulation of the counter i by the Local Adversary should be prevented, e.g., by implementing the counter in the TPM's protected NV memory (cf. Section 2.5.1; not shown in Fig. 7.11). Afterwards, the EV sends $eMSP^{group}_{cert}, rCC^{DAA}_{cred}, CC^{Sess}_{pk}, CI^{Sess}, \sigma_{CI^{Sess}}$, and $X1$ in its *PaymentDetailsReq* message to the CS (instead of the usual eMAID and contract certificate chain).

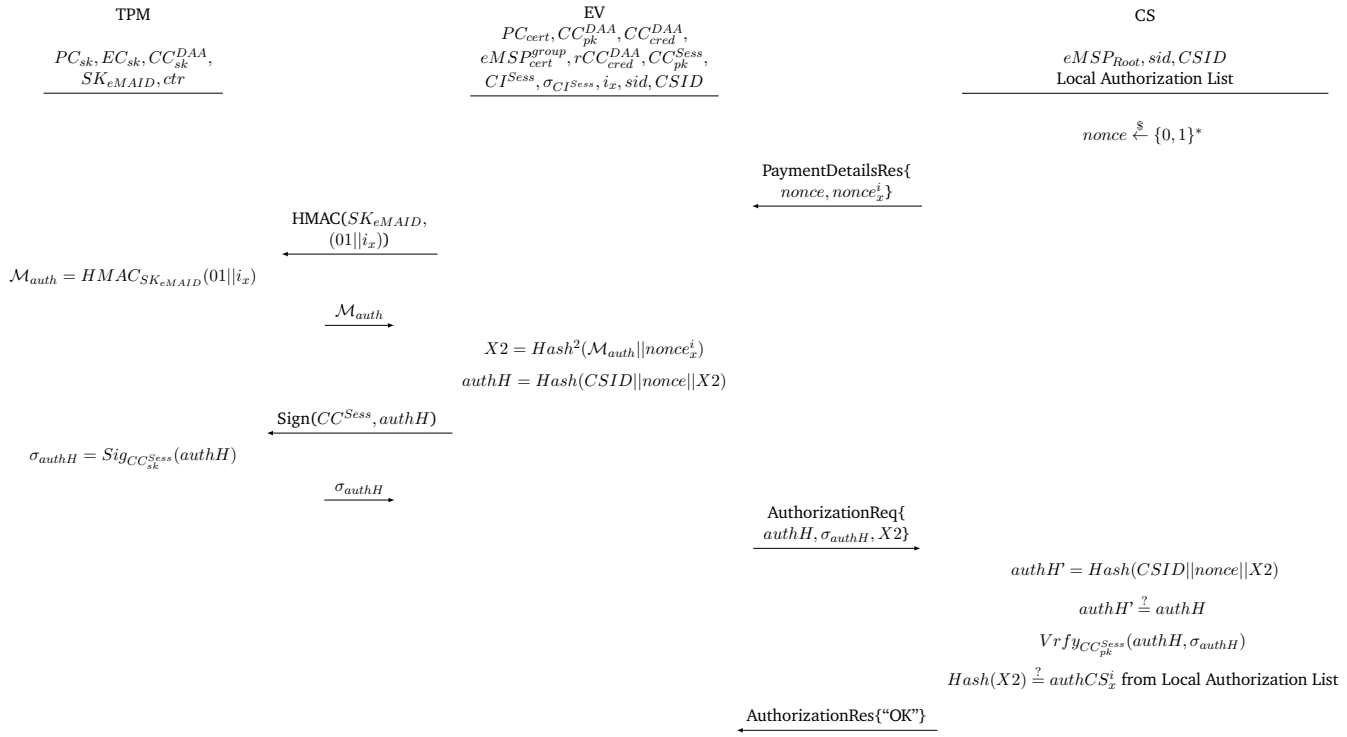


Figure 7.12.: Contract Credential Usage - Charge Authorization

The CS validates the EV's data in several steps. Note that a CSO may handle this validation procedure for their CSs (e.g., if a CS does not support DAA or does not have the required root installed). First, it uses $X1$ to find $nonce^i_x$ in its Local Authorization List. Second, it validates the authenticity of $eMSP^{group}_{cert}$ and that the certificate traces up to a valid eMSP root. Third, it requests the OCSP data for $eMSP^{group}_{cert}$ from its CSO to ensure that the certificate is not revoked (not shown in Fig. 7.11). Fourth, it verifies that rCC^{DAA}_{cred} is a valid credential that was originally issued (i.e., in its non-randomized form) by the eMSP owning the corresponding private key to $eMSP^{group}_{cert}$ (cf Section 2.5.4). Fifth, it validates that CC^{Sess}_{pk} corresponds to CI^{Sess} by calculating the expected *name* of CC^{Sess}_{pk} and checking that the expected *name* is equal to the *name* included in CI^{Sess}. Finally, the CS can verify the DAA signature $\sigma_{CI^{Sess}}$ over CI^{Sess} using rCC^{DAA}_{cred}. Note that the DAA signature verification process requires the CS to recompute the qualifying data c, which includes sid and thus the process fails if CC^{Sess}_{pk} was certified for a different session ID.

If all validations are successful, then the CS sends the usual V2G session $nonce$ together with the offline authorization $nonce^i_x$ to the EV in a *PaymentDetailsRes* message as shown in Fig. 7.12. The EV uses $nonce^i_x$ to build $X2$ as described in Section 7.1, again using an HMAC with SK_{eMAID} as $hmac_{ID_x}(\cdot)$. Additionally, the EV builds $authH$ as $Hash(CSID||nonce||X2)$ whereby $CSID$ is a unique identifier of the CS. $CSID$ is received by the EV in the subject name of the CS' certificate during the TLS handshake (not shown). Afterwards, $authH$ is signed by the EV's TPM with CC^{Sess}_{sk} and the resulting signature σ_{authH} is sent to the CS together with $authH$ and $X2$ in the EV's *AuthorizationReq* message.

After receiving the *AuthorizationReq* message, the CS verifies σ_{authH} using the previously received (and verified) CC^{Sess}_{pk}. Finally, the CS verifies the EV user's authorization to charge based on $X2$ either using its Local Authorization List or a request to the backend as described in Section 7.1. Later during the V2G session, the EV can sign meter receipts and other billing-relevant session data using CC^{Sess}_{sk} via its TPM (not shown).

8. Implementation

This section presents the proof-of-concept implementation of the privacy-preserving charge authorization and billing concept (Section 7), integrated into the respective EV charging protocols. Section 8.1 describes the implementation and Section 8.2 presents the measurements of the created overhead.

8.1. Proof-of-Concept Implementation

The combined method of privacy-preserving charge authorization and DAA-based PnC authentication was implemented as a proof-of-concept in order to show its feasibility and measure the resulting overhead. The implementation includes authorization whitelist management, offline- and online authorization, as well as PnC contract credential installation and usage (for DAA-based authentication and privacy-preserving authorization). Local EIM-based authorization, remote authorization/reservation requests, and credential pools are not implemented due to time constraints. Section 8.1.1 describes the setup of the proof-of-concept implementation and Section 8.1.2 describes the processes. The used message definitions are shown in Appendix B.

8.1.1. Proof-of-Concept Setup

The proof-of-concept implementation setup is shown in Fig. 8.1. The EV and CS are implemented on a Raspberry Pi 3 Model B board[1] each, i.e., single board computers with a “Quad Core 1.2GHz Broadcom BCM2837 64bit” CPU, 1GB of RAM, and a ”BCM43438 wireless LAN” uplink. The Raspberry Pi boards are running the “Raspbian GNU/Linux 9 (stretch)” operating system with Linux kernel version 4.14.66-v7+. The EV Pi is equipped with an Infineon Iridium board SLM 9670 TPM 2.0[2] as a hardware TPM implementation. The EV Pi and CS Pi are connected via PLC stamp micro 2 Evaluation boards[3] in order to emulate the PLC-based ISO 15118 communication channel.

The CSO, CCH, and eMSP are implemented as separate Virtual Machines (VMs) based on Oracle VM VirtualBox[4] version 6.1. For simplicity, the role of CPS is implemented as a separate process on the eMSP VM, i.e., the contract credential requests are decrypted and the corresponding responses are signed on the eMSP VM acting as CPS. The host machine (not shown in Fig. 8.1) is equipped with a “Hexa Core 3.3GHz Intel Core i7 5820K 64bit” CPU and 64 GB of RAM and is running the “Windows 10” version 2009 operating system. Each

[1] `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/` (visited on 2020-12-18)
[2] `https://www.infineon.com/cms/de/product/evaluation-boards/iridium-slm-9670-tpm2.0/` (visited on 2020-12-18)
[3] `https://in-tech-smartcharging.com/products/evaluation-tools/plc-stamp-micro-2-evaluation-board` (visited on 2020-12-18)
[4] `https://www.virtualbox.org/` (visited on 2020-12-18)

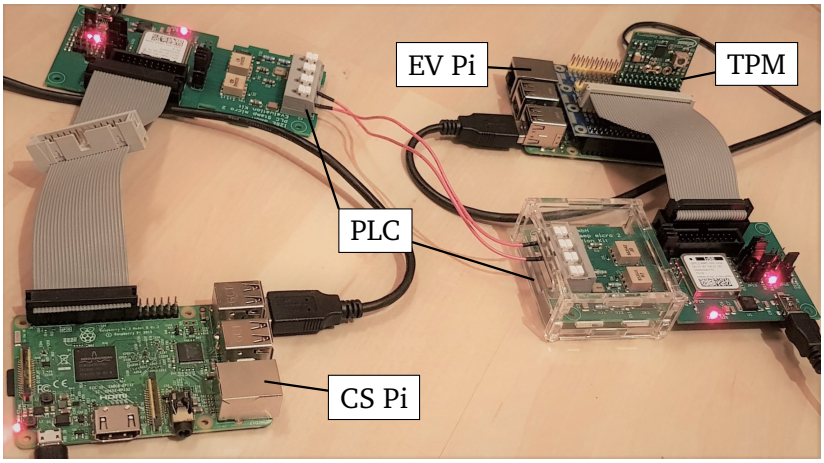


Figure 8.1.: Proof-of-Concept Implementation Setup

VM is configured with 6 cores and 16 GB of RAM and they are running the "Ubuntu 20.04.1 LTS" operating system with Linux kernel version 5.4.0-53-generic. The network interfaces of the VMs are configured as bridged adapters. In order to enable the OCPP communication between the CS Pi and the CSO VM and since CSs commonly use a wireless uplink [142], the CS Pi is connected to a WiFi access point in the network via its WLAN uplink.

The implementation of the ISO 15118 protocol is based on the open source Java reference implementation RISE V2G.[5] The RISE V2G implementation was adjusted such that it only loads key- and certificate stores once.[6] The implementation of the OCPP 2.0.1 protocol is based on the open source Python OCPP framework from *The Mobility House*.[7] The OCPI protocol is implemented in Python based on the `flask`[8] and `requests`[9] libraries for web service communication. Serialization and serialization of the OCPI JSON messages is implemented with the `orjson`[10] library. The authorization whitelists are implemented as MongoDB[11] (version 4.4.1) databases with an index[12] over the id_x^i field (cf. Section 7.1) and accessed from Python using the `pymongo`[13] library. All interactions between the host (the EV Pi) and the TPM are implemented in C with the TPM2-TSS.[14] The EV Pi's interactions with its TPM are running in parallel to the ISO 15118 process in order to pre-calculate steps (e.g., key authorization policies or key generations) whenever possible. The resulting measurements are representative of an optimized integration of the solution into existing charging protocol

[5] `https://github.com/V2GClarity/RISE-V2G/` (visited on 2020-11-02)
[6] The stores are kept in RAM after their first load for subsequent accesses, as we found that the loading of these stores caused a significant overhead in both the default implementation as well as in our modified implementation. In fact, as the default implementation loads theses stores more often, our modified implementation significantly outperformed it in most processes based on preliminary measurements. Hence, this change was required in order to accurately gauge the additional overhead of our proposed solution. The reported times for the default implementation (as baseline) and for our modified implementation in Section 8.2 both include this change. The only other changes to the default implementation are those needed for performance measurements.
[7] `https://github.com/mobilityhouse/ocpp` (visited on 2020-11-07)
[8] `https://flask.palletsprojects.com/en/1.1.x/` (visited on 2020-11-11)
[9] `https://requests.readthedocs.io/en/master/` (visited on 2020-11-11)
[10] `https://github.com/ijl/orjson` (visited on 2020-11-11)
[11] `https://www.mongodb.com/` (visited on 2020-11-11)
[12] `https://docs.mongodb.com/manual/indexes/` (visited on 2020-11-11)
[13] `https://github.com/mongodb/mongo-python-drivern` (visited on 2020-11-11)
[14] `https://github.com/tpm2-software/tpm2-tss` (visited on 2020-10-24)

processes. Inter-process communication is implemented using named pipes.[15] The implementation of all DAA calculations that do not involve the TPM (the host's operations during DAA signature creation, the verification of DAA signatures, and the creation of DAA credentials by the eMSP as issuer) are based on the C++ open source implementation from [205].[16]

8.1.2. Proof-of-Concept Processes

The processes of the proof-of-concept implementation can be divided into authorization whitelist management, contract credential installation, and charge authorization. The following sections describe the implemented processes.

Whitelist Management

The initial generation of an eMSP's authorization whitelist is shown as Python-style pseudocode in Listing 8.1. The eMSP generates the symmetric keys ID_x (or SK_{eMAID} in for PnC) for a set amount of users (x) and calculates the id_x^i and $auth_x^i$ values for a set amount of authorizations (i) as described in Section 7.1. The generation of these values is parallelized using the `Pool` class (initialized with eight worker processes) of the Python `multiprocessing`[17] library (not shown in Listing 8.1). The results are stored in the eMSP's MongoDB with a unique index over the id_x^i field.

```
idT_List = []
for x in range(NUM_USERS):
  ID_x = generateNewSymKey() # == SK_EMAID in case of PnC
  idT = {'ID_x': ID_x, 'auth': []}
  for i in range(NUM_AUTHS):
    id_ix = SHA256(HMAC(ID_x, '\x00' || i))
    auth_ix = HMAC(ID_x, '\x01' || i)
    idT['auth'].append({'i': i, 'id_ix': id_ix, 'auth_ix': auth_ix})
  idT_List.append(idT)
mongoDB.create_index('auth.id_ix', name='auth.id_ix', unique=True)
mongoDB.insert_many(idT_List)
```

Listing 8.1.: Whitelist Generation at eMSP

The distribution of the whitelist is started by the CSO since, in OCPI, lists of ID Tokens have to be pulled by the receiver and a push is only possible for single data elements (cf. [142], Sections 12). When the eMSP receives a request for a whitelist, they load up to n next unused authorization values $\langle id_x^i, auth_x^i \rangle$ for their users from the MongoDB using an aggregation query[18] as shown in Listing 8.2. The resulting list is shuffled and sent over OCPI using the existing *Token* message (cf. [142], Sections 12), whereby id_x^i is transferred in place of the usual UID and $auth_x^i$ is transferred in place of the usual eMAID. Both values are encoded as Base64 strings.

```
query = {'auth.used': {'$ne': True}}
project = {'_id': 0, 'auth': {'id_ix': 1, 'auth_ix': 1}}
match={'auth.i': {'$in': range(n)}}
```

[15] `https://man7.org/linux/man-pages/man7/fifo.7.html` (visited on 2020-11-02)
[16] `https://github.com/UoS-SCCS/ecc-daa` (visited on 2020-10-24)
[17] `https://docs.python.org/3/library/multiprocessing.html` (visited on 2020-12-18)
[18] `https://docs.mongodb.com/manual/aggregation/` (visited on 2020-12-18)

```
result = mongoDB.aggregate([{'$match': query}, {'$unwind': '$auth'}, {'$match': match}, {'
    $project': project}])
```

Listing 8.2.: Whitelist Distribution at eMSP

The CSO stores the values in its MongoDB and transforms them for offline authorization at the CS according to Section 7.1 and as shown in Listing 8.3. The calculations are again parallelized using the `Pool` class (initialized with eight worker processes) of the Python `multiprocessing` library (not shown in Listing 8.3). The transformed values are sent in Base64 encoding using the existing OCPP *SendLocalListRequest* message (cf. [148], Section D.2) as ID Tokens (cf. Listing 8.3). The existing *idToken* field is used to transmit the nonce as this field is required and limited to 36 characters via the OCPP JSON schema, i.e., only the 16 byte nonce fits. The other values are sent in a *customData* field, i.e., an optional field that exists in the OCPP JSON schema of all data types and can be used to transmit arbitrary data (cf. [148], Section P.1).

```
for t in Tokens:
  idCS_ix = SHA256(t['id_ix'] || CSID)
  nonce_ix = generateNewNonce()
  authCS_ix = HashN(t['auth_ix'] || nonce_ix, N=3)
  'idToken': {
    'idToken': nonce_ix,
    'type': 'Local',
    'customData':{
      'vendorId': CSO_ID,
      'idCS_ix': idCS_ix,
      'authCS_ix': authCS_ix,
      'IDeMSP': EMSP_ID}}
```

Listing 8.3.: Whitelist Generation Offline Authorization

The query to find an entry in the whitelist in response to an EV's authorization request is shown Listing 8.4 on the example of the eMSP's whitelist. The query uses the *findOneAndUpdate*[19] method searching for the provided id_x^i under the condition that the entry is not marked as used. If the entry is found it is marked as used and returned. The returned entry $\langle id_x^i, auth_x^i \rangle$ can be used to verify the user's authorization request as described in Section 7.1.

```
query = {'auth.id_ix': id_ix }
projection = {'_id': 0, 'auth': {'$elemMatch': {'id_ix': id_ix, 'used': {'$ne': True}}}}
idT = mongoDB.find_one_and_update(filter=query, projection=projection, update={'$set': {'
    auth.$.used': True}})
```

Listing 8.4.: Whitelist Query at eMSP

Contract Credential Installation

For the installation of contract credentials, the EV Pi is initialized as shown by the pseudocode in Listing 8.5. First, the endorsement credential key pair is generated in the EV's TPM as a primary key based on the EC template (cf. Table 7.1). Afterwards, the authorization policy digest, i.e., a `TPM2_PolicyAuthorize` for the OEM's public key, is generated. The provisioning key pair is generated underneath the EC key, based on the PC template (cf. Table 7.1), and sealed to the authorization policy digest. The generation of the PC key pair requires usage authorization for the EC key, which, in accordance with [191] and as mentioned in

[19] `https://docs.mongodb.com/manual/reference/method/db.collection.findOneAndUpdate/` (visited on 2020-12-18)

Section 7.2.1, is bound to the endorsement hierarchy authorization. For the proof-of-concept implementation, the endorsement hierarchy authorization is set to a password.

The provisioning certificate for the public provisioning key, including the public EC key and the authorization policy digest (cf. Section 7.2.1), is generated and signed by the OEM. Afterwards, the `TPM2_PolicyPCR` policy digest for the EV's expected firmware state is generated and signed by the OEM in order to authorize its usage in the `TPM2_PolicyAuthorize` command. For the proof-of-concept implementation, the EV's firmware is emulated by a single file on the EV Pi. The measured boot process (cf. Section 2.5.2) is emulated by extending the file into the TPM's PCR. The results of the EV initialization are stored on the EV Pi. The EC key pair is permanently stored in the TPM via the `TPM2_EvictControl` command (cf. Section 2.5.1).

```
/* Provisioning credential generation */
EC = TPM2_CreatePrimary(EC_Template);
authPolicyOEM = generateAuthPolicy(OEM.pub);
sessionEC = TPM2_StartAuthSession(TPM2_SE_POLICY);
sessionEC = TPM2_PolicySecret(sessionEC, TPM_RH_ENDORSEMENT);
PC = TPM2_Create(EC, sessionEC, PC_Template, authPolicyOEM);
PC_Cert = generateCertificate(OEM.priv, PC.pub, EC.pub, authPolicyOEM);

/* Policy PCR generation */
polDigPCR = generatePolicyPCRDigest(firmwareData);
polSigPCR = sign(OEM.priv, polDigPCR);

/* Store results */
TPM2_EvictControl(EC);
storePC(PC, PC_Cert);
storePolPCR(polDigPCR, polSigPCR);
```

Listing 8.5.: EV Initialization

For certificate installation via ISO 15118, the EV generates an installation request as shown in Listing 8.6 (the process of loading key pairs into the TPM is not shown). First the contract credential DAA key pair is generated underneath the EC key, based on the CC^{DAA} template (cf. Table 7.1), and sealed to the authorization policy digest. The generation of the CC^{DAA} key pair requires usage authorization for the EC key as previously described for the process in Listing 8.5. The EV builds a contract certificate request by concatenating its provisioning certificate and the public CC^{DAA} key.

The contract certificate request is signed with the private provisioning key in the EV's TPM. Usage of the private provisioning key requires a correct assertion of its policy (cf. Fig. 7.5), i.e., the signature over the `TPM2_PolicyPCR` policy digest is verified with the public OEM key in the TPM. Afterwards, a policy session is started and extended with the `TPM2_PolicyPCR` command. Finally, the `TPM2_PolicyAuthorize` command is executed and the resulting policy can be used to authorize a signature creation with the private provisioning key.

The EV encrypts its contract certificate request (appended with the signature) based on the CPS' public key. For this, CPS' certificate is received over ISO 15118 in the CS' *ServiceDiscoveryRes* message (cf. *SACertChain* in Listing B.1). The EV validates the certificate based on its locally installed V2G root and uses the public key from the CPS' certificate together with a newly generated ephemeral ECDH private key in the ECDH algorithm to generate a symmetric key. The symmetric key is used to encrypt the contract certificate request and signature using Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM). The encrypted request and the ephemeral ECDH public key are sent in the EV's *CertificateInstallationReq* message (cf. Listing B.2) to the CS (the Initialization Vector (IV) and authentication tag of the AES GCM encryption are included as part of the encrypted request).

```
/* Contract credential request generation */
sessionEC = buildPolicyEC();
CC_DAA = TPM2_Create(EC, sessionEC, CC_DAA_Template, authPolicyOEM);
certReq = buildCertificateRequest(PC_Cert, CC_DAA.pub);

/* Contract credential request signing */
certReqDig = SHA256(certReq);
OEM_Pub = TPM2_LoadExternal(OEM.pub);
valTicket = TPM2_VerifySignature(OEM_Pub, polDigPCR, polSigPCR);
session = TPM2_StartAuthSession(session, TPM2_SE_POLICY);
session = TPM2_PolicyPCR(session, PCR_Selection);
session = TPM2_PolicyAuthorize(session, polDigPCR, Name(OEM_Pub), valTicket);
certReqSig = TPM2_Sign(PC, session, certReqDig);
signedCertReq = (certReq || certReqSig);

/* Contract credential request encryption */
certPathVal(V2G_Root, CPS_Cert);
ECDHe = generateECKeyPair();
symKey = ECDH(CPS_Cert.subjPubKey, ECDHe.priv);
IV = generateRandomIV();
encryptedCertReq = AES_GCM_Enc(symKey, IV, signedCertReq);

/* Contract credential request sending */
certificateInstallationReq = CertInstallReq(encryptedCertReq, ECDHe.pub);
sendToCS(certificateInstallationReq);
```

Listing 8.6.: Contract Credential Request Generation

The CS forwards the request to the CSO using the existing OCPP message *Get15118EVCertificateRequest* (cf. [148], Section M.4). As this message simply transfers the EXI encoded ISO 15118 request message as Base64 string, no changes are required. The CSO forwards the request to the CPS using a new OCPI message that simply transfers the EXI request data as Base64 string. The CPS decrypts the request, validates the provisioning certificate based on the OEM root, verifies the request signature based on the public key from the provisioning certificate, and forwards the request to the eMSP.

The eMSP generates contract credentials for the EV as shown in Listing 8.7. The eMSP starts by generating the ECC-DAA credential for the public DAA key form the EV's request and encrypts the credential using a newly generated symmetric key ("symKey" in Listing 8.7). Afterwards, this symmetric key is encrypted using the TPM credential protection mechanism (cf. Section 2.5.3). For this, the eMSP generates a new ephemeral ECDH private key and uses this private key together with the EV's EC public key in the ECDH algorithm to generate a shared secret ("seedCred"). The shared secret is used in a KDF ("KDFa"; cf. [195], Section 11.4.9) to generate a symmetric encryption key ("symKeyCred") and a symmetric HMAC key ("symKeyCredHMAC"). The symmetric encryption key is used to encrypt[20] the previously generated "symKey" and the symmetric HMAC key is used to calculate an HMAC over the resulting ciphertext. The resulting HMAC output appended with the encrypted "symKey" can be decrypted by the EV's TPM using the `TPM2_ActivateCredential` command with the EC private key.

Additionally, the eMSP generates the symmetric eMAID key and encrypts it for the EV's TPM with the EC public key. The encryption process uses the duplicate protection mechanism (cf. Section 2.5.1) and, similar to the credential protection mechanism, requires the generation of a new ephemeral ECDH private key as well as

[20] Encryption for the credential protection mechanism always uses Cipher Feedback Mode (CFB) (cf. [195], Section 24.4).

the derivation of a symmetric encryption key and a symmetric HMAC key. The symmetric keys are used to encrypt[21] the eMAID key and calculate an HMAC over the resulting ciphertext respectively.

The eMSP sends the credential installation response data to the CPS. This data includes the eMSP's certificate with its public group key (cf. Section 7.4b), the eMAID key's public attributes (cf. Table 7.1), the encrypted eMAID key, the encrypted "symKey" data, the encrypted DAA credential, and both ephemeral ECDH public keys.

```
/* Generate and encrypt DAA credential */
CC_DAA_Cred = generateDAACredential(CC_DAA_Pub, EMSP_GroupKey);
symKey = generateNewSymKey();
IV = generateRandomIV();
encryptedDAACred = AES_GCM_Enc(symKey, IV, CC_DAA_Cred);

/* Encrypt symKey with 'Credential Protection' using EC_Pub */
EC_Pub = PC_Cert.SIA.EC_Pub;
authPolicyOEM = PC_Cert.SIA.authPolicyOEM;
CC_DAA_Pub_Name = Name(CC_DAA_Pub, authPolicyOEM);
ECDHeCred = generateECKeyPair();
seedCred = ECDH(EC_Pub, ECDHeCred.priv);
symKeyCred = KDFa(seedCred, 'STORAGE', CC_DAA_Pub_Name);
encryptedSymKey = AES_CFB_Enc(symKeyCred, 0, symKey);
symKeyCredHMAC = KDFa(seedCred, 'INTEGRITY', NULL);
encryptedSymKeyHMAC = HMAC(symKeyCredHMAC, encryptedSymKey || CC_DAA_Pub_Name);
credentialSymKey = (encryptedSymKeyHMAC || encryptedSymKey);

/* Generate SK_EMAID and encrypt it for TPM2_Import() using EC_Pub */
SK_EMAID = generateNewSymKey();
SK_EMAID_Name = Name(SK_EMAID);
ECDHeImport = generateECKeyPair();
seedImport = ECDH(EC_Pub, ECDHeImport.priv);
symKeyImport = KDFa(seedImport, 'STORAGE', SK_EMAID_Name);
encryptedSK_EMAID = AES_CFB_Enc(symKeyImport, 0, SK_EMAID);
symKeyImportHMAC = KDFa(seedCred, 'INTEGRITY', NULL);
encryptedSK_EMAID_HMAC = HMAC(symKeyImportHMAC, encryptedSK_EMAID || SK_EMAID_Name);
duplicatedSK_EMAID = (encryptedSK_EMAID_HMAC || encryptedSK_EMAID);

/* Send credential installation response to CPS */
SK_EMAID_Pub = PublicAttributes(SK_EMAID);
credInstallationData = credInstallData(EMSP_Group_Cert, SK_EMAID_Pub, duplicatedSK_EMAID,
   ECDHeImport.pub, credentialSymKey, ECDHeCred.pub, encryptedDAACred);
sendToCPS(signedInstallationData);
```

Listing 8.7.: Contract Credential Generation

The CPS signs the eMSP's data and builds the *CertificateInstallationRes* message (cf. Listing B.3).[22] The message is forwarded to the CSO using a new OCPI message that simply transfers the EXI encoded response data as Base64 string. The CSO forwards the response to the CS using the existing OCPP *Get15118EVCertificateResponse* message (cf. [148], Section M.4). As this message simply transfers the EXI encoded ISO 15118 response message as Base64 string, no changes are required. Finally, the CS forwards the EXI encoded ISO 15118 response message to the EV.

[21] Encryption for the duplicate protection mechanism always uses CFB (cf. [195], Section 23.3.2.3).

[22] The signature is implemented using the XML signature mechanism (cf. [13]) as this mechanism is currently employed by ISO 15118 for its application layer signatures (cf. [97], Section 7.9.2.4.2; as mentioned in Section 2.4.1).

The EV handles the certificate installation response as shown in Listing 8.8. The EV starts by verifying the CPS' signature. Afterwards, the EV instructs its TPM to decrypt the encrypted symmetric key ("symKey" from Listing 8.7) that was used to encrypt the DAA credential with its EC private key using the `TPM2_ActivateCredential` command. The `TPM2_ActivateCredential` command requires usage authorization for the EC key as previously described for the process in Listing 8.5. The resulting symmetric key is used to decrypt the DAA credential and the EV verifies the credential with the eMSP's public group key. Finally, the EV imports the encrypted eMAID key into its TPM using the `TPM2_Import` command. The `TPM2_Import` command again requires usage authorization for the EC key as previously described for the process in Listing 8.5. The eMAID key is imported into the TPM's hierarchy underneath the EC key (cf. Section 2.5.1).

```
/* Decrypt CC_DAA_Cred based on TPM2_ActivateCredential */
sigInstallData = certificateInstallationRes.signedInstallationData;
Verify(CPS_Cert.subjPubKey, sigInstallData, certificateInstallationRes.signature);
sessionEC = buildPolicyEC();
symKey = TPM2_ActivateCredential(EC, CC_DAA, sessionEC, sigInstallData.credentialSymKey,
   credentialSymKey.ECDHeCred_Pub);
CC_DAA_Cred = AES_GCM_Dec(symKey, sigInstallData.encryptedDAACred);
Verify(sigInstallData.EMSP_Group_Cert.groupPubKey, CC_DAA_Cred);

/* Decrypt SK_EMAID based on TPM2_Import */
sessionEC2 = buildPolicyEC();
SK_EMAID = TPM2_Import(EC, sessionEC2, sigInstallData.SK_EMAID_Pub, sigInstallData.
   duplicatedSK_EMAID, sigInstallData.ECDHeImport_Pub);
```

Listing 8.8.: Contract Credential Response Hanlding

Charge Authorization

The process of an EV starting a charge authorization, i.e., building a *PaymentDetailsReq* message for the CS (cf. Section 7.2.5), is shown in Listing 8.9. The EV starts by instructing its TPM to generate a session key pair based on the CC^{Sess} template (cf. Table 7.1) underneath the EC key. The generation of the CC^{Sess} key pair requires usage authorization for the EC key as previously described for the process in Listing 8.5. Afterwards, the EV randomizes its DAA credential and instructs its TPM to calculate the commit value and counter, providing the CC^{DAA} key, part of the randomized credential, and no basename. The authorization policy session for the CC^{DAA} key is built the same way as described in Listing 8.6, i.e., the same way as for signature generation with the private PC key.

The EV certifies the session key pair with its CC^{DAA} key as described in Section 7.2.5. The EV starts by calculating the qualifying data c including a fixed string, the randomized DAA credential, the commit value, and the current ISO 15118 V2G session ID. The qualifying data is used together with the commit counter in order to certify the CC^{Sess} key pair based on an ECC-DAA signature using the private CC^{DAA} key. The authorization policy session for the CC^{DAA} key is again built the same way as described in Listing 8.6.

The EV uses its TPM to calculate $X1$ as described in Section 7.2.5, whereby the counter i is implemented via the TPM's NV memory (not shown in Listing 8.9), and builds its *PaymentDetailsReq* message (cf. Listing B.4). The *PaymentDetailsReq* message includes the eMSP's certificate (received during credential installation), the randomized DAA credential, public CC^{Sess} key, the certification data structure for the CC^{Sess} key, the DAA signature over the certification data structure, and the $X1$ value. The *PaymentDetailsReq* message is sent to the CS.

```
/* Generate session key pair */
sessionEC = buildPolicyEC();
CC_Sess = TPM2_Create(EC, sessionEC, CC_Sess_Template);

/* Randomize credential and calculate commit value */
rCC_DAA_Cred = randomize(CC_DAA_Cred);
sessionAuth = buildPolicyAuth(); // Policy for PC or CC_DAA
E, counter = TPM2_Commit(CC_DAA, sessionAuth, rCC_DAA_Cred.S);

/* Certify session key pair */
bufferC = ('credential data' || rCC_DAA_Cred || E || V2G_SessionID);
c = SHA256(bufferC);
sessionAuth2 = buildPolicyAuth(); // Policy for PC or CC_DAA
CI_Sess, CI_Sig = TPM2_Certify(CC_Sess, CC_DAA, sessionAuth2, c, counter);

/* Generate and send PaymentDetailsReq */
M_id = TPM2_HMAC(SK_EMAID, 0x00 || ++i);
X1 = SHA256(M_id);
paymentDetailsReq = PaymentDetailsReq(EMSP_Group_Cert, rCC_DAA_Cred, CC_Sess.pub, CI_Sess,
   CI_Sig, X1);
sendToCS(paymentDetailsReq);
```

Listing 8.9.: Payment Details Request Generation

The CS verifies the *PaymentDetailsReq* data as described in Section 7.2.5. That is, the CS validates the eMSP certificate based on an available eMSP root. The eMSP's public group key is used to verify the EV's randomized DAA credential, the randomized DAA credential is used to verify the signature over certification data structure, and the certification data structure is used to validate the public CC^{Sess} key. If $X1$ is available in the CS' local whitelist, the pre-generated nonce is used for offline authorization and otherwise, a fresh nonce is generated for online authorization (cf. Section 7.1). The CS builds its *PaymentDetailsRes* message (cf. Listing B.5) including the authorization nonce and a fresh challenge.

The EV calculates its authorization value $X2$ based on the CS' authorization nonce ("authNonce" called $nonce_x^i$ in Section 7.2.5) as shown in Listing 8.10 and described in Section 7.2.5. Afterwards, the EV builds the $authH$ value and signs it using the private CC^{Sess} key via its TPM. In the proof-of-concept implementation, this signature is implemented using the XML signature mechanism (cf. [13]) as this mechanism is currently employed by ISO 15118 for its application layer signatures (cf. [97], Section 7.9.2.4.2; as mentioned in Section 2.4.1). For this, the EV first builds the *AuthorizationReq* message (cf. Listing B.6) including the data to be signed (the $CSID$, the CS' "challenge" called $nonce$ in Section 7.2.5, and $X2$; cf. Section 7.2.5). The message is EXI encoded before it is hashed and the hash is used to build an XML *SignedInfo* element. The *SignedInfo* element is then again EXI encoded before it is hashed and the resulting hash is signed by the EV's TPM using the private CC^{Sess} key. The signature is placed in the header of the *AuthorizationReq* message before it is sent to the CS.

```
/* Calculate authorization value */
M_auth = TPM2_HMAC(SK_EMAID, 0x01 || i);
X2 = SHA256(SHA256(M_id || authNonce));

/* Generate and send AuthorizationReq */
authorizationReq = AuthorizationReq(CSID, challenge, X2);
authH = SHA256(EXI(XMLSignedInfo(SHA256(EXI(authorizationReq))))); // For XML Signature
authH_Sig = TPM2_Sign(CC_Sess, authH);
sendToCS(authH_Sig, authorizationReq);
```

Listing 8.10.: Authorization Request Generation

The CS verifies the signature using the previously received public CC^{Sess} key. Afterwards, the CS either (if possible) authorizes the charge session offline or sends an online authorization request over OCPP to the CSO using the existing OCPP message *AuthorizeRequest* (cf. [148], Section C.2) including the new data via a *customData* field as described for the process in Listing 8.3 (including $X1$ instead of $idCS_x^i$ and $X2$ instead of $authCS_x^i$). Similarly, the CSO and CCH either authorize the session based on their whitelists or forward the request over OCPI using the existing *Token* message (cf. [142], Sections 12) as described for whitelist distribution. The eMSP either authorizes or denies the session (cf. Section 7.1.2).

8.2. Measurements

The proposed solution for privacy-preserving charge authorization and billing creates additional overhead in the processes of whitelist management, credential installation, and charge authorization. The following sections describe the measured overhead of the proof-of-concept implementation in comparison to the existing methods of the EV charging protocols.

For the whitelist management functions, we assume a scenario with 10,000,000 EV users (Germany's goal for the number of registered EVs by 2030;[23] note that with country-specific whitelists, charging in a different country would always require an online authorization request). Additionally, we assume that every EV user has a contract with an eMSP and that EV users are distributed evenly across 36 eMSPs,[24] i.e., about 277,777 EV users per eMSP. Furthermore, we assume that EV's are charged once a day on average (cf. [176]). The generation of whitelist values uses SHA256 (32 bytes output) as a hash function and as basis for the HMAC as it is considered secure by current standards (e.g., [12]).

8.2.1. Communication and Storage Overhead

The solution creates communication and storage overhead at several data flows and stores. The reported sizes of ISO 15118 messages correspond to the byte size of their EXI encoded form. The results are summarized in Table 8.1.

Authorization whitelists need to be distributed regularly and the included values are larger than the usual UID/eMAID. Given the assumed scenario, an eMSP's whitelist (including the $\langle id_x^i, auth_x^i \rangle$ values; together 64 bytes) for one week of charge authorizations, i.e., 7 values per user, would be 124.4 MB large. The size of the resulting OCPI whitelist message is 536.7 MB as the values are Base64 encoded and as additional overhead is added through the mandatory fields in the OCPI Token data type like the issuer name, country code, and party ID, which are added to every token. An CSO's transformed list to a CS (including the $\langle idCS_x^i, nonce_x^i, authCS_x^i \rangle$ values; together 80 bytes) for one eMSP and one week of charge authorization would be 155.5 MB large. The size of the resulting OCPP whitelist is 556.1 MB as the values are again Base64 encoded and as additional overhead is added through mandatory fields like the token type and the vendor ID (as shown in Listing 8.3). In practice, whitelists could further be divided by region in order to reduce the overhead, assuming that the resulting anonymity set size remains large enough.[25] The communication and

[23] `https://www.bundesregierung.de/breg-en/issues/climate-action/klimaschonender-verkehr-1795842` (visited on 2020-12-23)

[24] At the time of writing, the number of registered German eMSP IDs is 719 (cf. `https://bdew-codes.de/Codenumbers/EMobilityId/ProviderIdList`; visited on 2020-12-23). In the assumed scenario, about 5% of these eMSPs cover all users in order to obtain an estimate of the load on larger eMSPs.

[25] An analysis of the corresponding sizes is beyond the scope of this thesis.

storage overhead of authorization whitelists in current EV charging protocols is argued to be negligible as lists do not need to be updated regularly[26] and only include one value per user.

The increased size of authorization values also affects all authorization request messages. Authorization requests need to include $X1$, ID_{eMSP}, $X2$, and $nonce_x^i$ (together 85 bytes assuming a 5 byte ID_{eMSP} as defined in ISO 15118; cf. country code and provider ID in [97], Section H.1) compared to the usual RFID UID (4 or 7 byte; cf. [110, 61]) or the eMAID (15 byte; cf. [97], Section H.1).

Certificate installation requests are larger due to the certificate extension in the provisioning certificate and the encryption for the CPS. This change also affects the provisioning certificate pool and the EV's storage. In order to enable the encryption of the certificate requests, overhead is further added in the *ServiceDiscoveryRes* message by the inclusion of the CPS_{cert} certificate chain. The modified *CertificateInstallationReq* messages are 952 bytes large, whereby the provisioning certificate alone is 641 bytes. The modified *ServiceDiscoveryRes* messages are 1530 bytes large, whereby the CPS_{cert} certificate chain alone is 1421 bytes. For comparison, the size of *CertificateInstallationReq* messages in the default RISE V2G implementation is 811 bytes (with a 471 byte provisioning certificate) and the size of *ServiceDiscoveryRes* messages is 101 bytes. In order to reduce the overhead of sending the CPS_{cert} certificate chain for encryption, the chain could alternatively be requested by an EV via the optional *ServiceDetailReq* message for the certificate installation service and sent by the CS in its corresponding *ServiceDetailRes* message. While this method requires the exchange of an additional message pair if a certificate installation is required, it reduces the size of *ServiceDiscoveryRes* messages by 1429 bytes if no certificate installation is required (which is expected to be the more common case).

The size of certificate installation responses is increased due to the included eMSP certificate with the public group key extension and due to the larger contract credential data (encrypted DAA credential, encrypted eMAID key, etc.). This change also affects the contract certificate pool and the EV's storage. The modified *CertificateInstallationRes* messages are 4633 bytes large, whereby eMSP's certificate alone is 863 bytes and the contract credential data is 769 bytes. For comparison, the size of *CertificateInstallationRes* messages in the default RISE V2G implementation is 3637 bytes with a 472 byte eMSP CA certificate and 599 bytes of contract credential data.

Note that ISO 15118 limits the size of certificates to 800 bytes (cf. [97], Requirement V2G2-010). Hence, for standard compliance, the requirement would have to be changed or an eMSP's public group key would have to be transferred differently, e.g., in a separate certificate or via a different kind of certificate extension. For example, by including the public group key via a custom extension (as allowed by RFC 5280; cf. [35], Section 4.2) that only consists of an OID and the public group key in byte form, the certificate is 755 bytes large.

Additionally, OCPP limits the size of Base64 encoded ISO 15118 *CertificateInstallationRes* messages to 5600 characters via its JSON schema [148]. A 4633 byte EXI message, however, is encoded to 6180 Base64 characters and thus does not fit into OCPP's limit. In order to address this incompatibility, the CPS_{cert} certificate chain (1421 bytes as mentioned previously) could be omitted from *CertificateInstallationRes* messages if the corresponding request was encrypted for the CPS, i.e., if the EV already received the certificate chain in a previous message, assuming that the same CPS key pair is used for en-/decryption and signatures. Alternatively, in order to allow the use of different keys for en-/decryption and signatures, the *CertificateInstallationRes* message could only include the CPS's leaf certificate (at most 800 bytes due to the ISO 15118 limit) as long as the leaf certificate is issued by the same Sub-CA from the previously received CPS_{cert} certificate chain.

[26] Each user's ID Token only has to be generated once by an eMSP and distributed once to each CSO. Updates are only necessary if values have changed, e.g., if new users are added or users are removed.

Table 8.1.: Communication Overhead

Process	Message size	
	Normal	Proposed
Whitelist Backend	*Negligible*	124.4 MB per eMSP per week (536.7 MB OCPI data)
Whitelist CS	*Negligible*	155.5 MB per eMSP per week (556.1 MB OCPP data)
Authorization Request Values	4 - 15 bytes	85 bytes
ServiceDiscoveryRes	101 bytes	1530 bytes
CertificateInstallationReq	811 bytes	952 bytes
CertificateInstallationRes	3637 bytes	4633 bytes
PaymentDetailsReq	1452 bytes	1973 bytes
PaymentDetailsRes	51 bytes	55 bytes
AuthorizationReq	308 bytes	355 bytes

The size of *PaymentDetailsReq* messages is increased due to the increased size of the eMSP certificate, the included randomized DAA credential, the included $X1$, and the included public session key along with its certification data and -signature. This change also affects the OCPP *AuthorizeRequest* message if the CS forwards the data to the CSO for verification. The modified *PaymentDetailsReq* messages are 1973 bytes large, whereby the randomized DAA credential is 382 bytes,[27] the $X1$ value is 32 bytes, and the session key data is 291 bytes. For comparison, the size of *PaymentDetailsReq* messages in the default RISE V2G implementation (with the normal contract certificate chain and eMAID) is 1452 bytes.

The size of *PaymentDetailsRes* messages is increased to 55 bytes due to the additional authorization nonce, compared to its size of 51 bytes in the default RISE V2G implementation.[28] Additionally, the size of *AuthorizationReq* messages is increased to 355 bytes due to the included CSID and $X2$, compared to its size of 308 bytes in the default RISE V2G implementation.

8.2.2. Computational Overhead

The solution creates computational overhead at several processes. All times were measured in the EV Pi's ISO 15118 implementation (as this best reflects the impact on the user) using Java's *System.nanoTime()*[29] function unless specified otherwise. All measurements were repeated 100 times and we report the averages along with the standard deviation (σ). The results are summarized in Table 8.2. Appendix C shows measured times in detail.

The modified authorization whitelists need to be generated regularly and, for offline authorization, require a transformation from the CSO. Whitelist management times are measured using Python's *time.time()*[30]

[27] The DAA credential alone is 256 bytes, however, the credential is serialized based on the implementation of [205], which increases its size to the reported 382 bytes.

[28] As the size of the authorization nonce is 16 bytes, it is unclear as to why the EXI encoded message is only 4 bytes larger. A detailed analysis of the EXI encoding process is beyond the scope of this thesis.

[29] `https://docs.oracle.com/javase/8/docs/api/java/lang/System.html#nanoTime--` (visited on 2020-12-21)

[30] `https://docs.python.org/3/library/time.html#time.time` (visited on 2020-12-21)

function. The time for whitelist generation at the eMSP was 10197.8 ms (± 325.8 ms). The time from sending a whitelist from the eMSP to the CSO (measured from the CSO sending its OCPI request until receiving the eMSP's response via the CCH) was 42235.1 ms (± 1296.9 ms). The time for whitelist transformation at the CSO was 10042.5 ms (± 344.6 ms). The time for sending a part of the whitelist with only 2000 entries from the CSO to a CS (measured from the CSO's *SendLocalListRequest* message until receiving the CS confirmation via the *SendLocalListResponse* message) was 5038.9 ms (± 85.6 ms). Extrapolated to the whole whitelist, this would result in about 82 minutes, showing that the OCPP method of whitelist distribution is not suitable for the proposed privacy-preserving list. However, OCPP already includes methods for transferring larger amounts of data (e.g., logs from a CS or firmware updates to a CS) whereby the CSO sends a Uniform Resource Locator (URL) to the CS and the CS up-/downloads data to/from this location via an out-of-band method (e.g., using File Transfer Protocol (FTP); cf. [148], Section L and Section N). Transferring the full OCPP whitelist in compressed form[31] from CSO to CS using an out-of-band method[32] took 33473.6 ms (± 2463.8 ms) (measured using *date*[33]). The distribution of OCPI whitelists could be optimized in a similar fashion. Transferring the full OCPI whitelist in compressed form[34] via an out-of-band method[35] directly from eMSP to CSO took 740.5 ms (± 292.1 ms) (measured using *date*). As the processes of authorization whitelist generation and distribution in current EV charging protocols does not have to be performed regularly, the overhead of the current processes is argued to be negligible and was not implemented/measured.

The generation of certificate installation requests now requires the generation of a DAA key pair, the validation of the PC key's policy, a signature generation in the TPM, the validation of the CPS' certificate chain, and the encryption of the request. The time for certificate installation request generation was 1858.8 ms (± 124.1 ms). For comparison, the default RISE V2G implementation of the certificate installation request generation process, i.e., loading the EV's key pair from a Java *KeyStore*[36] and using it to sign the request on the host, took 1596.9 ms (± 23.3 ms).

The handling of certificate installation requests and generation of certificate installation responses now requires the decryption of the request, the generation of a DAA credential, the encryption of the DAA credential using the credential protection mechanism, the generation of the eMAID key, and the encryption of the eMAID key using the duplicate mechanism. The time for the handling of a certificate installation request and the generation of a certificate installation response (measured from before sending the built *CertificateInstallationReq* message until after receiving the *CertificateInstallationRes* message) was 341.9 ms (± 30.2 ms). The time for this process on the eMSP alone (which also implements the CPS role; cf. Section 8.1.1) measured using Python's *time.time()* function was 100.7 ms (± 11.1 ms). For comparison, the default RISE V2G implementation of the certificate request handling and response generation process – i.e., the validation of the request, the loading of a pre-generated contract certificate and key pair, the encryption of the pre-generated private contract key, and the generation of a signature over the response (all by the CS)[37] – took 409.2 ms (± 20.1 ms).

The handling of certificate installation responses now requires the execution of the `TPM2_ActivateCredential` and `TPM2_Import` commands as well as the verification of the DAA credential using the eMSP's public group key. The time for the handling of certificate installation response was 717.1 ms (± 24.7 ms). For

[31] The OCPP list was compressed using `xz` resulting in a 169.9 MB file for the 556.1 MB OCPP data; cf. `https://linux.die.net/man/1/xz` (visited on 2020-12-23)
[32] The out-of-band method was implemented using `netcat`; cf. `https://linux.die.net/man/1/nc` (visited on 2020-12-23)
[33] `https://man7.org/linux/man-pages/man1/date.1.html` (visited on 2020-12-23)
[34] The OCPI list was compressed using `xz` resulting in a 132.8 MB file for the 536.7 MB OCPI data.
[35] The out-of-band method was implemented using `netcat`.
[36] `https://docs.oracle.com/javase/8/docs/api/java/security/KeyStore.html` (visited on 2020-12-21)
[37] The inclusion of backend actors in the default certificate installation response generation process was not implemented due to time constraints.

comparison, the default RISE V2G implementation of this process, i.e., the decryption of the contract key on the host and storing the contract credentials in a Java *KeyStore*, took 1685.7 ms (± 62.9 ms).

The generation of a *PaymentDetailsReq* message for the PnC mechanism now requires the creation of a session key pair in the TPM, the randomization of the DAA credential, the certification of the session key (including the `TPM2_Commit` command, the generation of a hash on the host, and the `TPM2_Certify` command, whereby both TPM commands require a validation of the DAA key pair's policy), and the generation of $X1$ (an HMAC in the TPM and a hash on the host). The time for *PaymentDetailsReq* generation directly after credential installation was 1063.4 ms (± 32.5 ms). Note that the performance of *PaymentDetailsReq* generation directly after credential installation is significantly affected by the fact that most of the processes cannot be pre-calculated as the required data is only available after receiving and handling the *CertificateInstallationRes* message. The performance of *PaymentDetailsReq* generation with already installed credentials from a previous session was 9.9 ms (± 1.9 ms). With already installed credentials, the EV is able to finish the loading of the required keys into its TPM, the creation of a session key pair in its TPM, the randomization of the DAA credential, the calculation of two policy sessions (for the `TPM2_Commit` and `TPM2_Certify` commands), and the execution of the `TPM2_Commit` command (all steps prior to building the qualifying data c; cf. Fig. 7.10) during the communication setup between EV and CS. Additionally, the execution of the `TPM2_Certify` command can be started as soon as the EV receives the ISO 15118 V2G session ID in the header of the CS's *SessionSetupRes* message and is finished before the *PaymentDetailsReq* message can be sent. For comparison, the default RISE V2G implementation of this process, i.e., only loading the EV's certificate and building the message, took 24.3 ms (± 50.7 ms).

The handling of a *PaymentDetailsReq* message now requires the verification of the DAA credential, the verification of the DAA signature over the certification data structure, the validation of the session key based on the certification data structure, as well as the query for $X1$ in the CS' whitelist. The time for handling a *PaymentDetailsReq* message (measured from before sending the built *PaymentDetailsReq* message until after receiving the *PaymentDetailsRes* message) was 322.5 ms (± 18.7 ms). For comparison, the default RISE V2G implementation of this process, i.e., the validation of the EV's certificate chain, took 94.1 ms (± 58.8 ms).

The generation of an *AuthorizationReq* message for the PnC mechanism now requires the generation of an HMAC by the EV's TPM, a signature by the TPM (using the private session key), and the calculation of two hashes on the host for $X2$. The time for *AuthorizationReq* generation was 129.2 ms (± 4.9 ms). For comparison, the default RISE V2G implementation of this process, i.e., a signature generation by the host, took 73.3 ms (± 9.8 ms).

The offline authorization on the CS only requires an additional hash or two hashes by the backend actors in case of online authorization. The times for authorization handling were measured from before sending the built *AuthorizationReq* message until after receiving the *AuthorizationRes* message, i.e., the times include the signature verification by the CS. The time for offline authorization was 79.0 ms (± 10.7 ms). The time for online authorization via the CSO was 121.3 ms (± 15.1 ms). Most of the additional overhead compared to the offline case is caused by the additional messages as it only took the CSO 6.7 ms (± 2.4 ms; measured using Python's *time.time()* function) to validate the user's authorization status based on its whitelist. The time for online authorization via the eMSP was 199.4 ms (± 21.7 ms). For comparison, the default RISE V2G implementation of this process, i.e., only the signature verification by the CS with no additional authorization checks, took 67.3 ms (± 6.6 ms).

Table 8.2.: Computational Overhead

Process	Time in ms			
	Mean	σ	Max	Min
Normal Whitelist Management	*Negligible*			
Privacy-Preserving Whitelist Management				
List Generation at the eMSP	10197.8	±325.8	11284.0	9657.2
List Distribution to the CSO (OCPI)	42235.1	±1296.9	46903.5	38258.4
List Distribution to the CSO (Out-of-Band)	740.5	±292.1	1711.7	482.5
List Transformation at the CSO	10042.5	±344.6	11141.1	9072.7
List Distribution to the CS (OCPP)	-	-	-	-
List Distribution to the CS (Out-of-Band)	33473.6	±2463.8	43139.9	29415.4
Normal Contract Credential Installation				
CertificateInstallationReq Generation	1596.9	±23.3	1742.2	1569.2
CertificateInstallationReq Handling	409.2	±20.1	457.9	361.2
CertificateInstallationRes Handling	1685.7	±62.9	1831.1	1603.6
Privacy-Preserving Contract Credential Installation				
CertificateInstallationReq Generation	1858.8	±124.1	2880.8	1805.2
CertificateInstallationReq Handling	341.9	±30.2	440.3	293.8
CertificateInstallationRes Handling	717.1	±24.7	760.4	663.0
Normal Charge Authorization				
PaymentDetailsReq Generation	24.3	±50.7	138.6	1.0
PaymentDetailsReq Handling	94.1	±58.8	220.5	54.9
AuthorizationReq Generation	73.3	±9.8	96.9	41.0
Offline Authorization	67.3	±6.6	85.9	56.3
Privacy-Preserving Charge Authorization				
PaymentDetailsReq Generation (after install)	1063.4	±32.5	1126.7	960.6
PaymentDetailsReq Generation (not after install)	9.9	±1.9	16.2	7.1
PaymentDetailsReq Handling	322.5	±18.7	363.8	282.2
AuthorizationReq Generation	129.2	±4.9	150.8	118.2
Offline Authorization	79.0	±10.7	111.4	62.3

9. Evaluation

The following sections evaluate the proposed concept for privacy-preserving charge authorization and billing (cf. Section 7), based on the requirements defined in Section 6, and under consideration of the set goals (cf. Section 4.3) as well as the given adversary model and the discussed threats (cf. Section 5).

9.1. Functional Evaluation

The proposed solution addresses the functional requirements from Section 6.3 as discussed in the following:

FR_1 *Session Stop Restrictions:* The proposed solution is able to mostly offer the same session stop restrictions as the current mechanisms. That is, the stopping of as session via local EIM-based authorization can be restricted to the same token that was used to start the session. This restriction can also be validated by an offline CS. The stopping of a session by a different token in the same group is still possible, however, only via an online request in order to meet the set privacy goals.

FR_2 *Support Existing Billing Relations:* The proposed solution supports existing billing relations. That is, independent of the authorization mechanism, a CSO is informed about the ID of a user's eMSP and can thus bill the eMSP for the charging session. The eMSP is informed about the user's transaction pseudonym ID Token can link it to the user's identity. Thus the eMSP can bill the user for the charging session.

FR_3 *Minimal Overhead:* The overhead of the proposed solution was measured based on a proof-of-concept implementation as detailed in Section 8. The overhead in the contract credential installation process is very low. In fact, the proof-of-concept implementation even outperformed the baseline as the entire installation process was 774 ms faster.[1] The overhead in the PnC authentication process is arguably low. While the time for the entire PnC authentication process[2] is increased by 269.9 ms, note that the time for the prior ISO 15118 session setup alone is 3305.3 ms ($\pm$ 25.3).[3] Thus, the overhead in PnC authentication is argued to be insignificant with regard to user experience. The overhead that is caused by the proposed authorization mechanism during a charge session is also shown to be low (11.7 ms for offline authorization on the CS) and its cryptographic requirements are kept minimal to enable its use for local EIM-based charge authorization.

[1] The time for the "entire installation process" as mentioned here is calculated as the sum of the mean execution times of the sub-processes as reported in Table 8.2.

[2] The time for the "PnC authentication process" as mentioned here is calculated as the sum of the mean execution times of the first three authorization sub-processes (using the "*PaymentDetailsReq* Generation (not after install)" time as it reflects the more common case) as reported in Table 8.2.

[3] The time for the prior ISO 15118 session setup as reported here was measured on the EV Pi using the same setup as in Section 8. The time starts from right before the EV sends its "SECC Discovery Request" message (cf. Fig 2.5a) until before the EV starts building its *PaymentDetailsReq* message (without a certificate installation in between).

The overhead that is introduced in the authorization whitelist management processes, however, is significant as the proposed method requires regular updates for all EV users. The proof-of-concept implementation shows that the overhead is manageable by backend systems, even when scaled to the expected numbers of future EV adoption, especially when considering that these processes run independently of charging sessions, i.e., do not affect user experience and thus are less time critical. The overhead for offline whitelist distribution to CSs on the other hand might be too high and could require further optimizations (e.g., by further dividing whitelists by region as mentioned in Section 8.2.1). Alternatively, an OCPP Local Controller (LC) (cf. Section 2.4.2) could be used to reduce the overhead of offline whitelists, i.e., by distributing the offline whitelists to LCs (instead of CSs) and having these LCs answer authorization requests form CSs without an online connection to the CSO. Since, when using this LC method, the compromise of a single LC by a Local Adversary affects multiple CSs, an investment into hardware-supported (e.g., TPM-based) security features for LCs might be justified. A more detailed analysis of the LC method, however, is beyond the scope of this thesis. A different possibility for reducing the overhead of offline whitelist distribution is to use a successful DAA-based PnC authentication as implicit charge authorization at an offline CS (while still receiving the authorization values for billing). Note that using PnC authentication as implicit charge authorization is already one of the possibilities considered by current charge protocols (cf. Section 4.2.2). Using this method for offline authorization would allow the omission of authorization values for the PnC case from offline whitelists.

FR_4 *Support for Offline Operation:* The proposed solution supports the operation of CSs in semi-online mode, i.e., CSs can operate mostly offline with periodic synchronizations wit their CSO. The only prerequisites for offline operation are that the CS is provided with an authorization whitelist and with eMSP roots, which are the same prerequisites as in the existing charging protocols. The main changes for offline authorization are that authorization whitelists need to be updated more frequently and that DAA signatures need to be verified locally.

FR_5 *Support for Reservations:* As discussed in Section 7.1.3 the proposed solution supports the reservation of CSs by an EV user via their eMSP and a reserved CS can enforce that a reservation is only consumed by the user who placed it.

FR_6 *Support for Authorization Mechanisms:* The proposed solution supports the existing authorization methods, i.e., it supports local EIM-based authorization (cf. Section 7.1.2), local PnC-based authorization (cf. Section 7.2), and remote authorization (cf. Section 7.1.3). Furthermore, EIM-based authorization is possible without changes to the EV. Note that without any changes to the EV, the EV still sends static identifiers and thus counteracts the privacy benefits of the authorization. Hence, at least minimal changes to EVs in order to de-identify their outgoing traffic would be required for a reasonable use of the privacy-preserving EIM-based authorization.

FR_7 *Support Contract Credential Installation:* The proposed solution supports the automatic installation of contract credentials into an EV following the general flow of the existing methods (cf. Section 7.2.4). Regarding usability, an EV user only has to register their EV (using existing methods) at the eMSP and installation is performed automatically during the first charging session, i.e., the usability is not impacted in comparison to the existing protocols.

FR_8 *Support Credential Pools:* The proposed solution supports the use of credential pools for the temporal decoupling of credential installation-related processes (cf. Section 4.2.5).

FR_9 *Minimal Changes:* As shown by the proof of concept implementation, an integration of the proposed solution with only minimal changes is generally possible (cf. Section 8). While some conflicts with the existing specifications were identified (regarding the size of the eMSP's certificate for inclusion in

ISO 15118 messages and the size of Base64 encoded ISO 15118 *CertificateInstallationRes* messages for OCPP; cf. Section 8.2), it is possible to address these conflicts with simple workarounds as discussed in Section 8.2. We encountered no other timing or size constraints that could not be met by the implementation. As the proposed solution is compatible with the processes and data flows of existing EV charging protocols, we argue that FR_9 is met.

9.2. Security Evaluation

The proposed solution addresses the security requirements from Section 6.1, under consideration of the adversary model and threats from Section 5 and the goals from Section 4.3, as discussed in the following:

SR_1 *Secure EV Credentials:* All of the EV's credentials are protected by its TPM (cf. Section 2.5.1), which protects them from information disclosure during their storage (cf. ST_{13}) and usage (cf. ST_{14}). In particular, the EV's provisioning credentials, i.e., the PC key pair and the added EC key pair, are generated in the EV's TPM and their private values never leave this TPM in clear text. Since the extraction of credentials from the TPM is excluded by the adversary model (cf. Section 5.1), an adversary cannot extract an EV's provisioning credentials during storage (ST_{13} *BF*). Similarly, provisioning credentials can only be used in the EV's TPM, preventing their extraction by an adversary during usage (ST_{14} *BF* and *AF3*).

Regarding contract credentials, the DAA key pairs (CC^{DAA}) and session key pairs (CC^{Sess}) are generated by the EV's TPM and thus, similarly to the provisioning credentials, cannot be extracted by an adversary during their storage (ST_{13} *AF*) and usage (ST_{14} *AF1* and *AF4*). The eMAID key (SK_{eMAID}) is generated securely at the eMSP and encrypted for the EV's TPM using the EC public key with the TPM's protection mechanisms. Hence, as long as an adversary cannot spoof an EV for credential installation and decrypt the corresponding response, the eMAID key is secure.

Since the authenticity of provisioning credentials is protected by a trusted OEM's provisioning certificate, an adversary cannot spoof an EV for credential installation based on falsified provisioning credentials. Since provisioning credentials are protected from information disclosure, an adversary cannot spoof an EV for credential installation based on obtained provisioning credentials (ST_2 *AF2*).

Additionally, in order to show that an adversary cannot maliciously obtain contract credentials based on a flaw in the credential installation protocol, in Appendix D we show that the modified join procedure maintains the symbolically proven security properties of the original join procedure from [205]. That is, the symbolic model of [205] is adjusted in order to reflect the changes proposed in Section 7.2 and the resulting model is shown to provide injective agreement [126] to the signer (the EV) with the issuer (the eMSP) on the DAA credentials. While the inclusion of SK_{eMAID} is not modeled, it is argued to be secure as its protection is based on the TPM's mechanisms and uses the EC key pair (i.e., the same key pair that is used for the protection of the DAA credentials).

Note, however, that with the proposed method, the replay of certificate installation request messages is possible (e.g., by a manipulated CS) as the freshness of these messages is not ensured. Hence, injective agreement in the reverse direction (to the issuer with the signer on the join request) cannot be provided.[4]

[4]Note that injective agreement to the eMSP (issuer) is not desirable for the use case as, for ISO 15118-20 [98], a CPS should be able to forward the certificate installation request of one EV to multiple eMSPs (to collect all possible contract credentials of one user) and thus at most injective agreement to the CPS with the signer on the credential request should be considered.

While not proven by the original model from [205], the join procedure from [205] arguably provides this property due to the issuer's nonce (cf Section 2.5.4). As certificate installation requests in current EV charging protocols do not ensure freshness, the lack of this security property does not decrease the security level of credential installation. Additionally, even if certificate installation requests can be replayed, the corresponding responses are still encrypted for the correct EV's TPM and thus the mentioned reverse injective agreement property is argued not to be required for SR_1.

Since contract credentials are protected from information disclosure, an adversary cannot spoof an EV to a CS based on obtained contract credentials (ST_2 *BF*). Since DAA key pairs (CC^{DAA}) are certified by trusted eMSPs, an adversary cannot spoof an EV to a CS based on falsified DAA credentials. Since session key pairs (CC^{Sess}) are certified by the TPM using the DAA key pair, an adversary cannot spoof an EV based on falsified session credentials.

Since the credential-based EV spoofing threats are prevented, the resulting elevation of privilege threat is no longer possible (ST_{16} *BF*). Similarly, the resulting information disclosure of transient data over ISO 15118 is no longer possible (ST_{14} *AF2*). Furthermore, the repudiation of charge authorizations and charge session data that resulted from credential-based EV spoofing are prevented (ST_6 *BF* and ST_7 *AF*).

SR_2 *Secure Charge Authorization:* The method of local EIM-based charge authorization is argued to meet the security goals under the assumed adversary as discussed in the following.

An adversary cannot guess valid authorization values without knowing the corresponding secret key or breaking the cryptographically secure hash/HMAC functions. Since the data stores and processes of EIM devices as well as the used cryptographic primitives are assumed to be secure (cf. Section 5.1), it is not possible for an adversary to guess valid authorization values (ST_{10} *BF* and *AF*), which also prevents the resulting spoofing threat (ST_3 *AF4*).

Additionally, it is not possible for an adversary to authorize a charging session at any CS solely based on a disclosed CS authorization whitelist as whitelists only contain hashes of the actual authorization values (ST_3 *BF* and *AF2*). At most, an adversary could use a disclosed CS authorization whitelist in combination with an information disclosure at an EV user's local EIM device in order to spoof the affected user at the affected CS (the adversary uses the information from the whitelist in order to receive valid authorization values from the user's local EIM device and replays these values at the CS). Since this attack has a relatively low impact in comparison to the required effort (the adversary can only spoof the attacked user at the attacked CS and only for a limited amount of authorizations), we argue that the resulting risk is low enough to be acceptable.

Furthermore, it is not possible for an adversary to authorize a charging session at any CS by replaying previously transmitted authorization data (ST_3 *AF1* and *AF3*) as the transmitted authorization values are CS-specific (replay at a different CS is not possible) and can only be used once (replay at the same CS is not possible). At most, an adversary could use the previously discussed attack (based on a disclosed CS authorization whitelist and information disclosure at an EV user's local EIM device), which is argued to be of low risk.

Since the EV user spoofing threats are addressed (or reduced to an acceptable level of risk), the resulting authorization-related repudiation threats are argued to be addressed (ST_6 *AF1* and ST_7 *AF*). Similarly, the resulting elevation of privilege threat is argued to be addressed (ST_{16} *AF*). While a manipulated CS can still report wrong authorization values for a charging session (ST_6 *AF2*), this threat is limit to received authorization values and by the fact that the same authorization values cannot be reported multiple times without being detected as an error. Thus, the remaining risk is judged to be acceptable.

The authorization status of EV users can be revoked using the existing methods, i.e., using differential whitelist updates over secure channels (ST_{12} *BF*). While the revocation of a single user allows for the linkability of the revoked authorization values, only unused authorization values need to be revoked. If only unused authorization values can be linked, it is not possible to further link this data to any charge session information and thus we do not consider the linkability of unused authorization values as a privacy threat. While eMSPs might not be instantly informed about the authorizations of all their users (e.g., if authorization was based on a whitelist), they must still be regularly updated on their users' charge sessions/authorizations for billing purposes. Thus, the revocation of a single user at most enables the linkability of the user's few most recent charging sessions, which is argued to be acceptable.

Note that, as TPMs in the PnC case and RFID tags in the EIM case are assumed to be tamper-resistant (cf. Section 5.1), i.e., as key compromises are not considered, the proposed method does not provide forward security/privacy. While the local EIM-based method could be adjusted to incorporate forward security/privacy (e.g., by replacing the counter with a hash-chain based key update mechanism; cf. Section 3.1) the PnC-based method would require more consideration as its functions are limited by the TPM's specification. In order to implement a key update mechanism with the PnC-based method, the new key would have to be held in plaintext by the host, i.e., outside the TPM's protection, before it can be imported into the TPM, making the key vulnerable to compromise even without breaking the TPM's assumed tamper-resistance. As the devices (for EIM and PnC) all have unique keys and their authorization status can be revoked, the compromise of a single device, however, has a very limited damage potential. Thus, it is likely that the cost of a successful compromise is higher than the benefit. Due to this observation and considering the adversary model/assumptions of Section 5.1, a method for forward security/privacy is not considered necessary for the use case.

Additionally, as denial of service attacks are out of scope (cf. Section 5), desynchronization attacks (cf. Section 3.1) are not considered. While the proposed method can handle minor desynchronizations (depending on how many authorization values an eMSP pre-calculates), it is likely that a determined adversary could still conduct a successful desynchronization attack. This issue could, for instance, be addressed via a recovery function, i.e., as no key updates mechanism is used, a desynchronization device could encrypt its counter for the eMSP and the eMSP could use a "key search" procedure (cf. Section 3.1) to recover the device's state. However, as denial of service attacks are out of scope, this method is not considered in further detail.

SR_3 *EV Integrity:* All of the EV's PnC keys are conditioned on a `TPM2_PolicyAuthorize` policy that is used to authorize a `TPM2_PolicyPCR` digest. Thus, the corresponding private keys can only be used if the EV booted into a trusted software state. In particular, as per [66], the PC key pair is sealed to this policy such that the EV can only sign certificate installation requests if its firmware was not manipulated. Additionally, since the DAA join protocol is extended to include the DAA key pair's expected policy in the credential protection encryption mechanism (using the PC_{cert} based method from [66]), an EV's DAA contract credentials must be sealed to the expected policy in order for the TPM to allow the decryption of DAA credentials (called trustworthy credential enrollment in [66]). Hence, if an EV can use valid (randomized) DAA credentials during PnC-based authentication at a CS, it is guaranteed that the corresponding DAA key pair is conditioned on the authorization policy and the EV can only create a DAA signature if it is booted into a trusted software state.

Since authentication of the EV is conditioned on the integrity of its software state, it is no longer possible to spoof an EV based on a tampered firmware data store (ST_2 *BF*, *AF1*, and *AF2*). Similarly, the resulting information disclosure (ST_{14} *BF*, *AF1*, and *AF2*), repudiation (ST_6 *BF* and ST_7 *AF*), and elevation of privilege threats (ST_{16} *BF*) are no longer possible.

Note, however, that since the certificate installation request contains no sessions specific data (as is the case in ISO 15118), it is possible to replay old certificate installation requests (as mentioned for SR_1). Hence, an EV with a manipulated firmware could send an old request and receive a valid response, which could arguably be considered a spoofing threat. However, as the returned DAA credential is protected using the TPM's credential protection mechanism (cf. Section 2.5.3; with the public EC key from the EV's provisioning certificate for encryption and the public DAA key as the key that is associated with the credential), the DAA credential can only be decrypted if the DAA key pair was generated in the same TPM as the EC key pair and with the expected attributes (cf. trustworthy credential enrollment in [66]). Since the DAA key pair's expected attributes include the authorization policy, an EV with a manipulated firmware that received a valid certificate installation response using a replayed request can still only decrypt the received credential if its DAA key is sealed to the correct policy and thus can only authenticate itself if its firmware is re-set into a trusted state. Hence, this certificate request spoofing threat does not affect the assurance of an EV's software integrity with respect to charge authorization and SR_3 is argued to be met.

SR_4 *End-to-End Protection of Charge Session Data:* With the proposed PnC solution, all billing-relevant charge session data can be authenticity protected between the EV and the corresponding eMSP. For one, since the eMSP's public DAA group key is only added to its certificate via an extension and does not replace the existing key, the existing key can still be used to protect the authenticity of an eMSP's tariffs using the existing method. Secondly, since the EV generates a session key in its TPM, which is certified based on the EV's DAA contract key with a credential from the eMSP, the EV can use this session key via its TPM for authenticity protection of any billing-relevant data (by signing the data) throughout the charge session (ST_7 *BF*). Note, however, that meter values originate from the CS and thus cannot fully be protected using the proposed method, i.e., a manipulated CS could tamper with the meter values before sending them to the EV where they are signed. In order to address this threat, CSs could, e.g., be equipped with tamper-protected energy meters that sign all of their reported meter values. In this case, an EV could verify the authenticity of meter values before it signs them with its session key (note that a meter's signature cannot be reported to the eMSP as this could reveal the user's charge location). As the assurance of trustworthiness for CS-originating data, however, is beyond the scope of this thesis, we argue that SR_4 is adequately addressed.

9.3. Privacy Evaluation

The proposed solution addresses the privacy requirements from Section 6.2, under consideration of the adversary model and threats from Section 5 and the goals from Section 4.3, as discussed in the following:

PR_1 *Data Minimization:* With the proposed methods for privacy-preserving charge authorization and billing, CSs and their CSOs only learn a charge session's information (location, duration, consumption, etc.), a transaction pseudonym of the user, and the user's eMSP, which is in line with the data minimization goals (Section 4.3). The eMSPs on the other hand only learn a charge session's information without the location and are able to identify the specific EV user (by linking the transaction pseudonym to the user's identity) for billing purposes, which too is in line with the data minimization goals (Section 4.3). The CCH only forwards the charge session's information (without the location) and the user's transaction pseudonym between CSO and eMSP. As the pseudonyms are not linkable and the charge session information cannot be linked to a location, the transferred data is arguably de-identified enough in order to not constitute personal data. Thus, PR_1 is argued to be met with regard to authorization and

billing data. Additionally, for contract credential installation, the only revealed personal data is the certificate installation request (including the PCID as EV identifier and TPM public keys as TPM and thus EV identifiers) at the CPS and the eMSP, which again is in line with the data minimization goals (Section 4.3) and arguably results in a negligible privacy risk as installations are relatively infrequent. As the focus of this thesis are the required processes for charge authorization and billing, ST_{15} is thus argued to be addressed.

PR_2 *Unlinkable EV User Authorizations:* With the proposed solution, the authorization data of an EV user is different for each authorization request. Additionally, since the authorization data is calculated based on a secret key, only an actor with knowledge of this key can link different authorizations of the same user. Since this key is only available to the eMSP and to either the EIM device or to the TPM, i.e., the key is only available in data stores and processes that are assumed to be secure, multiple authorizations of the same user cannot be linked based on the calculated values. Note, however, that the authorization data contains the static eMSP ID and thus multiple authorizations of the same user could reliably be linked based on the eMSP ID if only one user charges via a specific eMSP. Hence, in practice, the provided anonymity of the solution is limited to the anonymity set size as defined by the amount of users that are expected to charge under the same eMSP ID. For instance, if an adversary knows that only one user of a specific eMSP charges at a given location, all authorization of this user at the location could be linked. As the eMSP ID, however, is transferred as part of the minimum required data, the resulting anonymity set size is assumed to be large enough to prevent linkability (cf. Section 5.1) and we thus argue that the corresponding linkability threats are addresses (PT_1 *BF* and *AF2*).

With the proposed solution, group-based charge stop authorization only allows for two sets of authorization data (one for the start and one for the stop) to be linked by any actor but the eMSP. Since this authorization data can, however, not be linked to any other authorizations, the limited linkability due to a group-based charge stop authorization is not considered a threat (PT_2 *BF*, *AF1*, and *AF2*). Thus, we argue that PR_2 is met.

PR_3 *Unlinkable PnC Authentications:* With the proposed solution, PnC authentication is based on a DAA signature and a session key pair. in general, DAA-based signatures cannot be linked if either different basenames are used or (as is the case in the proposed solution) if no basename is used. This property is shown to hold in the TPM's DAA-based `TPM2_Certify` process by [205]. While the session key pair is used for normal ECC-based signatures, i.e., linkable signatures, a different key pair is used for each session and thus the use of the session key pair does not allow the linking of different PnC authentications. Hence, PR_3 is argued to be met and the corresponding linkability threat is addressed (PT_1 *AF1*).

PR_4 *Unlinkability of EV Users and EVs:* With the proposed solution, the only use of EV IDs (the PCID and TPM public keys) is in the credential installation process.[5] This ID is only revealed to the CPS and the eMSP and any user specific response data is encrypted by the eMSP for the EV (or its TPM). Additionally, as the certificate installation response message for the CPS does not include any session or user identifiers, the CPS cannot link the EV's ID to any later received session or authorization data. Hence, only the eMSP can link an EV user to their EV's ID, which is in line with the discussed goals (Section 4.3). Thus, PR_4 is argued to be met and the corresponding linkability threats are addressed (PT_3 *BF*, *AF1*, and *AF2*).

[5] Note that the EVCC ID that is sent over ISO 15118 during V2G communication session setup (cf. Section 4.2.1), i.e., the only other EV ID (cf. Section 4.2), contains the EVCC's MAC address (cf. [97], [V2G2-879]), which is assumed to be randomized (cf. Section 5.1).

PR_5 *Unlinkability of Charge Sessions:* With the proposed solution, charge session data can only be linked to the user's transaction pseudonym, which is not considered a threat (PT_5 *BF* and *AF*). Only the eMSP can link transaction pseudonyms to the user's identity for billing purposes, which is in line with the discussed goals (Section 4.3). Additionally, as the user's authorization data is unlinkable (cf. PR_2), it can no longer be used to link multiple sessions of the same user by anyone but the eMSP (PT_4 *BF* and *AF1*). The linkability of charge sessions based on an EV's IDs is no longer possible since, as discussed for PR_4, EV IDs (in particular the PCID) can only be linked to session data by the eMSP (PT_4 *AF2*). The linking of multiple charge sessions based on only the sessions' data (PT_4 *AF3*) is assumed to not be possible as PR_1 is met (cf. Section 5.1). Similarly, linking session data to the charge location (PT_6 *AF1* and *AF2*) is argued to be addressed as PR_1 is met. Note that, as indicated by the assumptions in Section 5.1, the minimized data model might still allow for charge session linkability due to the results of [185] and further data generalization techniques (beyond the scope of this thesis) could be required to fully address PT_4 *AF3* and PT_6 *AF2*.

PR_6 *Unlinkability of EV Users and Locations:* With the proposed solution, CSs only receive the user's transaction pseudonym. Hence, the threat of linkability of an EV user to a charge location by a malicious CS is considered to be addressed (PT_7 *BF* and *AF1*). Similarly, eavesdropping on the local EIM communication only reveals transaction pseudonyms and thus the corresponding threat of linkability of an EV user to a charge location is considered to be addressed (PT_7 *AF2*). The linking of EV users to charge locations by backend actors (PT_7 *AF3* and *AF4*) is assumed not to be possible as PR_1 is met (cf. Section 5.1). Note that, similarly to PR_5, further data generalization techniques could be required to fully address PT_7 *AF4* (e.g., by generalizing the timestamps of live CS status updates such that an eMSP cannot link them to an EV user's session data). The threat of linking EV users to charge locations based on OCSP requests (PT_7 *AF5*) is addressed by using the charge authorization method for revocation (cf. SR_2). Thus, PR_6 is argued to be met.

PR_7 *Anonymity of EV Users:* With the proposed solution, different authorizations of an EV user are no longer linkable (cf. PR_2). Thus, the identifiability threats that result from an adversary being able to build movement profiles based on authorization data (PT_8 *BF*, *AF2*, *AF4*, and *AF5*) are argued to be addressed. Additionally, as linkability of charge sessions is addressed (cf. PR_5) and as PR_1 is met, the identifiability threats that result from an adversary being able to build movement profiles based on session data (PT_8 *AF1* and *AF3*) are also argued to be addressed. Note that, based on the results of [185] and as indicated by the assumptions in Section 5.1, the CSO's minimized data model might still allow them to build movement profiles as they still receive precise charge session data along with the session's location. However, as the practical applicability of the observations from [185] have, to the best of our knowledge, not been shown and as CSOs are (unlike the CCH and eMSP)[6] not informed about all sessions of a particular user, we consider the remaining privacy risk to be low. Thus, we argue that only an EV user's eMSP is able to identify the user, which is in line with the discussed goals (Section 4.3), and PR_7 is argued to be met.

[6] A CCH and eMSP are informed about all sessions of a particular user under the assumption that an EV user only has a contract with one eMSP and that all roaming traffic between this eMSP and CSOs is routed through one CCH.

10. Conclusion

EVs provide a promising alternative to traditional combustion engine vehicles and the adoption of EVs is steadily growing world wide. The charging of an EV generally requires that the corresponding EV user can be billed for the consumed energy. In order to enable a convenient billing process, several market roles have been established and communication protocols have been developed. These communication protocols, however, often neglect the import aspect of user privacy and a large amount of personal data is unnecessarily revealed to a variety of backend operators. Considering the strict provisions of contemporary data privacy law, namely of the GDPR, addressing these privacy risks is not only advantageous for EV users but also in the best interest of the EV charging backend operators.

This thesis provides an overview of the entities that are involved in EV charging focusing on their tasks and exchanged data. Based on a detailed analysis of open source communication protocols, the involved personal data is identified and a minimized data model is constructed under consideration of the entities' existing tasks and functions. Furthermore, a realistically strong adversary model for the EV charging use case is presented. The adversary model is used in order to conduct an extensive security- and privacy threat analysis based on the widely used and highly regarded STRIDE (for security) and LINDDUN (for privacy) threat modeling methodologies. The threat analysis highlights that current EV charging protocols particularly lack security considerations for a local adversary with physical access to components. Additionally, lacking privacy considerations are identified that enable an adversary to build detailed movement profiles of EV users, which can be used to infer particularly sensitive information about the affected users. For instance, movement profiles could reveal information about an individual's wealth, residence, or hospital visits and they could enable real world attacks like burglaries.

In order to address the identified threats, this thesis presents a privacy-preserving solution for charge authorization. The proposed authorization method supports existing concepts in EV charging protocols (including offline authorization and CS reservations) to the greatest possible extend while enabling the implementation of the presented minimized data model and allowing for privacy-preserving user-specific billing. Furthermore, the possibility of using a TPM-based DAA scheme for privacy-preserving PnC authentication is investigated. The DAA-based PnC authentication method is combined with the privacy-preserving authorization mechanism in order to provide the highest possible level of security while maintaining compatibility to existing billing relations and authorization concepts. The solution can be integrated into existing protocols based on minor changes in message definitions and message handling while not requiring fundamental changes to functions of involved entities or to the overall message flow.

A proof-of-concept implementation is developed in order to show the feasibility of the proposed solution. In particular, the implementation shows that the DAA scheme can be integrated into EV charging protocols with only minimal overhead. Similarly, the proposed authorization method is shown to cause only minimal overhead when verifying a user's authorization status. Hence, the impact on the user experience is argued to be insignificant. However, the required overhead for preparing the proposed authorization method is shown to be relatively high. While the implementation shows that the overhead is manageable in the scope of backend

systems, the distribution of whitelists for offline authorization at a CS could require further performance optimizations.

The solution is evaluated under the considered adversary model and argued to adequately address the existing security- and privacy threats of EV charging protocols. In particular, due to the use of a TPM as hardware security anchor, the solution can provide security even in the presence of a local adversary with physical access to components. Additionally, the privacy-preserving authorization concept and, in the PnC case, used DAA scheme can effectively address the building of movement profiles based on authorization data. Thus, and considering the solution's extensive support for existing concepts and the generally low overhead, the presented concept is argued to be an appropriate method for addressing the privacy concerns of EV charging and for bringing the considered protocols closer in line with the GDPR's provisions.

The evaluation further highlights some limitations of the proposed solution that could be addressed by future work. With regard to privacy, the threat analysis shows that CSO may be able to link the charge session data of an EV user even if the authorization data is not linkable. Thus, an approach for reducing the privacy risk of charge session data while still supporting the billing and load management functions could be required. Additionally, the automatic installation of the DAA PnC credentials still identifies the user's EV to the CPS and eMSP. While it was argued that this is required for a solution not to impact usability, more privacy-friendly solutions could be investigated and usability evaluated. With regard to security, the main restrictions of the solution are a result of the (assumed) lacking security features on the CS. Thus, e.g., the integration of a minimal trusted computing base on CSs could be further investigated. Additionally, further formal analysis is needed to verify the security properties of the solution.

Bibliography

[1] Mohamed Abomhara and Geir M Køien. “Security and privacy in the Internet of Things: Current status and open issues.” In: *2014 international conference on privacy and security in mobile systems (PRISMS)*. IEEE. 2014, pp. 1–8.

[2] Joseph Antoun, Mohammad Ekramul Kabir, Bassam Moussa, Ribal Atallah, and Chadi Assi. “A Detailed Security Assessment of the EV Charging Ecosystem.” In: *IEEE Network* (2020).

[3] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, P Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, et al. “The AVISPA tool for the automated validation of internet security protocols and applications.” In: *International conference on computer aided verification*. Springer. 2005, pp. 281–285.

[4] Will Arthur and David Challener. *A practical guide to TPM 2.0: Using the new trusted platform module in the new age of security*. Springer Nature, 2015.

[5] Article 29 Data Protection Working Party. *Opinion 05/2014 on Anonymisation Techniques*. WP 216. Apr. 2014. URL: https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf.

[6] Article 29 Data Protection Working Party. *Opinion 4/2007 on the concept of personal data*. WP 136. June 2007. URL: https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2007/wp136_en.pdf.

[7] Gildas Avoine, Iwen Coisel, and Tania Martin. “A privacy-restoring mechanism for offline RFID systems.” In: *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*. 2012, pp. 63–74.

[8] Gildas Avoine, Etienne Dysli, and Philippe Oechslin. “Reducing time complexity in RFID systems.” In: *International workshop on selected areas in cryptography*. Springer. 2005, pp. 291–306.

[9] Gildas Avoine, Cédric Lauradoux, and Tania Martin. “When compromised readers meet RFID.” In: *International Workshop on Information Security Applications*. Springer. 2009, pp. 36–50.

[10] Michael Barbaro and Tom Zeller Jr. *A Face Is Exposed for AOL Searcher No. 4417749*. Ed. by The New York Times. Aug. 2006. URL: https://www.nytimes.com/2006/08/09/technology/09aol.html (visited on 2020-11-21).

[11] Razvan Barbulescu and Sylvain Duquesne. “Updating key size estimations for pairings.” In: *Journal of Cryptology* 32.4 (2019), pp. 1298–1336.

[12] Elaine Barker. “NIST Special Publication 800-57 Part 1, Recommendation for Key Management: General.” In: *Revision 5* (May 2020).

[13] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. *XML signature syntax and processing version 1.1*. W3C Recommendation. Apr. 2013. URL: https://www.w3.org/TR/xmldsig-core1/.

[14] Daniel Barth-Jones. “The ’Re-Identification’ of Governor William Weld’s Medical Information: A Critical Re-Examination of Health Data Identification Risks and Privacy Protections.” In: *Then and Now* (July 2012).

[15] Robert Basmadjian, Benedikt Kirpes, Jan Mrkos, and Marek Cuchỳ. “A Reference Architecture for Interoperable Reservation Systems in Electric Vehicle Charging.” In: *Smart Cities* 3.4 (2020), pp. 1405–1427.

[16] James Bennett, Stan Lanning, et al. “The netflix prize.” In: *Proceedings of KDD cup and workshop*. Vol. 2007. New York. 2007.

[17] Bradley Berman. *The ISO standard for electric-vehicle “Plug-and-Charge” faces security concerns*. Ed. by SAE International. Retrieved on 04.09.2020. Aug. 2020. URL: https://www.sae.org/news/2020/08/iso-ev-plug-and-charge-standard-faces-security-concerns.

[18] Tamas Bisztray and Nils Gruschka. “Privacy Impact Assessment: Comparing Methodologies with a Focus on Practicality.” In: *Secure IT Systems*. Ed. by Aslan Askarov, René Rydhof Hansen, and Willard Rafnsson. Cham: Springer International Publishing, 2019, pp. 3–19. ISBN: 978-3-030-35055-0.

[19] The European Data Protection Board. *Guidelines 1/2020 on processing personal data in the context of connected vehicles and mobility related applications*. Adopted - version for public consultation. Jan. 2020. URL: https://edpb.europa.eu/our-work-tools/public-consultations-art-704/2020/guidelines-12020-processing-personal-data-context_de.

[20] Dan Boneh, Xavier Boyen, and Hovav Shacham. “Short group signatures.” In: *Annual International Cryptology Conference*. Springer. 2004, pp. 41–55.

[21] Matthew Bradbury, Phillip Taylor, Ugur Ilker Atmaca, Carsten Maple, and Nathan Griffiths. “Privacy Challenges With Protecting Live Vehicular Location Context.” In: *IEEE Access* 8 (2020), pp. 207465–207484.

[22] Ernie Brickell, Jan Camenisch, and Liqun Chen. “Direct anonymous attestation.” In: *Proceedings of the 11th ACM conference on Computer and communications security*. 2004, pp. 132–145.

[23] Ernie Brickell, Liqun Chen, and Jiangtao Li. “A (corrected) DAA scheme using batch proof and verification.” In: *International Conference on Trusted Systems*. Springer. 2011, pp. 304–337.

[24] Ernie Brickell, Liqun Chen, and Jiangtao Li. “A new direct anonymous attestation scheme from bilinear maps.” In: *International Conference on Trusted Computing*. Springer. 2008, pp. 166–178.

[25] BSI. *IT-Grundschutz-Kompendium*. Tech. rep. Federal Office for Information Security, Feb. 2020.

[26] Jan Camenisch and Anna Lysyanskaya. “Signature schemes and anonymous credentials from bilinear maps.” In: *Annual International Cryptology Conference*. Springer. 2004, pp. 56–72.

[27] Jan Camenisch and Els Van Herreweghen. “Design and implementation of the idemix anonymous credential system.” In: *Proceedings of the 9th ACM conference on Computer and communications security*. 2002, pp. 21–30.

[28] Alket Cecaj, Marco Mamei, and Franco Zambonelli. “Re-identification and information fusion between anonymized CDR and social network data.” In: *Journal of Ambient Intelligence and Humanized Computing* 7.1 (2016), pp. 83–96.

[29] CharIN. *Position Paper of Charging Interface Initiative e.V. – CP for ISO 15118 V2G PKI*. Charging Interface Initiative e.V., Aug. 2020. URL: https://www.charinev.org/media/position-papers-regulation/ (visited on 2020-12-10).

[30] David Chaum and Eugène Van Heyst. “Group signatures.” In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1991, pp. 257–265.

[31] Liqun Chen and Jiangtao Li. “Flexible and scalable digital signatures in TPM 2.0.” In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 37–48.

[32] Liqun Chen, Paul Morrissey, and Nigel P Smart. *DAA: Fixing the pairing based protocols*. Tech. rep. Cryptology ePrint Archive, Report 2009/198, 2009.

[33] Liqun Chen, Dan Page, and Nigel P Smart. “On the design and implementation of an efficient DAA scheme.” In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2010, pp. 223–237.

[34] European Commission. “Commission Directive 2008/63/EC of 20 June 2008 on competition in the markets in telecommunications terminal equipment (Codified version).” In: *OJ* L 162 (June 2008), pp. 20–26. URL: https://eur-lex.europa.eu/eli/dir/2008/63/oj.

[35] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and Tim Polk. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. RFC Editor, May 2008. URL: http://www.rfc-editor.org/rfc/rfc5280.txt.

[36] Crispin Cowan, F Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. “Buffer overflows: Attacks and defenses for the vulnerability of the decade.” In: *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX’00*. Vol. 2. IEEE. 2000, pp. 119–129.

[37] Lorrie Faith Cranor. “Necessary but not sufficient: Standardized mechanisms for privacy notice and choice.” In: *J. on Telecomm. & High Tech. L.* 10 (2012).

[38] Mathias Dalheimer. *Chaos Computer Club hacks e-motor charging stations*. Dec. 2017. URL: https://www.ccc.de/en/updates/2017/e-motor (visited on 2019-03-30).

[39] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. “Unique in the crowd: The privacy bounds of human mobility.” In: *Scientific reports* 3 (2013), p. 1376.

[40] Yves-Alexandre De Montjoye, Laura Radaelli, Vivek Kumar Singh, et al. “Unique in the shopping mall: On the reidentifiability of credit card metadata.” In: *Science* 347.6221 (2015), pp. 536–539.

[41] Yoni De Mulder, George Danezis, Lejla Batina, and Bart Preneel. “Identification via location-profiling in GSM networks.” In: *Proceedings of the 7th ACM workshop on Privacy in the electronic society*. 2008, pp. 23–32.

[42] Kevin Delmolino, Mitchell Arnett, Ahmed Kosba, Andrew Miller, and Elaine Shi. “Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab.” In: *International conference on financial cryptography and data security*. Springer. 2016, pp. 79–94.

[43] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. “A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements.” In: *Requirements Engineering* 16.1 (2011), pp. 3–32.

[44] Tim Dierks and Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. RFC Editor, Aug. 2008. URL: http://www.rfc-editor.org/rfc/rfc5246.txt.

[45] Tassos Dimitriou. “A lightweight RFID protocol to protect against traceability and cloning attacks.” In: *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM’05)*. IEEE. 2005, pp. 59–66.

[46] Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. “The role of the adversary model in applied security research.” In: *Computers & Security* 81 (2019), pp. 156–181.

[47] Danny Dolev and Andrew Yao. “On the security of public key protocols.” In: *IEEE Transactions on information theory* 29.2 (1983), pp. 198–208.

[48] Matt Duckham and Lars Kulik. “Location privacy and location-aware computing.” In: *Dynamic & mobile GIS: investigating change in space and time* 3 (2006), pp. 35–51.

[49] Khaled El Emam, Fida Kamal Dankar, Romeo Issa, Elizabeth Jonker, Daniel Amyot, Elise Cogo, Jean-Pierre Corriveau, Mark Walker, Sadrul Chowdhury, Regis Vaillancourt, et al. “A globally optimal k-anonymity method for the de-identification of health data.” In: *Journal of the American Medical Informatics Association* 16.5 (2009), pp. 670–682.

[50] ElaadNL. *EV Related Protocol Study*. Arnhem, The Netherlands. Jan. 2017. URL: https://www.elaad.nl/research/ev-related-protocol-study/ (visited on 2020-05-25).

[51] ElaadNL. *Exploring the Public Key Infrastructure for ISO 15118 in the EV Charging Ecosystem*. Arnhem, The Netherlands. Nov. 2018. URL: https://www.elaad.nl/news/publication-exploring-the-public-key-infrastructure-for-iso-15118-in-the-ev-charging-ecosystem/ (visited on 2020-09-21).

[52] ElaadNL. *Smart Charging Guide*. Arnhem, The Netherlands. June 2020. URL: https://www.elaad.nl/news/smart-charging-guide-english-language-edition-now-available/ (visited on 2020-11-28).

[53] eMI^3. *V1.0 Electric Vehicle ICT Interface Specifications - Part 2 : Business Objects*. Version 1.00. Apr. 2015. URL: https://emi3group.com/documents-links/.

[54] ENISA. *Cyber Security and Resilience of smart cars*. Good practices and recommendations. Heraklion, Greece, Dec. 2016.

[55] Council of the European Union European Parliament. “Directive (EU) 2015/1535 of the European Parliament and of the Council of 9 September 2015 laying down a procedure for the provision of information in the field of technical regulations and of rules on Information Society services.” In: *OJ* L 241 (Sept. 2015), pp. 1–15. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32015L1535.

[56] Council of the European Union European Parliament. “Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications).” In: *OJ* L 201 (July 2002), pp. 37–47. URL: https://eur-lex.europa.eu/eli/dir/2002/58/oj.

[57] Council of the European Union European Parliament. “Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.” In: *OJ* L 281 (Oct. 1995), pp. 31–50. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:31995L0046.

[58] Council of the European Union European Parliament. “Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation).” In: *OJ* L 119 (Apr. 2016), pp. 1–88. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[59] Marouane Fazouane, Henning Kopp, Rens W van der Heijden, Daniel Le Métayer, and Frank Kargl. “Formal verification of privacy properties in electric vehicle charging.” In: *International Symposium on Engineering Secure Software and Systems*. Springer. 2015, pp. 17–33.

[60] Ian Fette and Alexey Melnikov. *The WebSocket Protocol*. RFC 6455. RFC Editor, Dec. 2011. URL: http://www.rfc-editor.org/rfc/rfc6455.txt.

[61] Anatoliy M Firer, Dmitry S Silnov, and Anton A Horoshiy. “Security Analysis of Russian Transport Cards.” In: *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE. 2020, pp. 2054–2056.

[62] John Franks, Phillip M. Hallam-Baker, Jeffery L. Hostetler, Scott D. Lawrence, Paul J. Leach, Ari Luotonen, and Lawrence C. Stewart. *HTTP Authentication: Basic and Digest Access Authentication*. RFC 2617. RFC Editor, June 1999. URL: http://www.rfc-editor.org/rfc/rfc2617.txt.

[63] Julien Freudiger, Reza Shokri, and Jean-Pierre Hubaux. “Evaluating the privacy risk of location-based services.” In: *International conference on financial cryptography and data security*. Springer. 2011, pp. 31–46.

[64] Gerald Friedland, Gregor Maier, Robin Sommer, and Nicholas Weaver. “Sherlock holmes’ evil twin: on the impact of global inference for online privacy.” In: *Proceedings of the 2011 New Security Paradigms Workshop*. 2011, pp. 105–114.

[65] Gerald Friedland and Robin Sommer. “Cybercasing the Joint: On the Privacy Implications of Geo-Tagging.” In: *HotSec*. 2010, pp. 1–6.

[66] Andreas Fuchs, Dustin Kern, Christoph Krauß, and Maria Zhdanova. “HIP: HSM-Based Identities for Plug-and-Charge.” In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*. ARES ’20. Virtual Event, Ireland: Association for Computing Machinery, 2020. ISBN: 9781450388337. DOI: 10.1145/3407023.3407066. URL: https://doi.org/10.1145/3407023.3407066.

[67] Andreas Fuchs, Dustin Kern, Christoph Krauß, Maria Zhdanova, and Ronald Heddergott. “HIP-20: Integration of Vehicle-HSM-Generated Credentials into Plug-and-Charge Infrastructure.” In: *Computer Science in Cars Symposium*. CSCS ’20. Feldkirchen, Germany: Association for Computing Machinery, 2020. ISBN: 9781450376211. DOI: 10.1145/3385958.3430483. URL: https://doi.org/10.1145/3385958.3430483.

[68] Andreas Fuchs, Christoph Krauß, and Jürgen Repp. “Advanced remote firmware upgrades using TPM 2.0.” In: *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer. 2016, pp. 276–289.

[69] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. “De-anonymization attack on geolocated data.” In: *Journal of Computer and System Sciences* 80.8 (2014), pp. 1597–1614.

[70] Flavio D Garcia and Peter Van Rossum. “Modeling privacy for off-line RFID systems.” In: *International Conference on Smart Card Research and Advanced Applications*. Springer. 2010, pp. 194–208.

[71] Javier García-Villalobos, Inmaculada Zamora, José Ignacio San Martín, Francisco Javier Asensio, and Víctor Aperribay. “Plug-in electric vehicles in electric distribution networks: A review of smart charging approaches.” In: *Renewable and Sustainable Energy Reviews* 38 (2014), pp. 717–731.

[72] Simson L Garfinkel. “De-identification of personal information.” In: *National institute of standards and technology* (2015).

[73] Simson L Garfinkel, Ari Juels, and Ravikanth Pappu. “RFID privacy: An overview of problems and proposed solutions.” In: *IEEE Security & Privacy* 3.3 (2005), pp. 34–43.

[74] Dipayan P Ghosh, Robert J Thomas, and Stephen B Wicker. “A privacy-aware design for the vehicle-to-grid framework.” In: *2013 46th Hawaii International Conference on System Sciences*. IEEE. 2013, pp. 2283–2291.

[75] GIREVE. *eMIP Protocol*. v1.0.13. June 2020. URL: https://www.gireve.com/en/download.

[76] Philippe Golle. “Revisiting the uniqueness of simple demographics in the US population.” In: *Proceedings of the 5th ACM workshop on Privacy in electronic society*. 2006, pp. 77–80.

[77] Ralph Gross and Alessandro Acquisti. “Information revelation and privacy in online social networks.” In: *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*. 2005, pp. 71–80.

[78] Wenlin Han and Yang Xiao. “Privacy preservation for V2G networks in smart grid: A survey.” In: *Computer Communications* 91 (2016), pp. 17–28.

[79] Jane Henriksen-Bulmer and Sheridan Jeary. “Re-identification attacks—A systematic literature review.” In: *International Journal of Information Management* 36.6 (2016), pp. 1184–1192.

[80] Johannes Heurix, Peter Zimmermann, Thomas Neubauer, and Stefan Fenz. “A taxonomy for privacy enhancing technologies.” In: *Computers & Security* 53 (2015), pp. 1–17.

[81] Christina Höfer, Jonathan Petit, Robert Schmidt, and Frank Kargl. “POPCORN: privacy-preserving charging for eMobility.” In: *Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles*. 2013, pp. 37–48.

[82] Michael Howard and Steve Lipner. *The security development lifecycle*. Vol. 8. Microsoft Press Redmond, 2006.

[83] Jean-Pierre Hubaux, Srdjan Capkun, and Jun Luo. “The security and privacy of smart vehicles.” In: *IEEE Security & Privacy* 2.3 (2004), pp. 49–55.

[84] Hubject GmbH. *Ecosystem & PKI interfaces*. Retrieved on 21.09.2020. URL: https://support.hubject.com/hc/en-us/articles/360000360277-Ecosystem-PKI-interfaces.

[85] Hubject GmbH. *Hubject PKI*. 2020. URL: https://www.hubject.com/pki/ (visited on 2020-08-12).

[86] Hubject GmbH. *Hubject Plug&Charge Certificate Policy (CP) for the Hubject ISO 15118 V2G PKI*. Version 1.7. June 2020.

[87] Hubject GmbH. *Open InterCharge Protocol for Charge Point Operators*. Version 2.2. Mar. 2018. URL: https://www.hubject.com.

[88] Hubject GmbH. *Open InterCharge Protocol for Emobility Service Providers*. Version 2.2. Mar. 2018. URL: https://www.hubject.com.

[89] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo. “Cyber-physical systems security—A survey.” In: *IEEE Internet of Things Journal* 4.6 (2017), pp. 1802–1831.

[90] Shafiq Hussain, Asif Kamal, Shabir Ahmad, Ghulam Rasool, and Sajid Iqbal. “Threat modelling methodologies: a survey.” In: *Sci. Int.(Lahore)* 26.4 (2014), pp. 1607–1609.

[91] IEA. *Global EV Outlook 2019*. International Energy Agency, May 2019. URL: https://www.iea.org/reports/global-ev-outlook-2019 (visited on 2020-05-20).

[92] IEA. *Global EV Outlook 2020*. International Energy Agency, June 2020. URL: https://www.iea.org/reports/global-ev-outlook-2020 (visited on 2020-07-26).

[93] Infineon. *A safe for sensitive data in the car: Volkswagen relies on TPM from Infineon*. Jan. 2019. URL: https://www.infineon.com/cms/en/about-infineon/press/market-news/2019/INFATV201901-030.html (visited on 2019-03-29).

[94] IRENA. *Electric Vehicles: Technology Brief*. Feb. 2017. URL: https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2017/IRENA_Electric_Vehicles_2017.pdf (visited on 2020-05-20).

[95] ISO/IEC. *Information technology — Trusted platform module library — Part 1: Architecture*. ISO Standard 11889-1:2015. Geneva, Switzerland: ISO, Apr. 2016.

[96] ISO/IEC. *Road vehicles – Vehicle to grid communication interface – Part 1: General information and use-case definition*. ISO Standard 15118-1:2013. Geneva, Switzerland: ISO, Apr. 2013.

[97] ISO/IEC. *Road vehicles – Vehicle-to-Grid Communication Interface – Part 2: Network and application protocol requirements*. ISO Standard 15118-2:2014. Geneva, Switzerland: ISO, Apr. 2014.

[98] ISO/IEC. *Road vehicles – Vehicle-to-Grid Communication Interface – Part 2: Network and application protocol requirements*. ISO/DIS 15118-2:2018. Geneva, Switzerland: International Organization for Standardization, Dec. 2018.

[99] Joint Task Force. *Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy*. NIST Special Publication 800-37 Rev 2. NIST, Dec. 2018.

[100] Ari Juels. "RFID security and privacy: A research survey." In: *IEEE journal on selected areas in communications* 24.2 (2006), pp. 381–394.

[101] Ari Juels and John Brainard. "Soft blocking: Flexible blocker tags on the cheap." In: *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. 2004, pp. 1–7.

[102] Ari Juels, Ronald L Rivest, and Michael Szydlo. "The blocker tag: Selective blocking of RFID tags for consumer privacy." In: *Proceedings of the 10th ACM conference on Computer and communications security*. 2003, pp. 103–111.

[103] Ari Juels and Stephen A Weis. "Authenticating pervasive devices with human protocols." In: *Annual international cryptology conference*. Springer. 2005, pp. 293–308.

[104] European Court of Justice. *Judgment of the Court (Second Chamber) of 19 October 2016. Patrick Breyer v Bundesrepublik Deutschland. Request for a preliminary ruling from the Bundesgerichtshof. Reference for a preliminary ruling — Processing of personal data — Directive 95/46/EC — Article 2(a) — Article 7(f) — Definition of 'personal data' — Internet protocol addresses — Storage of data by an online media services provider — National legislation not permitting the legitimate interest pursued by the controller to be taken into account.* Case C-582/14. Oct. 2016. URL: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:62014CJ0582.

[105] Christos Kalloniatis, Evangelia Kavakli, and Stefanos Gritzalis. "Addressing privacy requirements in system design: the PriS method." In: *Requirements Engineering* 13.3 (2008), pp. 241–255.

[106] Mart van der Kam and Rudi Bekkers. *Comparative analysis of standardized protocols for EV roaming*. Report D6.1 for the evRoaming4EU project. Netherlands Knowledge Platform for Public Charging Infrastructure (NKL), May 2020.

[107] Mart van der Kam, Roland Ferweda, and Rudi NA Bekkers. "Developing roaming protocols for EV charging: Insights from the field." In: *8th Transport Research Arena TRA 2020*. 2020.

[108] Soo-Young Kang, Jong Hyuk Park, Muhammad Khurram Khan, and Jin Kwak. "Study on the common criteria methodology for secure ubiquitous environment construction." In: *Journal of Intelligent Manufacturing* 23.4 (2012), pp. 933–939.

[109] Adi Karahasanovic, Pierre Kleberger, and Magnus Almgren. "Adapting threat modeling methods for the automotive industry." In: *Proceedings of the 15th ESCAR Conference*. 2017, pp. 1–10.

[110] Timo Kasper, Ingo von Maurich, David Oswald, and Christof Paar. "Cloning cryptographic RFID cards for 25$." In: *5th Benelux workshop on information and system security. Nijmegen, Netherlands*. 2010.

[111] Vishnu Teja Kilari, Ruozhou Yu, Satyajayant Misra, and Guoliang Xue. "Robust Revocable Anonymous Authentication for Vehicle to Grid Communications." In: *IEEE Transactions on Intelligent Transportation Systems* 21.11 (2020), pp. 4845–4857.

[112] Dennis-Kenji Kipker and Christian Seipel. “Datenschutz in der E-Mobilität-Problemstellungen und mögliche Lösungsansätze.” In: *ATZelektronik* 14.3 (2019), pp. 44–49.

[113] Fabian Knirsch, Andreas Unterweger, and Dominik Engel. “Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions.” In: *Computer Science-Research and Development* 33.1-2 (2018), pp. 71–79.

[114] Heiko Knospe and Hartmut Pohl. “RFID security.” In: *Information Security Technical Report* 9.4 (2004), pp. 39–50. ISSN: 1363-4127. DOI: https://doi.org/10.1016/S1363-4127(05)70039-X. URL: http://www.sciencedirect.com/science/article/pii/S136341270570039X.

[115] Geir M Køien. “A privacy enhanced device access protocol for an IoT context.” In: *Security and Communication Networks* 9.5 (2016), pp. 440–450.

[116] Daniel Kouril, Ludek Matyska, and Michal Prochazka. “A robust and efficient mechanism to distribute certificate revocation information using the Grid Monitoring Architecture.” In: *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW’07)*. Vol. 1. IEEE. 2007, pp. 614–619.

[117] Irwin R Kramer. “The Birth of Privacy Law: A Century Since Warren and Brandeis.” In: *Cath. UL Rev.* 39 (1990).

[118] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104. RFC Editor, Feb. 1997. URL: http://www.rfc-editor.org/rfc/rfc2104.txt.

[119] Lucie Langer, Florian Skopik, Georg Kienesberger, and Qin Li. “Privacy issues of smart e-mobility.” In: *IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2013, pp. 6682–6687.

[120] Marc Langheinrich. “RFID and privacy.” In: *Security, Privacy, and Trust in Modern Data Management*. Springer, 2007, pp. 433–450.

[121] Hongyang Li, György Dan, and Klara Nahrstedt. “Portunes: Privacy-preserving fast authentication for dynamic electric vehicle charging.” In: *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE. 2014, pp. 920–925.

[122] Hongyang Li, György Dán, and Klara Nahrstedt. “Portunes+: Privacy-preserving fast authentication for dynamic electric vehicle charging.” In: *IEEE Transactions on Smart Grid* 8.5 (2016), pp. 2305–2313.

[123] Ticyan Li and Guilin Wang. “Security analysis of two ultra-lightweight RFID authentication protocols.” In: *IFIP international information security conference*. Springer. 2007, pp. 109–120.

[124] Joseph K Liu, Willy Susilo, Tsz Hon Yuen, Man Ho Au, Junbin Fang, Zoe L Jiang, and Jianying Zhou. “Efficient privacy-preserving charging station reservation system for electric vehicles.” In: *The Computer Journal* 59.7 (2016), pp. 1040–1053.

[125] Javier Lopez, Ruben Rios, Feng Bao, and Guilin Wang. “Evolving privacy: From sensors to the Internet of Things.” In: *Future Generation Computer Systems* 75 (2017), pp. 46–57.

[126] Gavin Lowe. “A hierarchy of authentication specifications.” In: *Proceedings 10th Computer Security Foundations Workshop*. IEEE. 1997, pp. 31–43.

[127] Jesus Luna, Neeraj Suri, and Ioannis Krontiris. “Privacy-by-design based on quantitative threat modeling.” In: *2012 7th International Conference on Risks and Security of Internet and Systems (CRiSIS)*. IEEE. 2012, pp. 1–8.

[128] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. “A study of MAC address randomization in mobile devices and when it fails.” In: *Proceedings on Privacy Enhancing Technologies* 2017.4 (2017), pp. 365–383.

[129] Erika McCallister, Timothy Grance, and Karen Scarfone. *Guide to protecting the confidentiality of personally identifiable information (PII)*. NIST Special Publication 800-122. NIST, Apr. 2010.

[130] Nancy R Mead and Ted Stehney. "Security quality requirements engineering (SQUARE) methodology." In: *ACM SIGSOFT Software Engineering Notes* 30.4 (2005), pp. 1–7.

[131] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. "The TAMARIN prover for the symbolic analysis of security protocols." In: *International Conference on Computer Aided Verification*. Springer. 2013, pp. 696–701.

[132] David Molnar and David Wagner. "Privacy and security in library RFID: Issues, practices, and architectures." In: *Proceedings of the 11th ACM conference on Computer and communications security*. 2004, pp. 210–219.

[133] Jean-Philippe Monteuuis, Jonathan Petit, Jun Zhang, Houda Labiod, Stefano Mafrica, and Alain Servel. "Attacker model for connected and automated vehicles." In: *ACM COMPUTER SCIENCE IN CARS SYMPOSIUM*. 2018.

[134] Mustafa A Mustafa, Ning Zhang, Georgios Kalogridis, and Zhong Fan. "Roaming electric vehicle charging and billing: An anonymous multi-user protocol." In: *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE. 2014, pp. 939–945.

[135] Suvda Myagmar, Adam J Lee, and William Yurcik. "Threat modeling as a basis for security requirements." In: *Symposium on requirements engineering for information security (SREIS)*. Vol. 2005. 2005, pp. 1–8.

[136] Arvind Narayanan and Vitaly Shmatikov. "Myths and fallacies of "personally identifiable information"." In: *Communications of the ACM* 53.6 (2010), pp. 24–26.

[137] Arvind Narayanan and Vitaly Shmatikov. "Robust de-anonymization of large sparse datasets." In: *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE. 2008, pp. 111–125.

[138] Thomas Narten, Richard Draves, and Suresh Krishnan. *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. RFC 4941. RFC Editor, Sept. 2007. URL: http://www.rfc-editor.org/rfc/rfc4941.txt.

[139] Gregory S Nelson. "Practical implications of sharing data: a primer on data privacy, anonymization, and de-identification." In: *SAS Global Forum Proceedings*. 2015, pp. 1–23.

[140] Netflix, Inc. *The Netflix Prize Rules*. Oct. 2006. URL: https://www.netflixprize.com/rules.html (visited on 2020-11-21).

[141] Helen Nissenbaum. "Privacy as contextual integrity." In: *Wash. L. Rev.* 79 (2004).

[142] NKL. *Open Charge Point Interface*. Version 2.2-d2. Netherlands Knowledge Platform for Charging Infrastructure. June 2020. URL: https://ocpi-protocol.org.

[143] Magnus Nystrom and Burt Kaliski. *PKCS #10: Certification Request Syntax Specification Version 1.7*. RFC 2986. RFC Editor, Nov. 2000. URL: http://www.rfc-editor.org/rfc/rfc2986.txt.

[144] OCA. *Improved security for OCPP 1.6-J*. White Paper. Arnhem, Netherlands: Open Charge Alliance, Mar. 2020. URL: http://www.openchargealliance.org/protocols/ocpp/ocpp-16/.

[145] OCA. *Open Charge Point Protocol 1.6 edition 2*. Open Standard. Arnhem, Netherlands: Open Charge Alliance, Sept. 2017. URL: http://www.openchargealliance.org/protocols/ocpp/ocpp-16/.

[146] OCA. *Open Charge Point Protocol 2.0.1 - Part 0 - Introduction*. Open Standard. Arnhem, Netherlands: Open Charge Alliance, Mar. 2020. URL: https://www.openchargealliance.org/protocols/ocpp-201/.

[147] OCA. *Open Charge Point Protocol 2.0.1 - Part 1 - Architecture & Topology*. Open Standard. Arnhem, Netherlands: Open Charge Alliance, Mar. 2020. URL: https://www.openchargealliance.org/protocols/ocpp-201/.

[148] OCA. *Open Charge Point Protocol 2.0.1 - Part 2 - Specification*. Open Standard. Arnhem, Netherlands: Open Charge Alliance, Mar. 2020. URL: https://www.openchargealliance.org/protocols/ocpp-201/.

[149] OCA. *Open Charge Point Protocol JSON 1.6*. Open Standard. Arnhem, Netherlands: Open Charge Alliance, Oct. 2015. URL: http://www.openchargealliance.org/protocols/ocpp/ocpp-16/.

[150] OCA. *Open Charge Point Protocol SOAP 1.6*. Open Standard. Arnhem, Netherlands: Open Charge Alliance, Oct. 2015. URL: http://www.openchargealliance.org/protocols/ocpp/ocpp-16/.

[151] Office of Management and Budget. *Managing Information as a Strategic Resource*. Circular A-130. OMB, July 2016.

[152] Miyako Ohkubo, Koutarou Suzuki, Shingo Kinoshita, et al. "Cryptographic approach to "privacy-friendly" tags." In: *RFID privacy workshop*. Vol. 82. Cambridge, USA. 2003.

[153] Paul Ohm. "Broken promises of privacy: Responding to the surprising failure of anonymization." In: *UCLA l. Rev.* 57 (2009).

[154] Adebayo Omotosho, Benjamin Ayemlo Haruna, and Olayemi Mikail Olaniyi. "Threat modeling of internet of things health devices." In: *Journal of Applied Security Research* 14.1 (2019), pp. 106–121.

[155] Simone Orcioni and Massimo Conti. "EV Smart Charging with Advance Reservation Extension to the OCPP Standard." In: *Energies* 13.12 (2020), p. 3263.

[156] Justin D Osborn and David C Challener. "Trusted platform Module evolution." In: *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)* 32.2 (2013), pp. 536–543.

[157] RK Pateriya and Sangeeta Sharma. "The evolution of RFID security and privacy: a research survey." In: *2011 International Conference on Communication Systems and Network Technologies*. IEEE. 2011, pp. 115–119.

[158] AJ Paverd, Andrew Martin, and Ian Brown. "Modelling and automatically analysing privacy properties for honest-but-curious adversaries." In: *Tech. Rep.* (2014).

[159] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M Estévez-Tapiador, and Arturo Ribagorda. "LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags." In: *Proc. of 2nd Workshop on RFID Security*. Vol. 6. 2006.

[160] Jonathan Petit, Michael Feiri, and Frank Kargl. "Revisiting attacker model for smart vehicles." In: *2014 IEEE 6th International Symposium on Wireless Vehicular Communications (WiVeC 2014)*. IEEE. 2014, pp. 1–5.

[161] Andreas Pfitzmann and Marit Hansen. *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*. v0.34. Aug. 2010. URL: https://dud.inf.tu-dresden.de/Anon_Terminology.shtml (visited on 2020-05-24).

[162] Victoria Y. Pillitteri and Tanya L. Brewer. *Guidelines for Smart Grid Cybersecurity: Volume 2 - Privacy and the Smart Grid*. NIST Interagency/Internal Report (NISTIR) - 7628 Rev 1. NIST, Sept. 2014.

[163] Bradley Potteiger, Goncalo Martins, and Xenofon Koutsoukos. “Software and attack centric integrated threat modeling for quantitative risk assessment.” In: *Proceedings of the Symposium and Bootcamp on the Science of Security*. 2016, pp. 99–108.

[164] Nadezhda Purtova. “The law of everything. Broad concept of personal data and future of EU data protection law.” In: *Law, Innovation and Technology* 10.1 (2018), pp. 40–81.

[165] Laurencas Raslavičius, Brian Azzopardi, Artūras Keršys, Martynas Starevičius, Žilvinas Bazaras, and Rolandas Makaras. “Electric vehicles challenges and opportunities: Lithuanian review.” In: *Renewable and Sustainable Energy Reviews* 42 (2015), pp. 786–800.

[166] Slobodan Ribaric, Aladdin Ariyaeeinia, and Nikola Pavesic. “De-identification for privacy protection in multimedia content: A survey.” In: *Signal Processing: Image Communication* 47 (2016), pp. 131–151.

[167] Yumi Sakemi, Tetsutaro Kobayashi, Tsunekazu Saito, and Riad Wahby. *Pairing-Friendly Curves*. Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-09. IETF Secretariat, Nov. 2020. URL: http://www.ietf.org/internet-drafts/draft-irtf-cfrg-pairing-friendly-curves-09.txt.

[168] Stefan Santesson, Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. RFC 6960. RFC Editor, June 2013. URL: http://www.rfc-editor.org/rfc/rfc6960.txt.

[169] Bruce Schneier. “Attack trees.” In: *Dr. Dobb’s journal* 24.12 (1999), pp. 21–29.

[170] Paul M Schwartz and Daniel J Solove. “The PII problem: Privacy and a new concept of personally identifiable information.” In: *NYUL rev.* 86 (2011).

[171] Yun Shen and Siani Pearson. “Privacy enhancing technologies: A review.” In: (2011).

[172] Nataliya Shevchenko, Timothy A Chick, Paige O’Riordan, Thomas P Scanlon, and Carol Woody. *Threat modeling: a summary of available methods*. Tech. rep. Carnegie Mellon University Software Engineering Institute Pittsburgh United …, 2018.

[173] Adam Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[174] Guttorm Sindre and Andreas L Opdahl. “Templates for misuse case description.” In: *Proceedings of the 7th International Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ’2001), Switzerland*. Citeseer. 2001.

[175] Laurens Sion, Kim Wuyts, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. “Interaction-based privacy threat elicitation.” In: *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE. 2018, pp. 79–86.

[176] John Smart and Stephen Schey. “Battery electric vehicle driving and charging behavior observed early in the EV project.” In: *SAE International Journal of Alternative Powertrains* 1.1 (2012), pp. 27–33.

[177] smartlab. *June’s Feature Friday #4: Going Live*. June 2020. URL: http://www.ochp.eu/2020/06/junes-feature-friday-4-going-live/ (visited on 2020-07-21).

[178] smartlab. *OCHPdirect extension to the Open Clearing House Protocol*. Version 0.2. Aug. 2016. URL: http://www.ochp.eu.

[179] smartlab. *Open Clearing House Protocol*. Version 1.4. Aug. 2016. URL: http://www.ochp.eu.

[180] Daniel J Solove. “A taxonomy of privacy.” In: *U. Pa. L. Rev.* 154 (2006).

[181] Daniel J Solove. “Conceptualizing privacy.” In: *Calif. L. Rev.* 90 (2002).

[182] Sarah Spiekermann and Lorrie Faith Cranor. “Engineering privacy.” In: *IEEE Transactions on software engineering* 35.1 (2008), pp. 67–82.

[183] Matthias Spöttle, Korinna Jörling, Matthias Schimmel, Maarten Staats, Logan Grizzel, Lisa Jerram, William Drier, and John Gartner. *Research for TRAN Committee – Charging infrastructure for electric road vehicles*. Study. Brussels: European Parliament, Policy Department for Structural and Cohesion Policies, June 2018.

[184] Mark Stegelmann and Dogan Kesdogan. "Design and evaluation of a privacy-preserving architecture for vehicle-to-grid interaction." In: *European Public Key Infrastructure Workshop*. Springer. 2011, pp. 75–90.

[185] Mark Stegelmann and Dogan Kesdogan. "Location privacy for vehicle-to-grid interaction through battery management." In: *2012 Ninth International Conference on Information Technology-New Generations*. IEEE. 2012, pp. 373–378.

[186] Latanya Sweeney. "k-anonymity: A model for protecting privacy." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 557–570.

[187] Latanya Sweeney. "Simple demographics often identify people uniquely." In: *Health (San Francisco)* 671.2000 (2000), pp. 1–34.

[188] Irfan Syamsuddin, Tharam Dillon, Elizabeth Chang, and Song Han. "A survey of RFID authentication protocols based on hash-chain method." In: *2008 Third International Conference on Convergence and Hybrid Information Technology*. Vol. 2. IEEE. 2008, pp. 559–564.

[189] CE Sandy Thomas. "Transportation options in a carbon-constrained world: Hybrids, plug-in hybrids, biofuels, fuel cell electric vehicles, and battery electric vehicles." In: *International Journal of hydrogen energy* 34.23 (2009), pp. 9279–9296.

[190] Jonel Timbergen. *The Role Standardization Plays in Premium EV Charging*. Sept. 2018. URL: https://www.hubject.com/wp-content/uploads/2018/09/whitepaper_iso15118.pdf (visited on 2020-05-20).

[191] Trusted Computing Group. *TCG EK Credential Profile*. Specification Ver. 2.3 - Rev. 2.0. July 2020.

[192] Trusted Computing Group. *TCG TPM v2.0 Provisioning Guidance*. Guidance Ver. 1.0 - Rev. 1.0. Mar. 2017.

[193] Trusted Computing Group. *TPM Main - Part 1 Design Principles*. Specification Version 1.2 - Rev. 116. Mar. 2011.

[194] Trusted Computing Group. *Trusted Computing Platform Alliance (TCPA)*. Main Specification Version 1.1b. Feb. 2002.

[195] Trusted Computing Group. *Trusted Platform Module Library - Part 1: Architecture*. Specification Family 2.0 - Rev. 01.38. Sept. 2016.

[196] Trusted Computing Group. *Trusted Platform Module Library - Part 2: Structures*. Specification Family 2.0 - Rev. 01.38. Sept. 2016.

[197] Trusted Computing Group. *Trusted Platform Module Library - Part 3: Commands*. Specification Family 2.0 - Rev. 01.38. Sept. 2016.

[198] Richard C Turkington. "Legacy of the Warren and Brandeis article: The emerging unencumbered constitutional right to informational privacy." In: *N. Ill. UL Rev.* 10 (1990).

[199] U.S. Department of Health and Human Services–Office for Civil Rights. "HIPAA Administrative Simplification." In: *Regulation Text* 45 CFR Parts 160, 162, and 164 (Mar. 2013).

[200] "U.S. Health insurance portability and accountability act of 1996." In: *Public L. No. 104-191* 110 Stat. 1936 (Aug. 1996).

[201] Félicien Vallet. “The GDPR and Its Application in Connected Vehicles—Compliance and Good Practices.” In: *Electronic Components and Systems for Automotive Applications*. Springer, 2019, pp. 245–254.

[202] VDE. *Handling of certificates for electric vehicles, charging infrastructure and backend systems within the framework of ISO 15118*. VDE-AR-E 2802-100-1:2019-12. Dec. 2019.

[203] Samuel D Warren and Louis D Brandeis. “The right to privacy.” In: *Harvard law review* (1890), pp. 193–220.

[204] Stephen A Weis, Sanjay E Sarma, Ronald L Rivest, and Daniel W Engels. “Security and privacy aspects of low-cost radio frequency identification systems.” In: *Security in pervasive computing*. Springer, 2004, pp. 201–212.

[205] Stephan Wesemeyer, Christopher J.P. Newton, Helen Treharne, Liqun Chen, Ralf Sasse, and Jorden Whitefield. “Formal Analysis and Implementation of a TPM 2.0-Based Direct Anonymous Attestation Scheme.” In: *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. ASIA CCS ’20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 784–798. ISBN: 9781450367509. DOI: 10.1145/3320269.3372197. URL: https://doi.org/10.1145/3320269.3372197.

[206] Alan F. Westin. *Privacy and Freedom*. New York: Ig Publishing, 1967.

[207] Jorden Whitefield, Liqun Chen, Ralf Sasse, Steve Schneider, Helen Treharne, and Stephan Wesemeyer. “A symbolic analysis of ecc-based direct anonymous attestation.” In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2019, pp. 127–141.

[208] Stephen B Wicker. “The loss of location privacy in the cellular age.” In: *Communications of the ACM* 55.8 (2012), pp. 60–68.

[209] Kim Wuyts. “Privacy Threats in Software Architectures.” PhD thesis. Arenberg Doctoral School - KU Leuven, 2015. URL: https://lirias.kuleuven.be/retrieve/295669.

[210] Kim Wuyts, Riccardo Scandariato, and Wouter Joosen. “Empirical evaluation of a privacy-focused threat modeling methodology.” In: *Journal of Systems and Software* 96 (2014), pp. 122–138.

[211] Kim Wuyts, Riccardo Scandariato, and Wouter Joosen. *LIND(D)UN privacy threat tree catalog*. Version 2.0. Sept. 2014. URL: https://www.linddun.org/downloads (visited on 2020-07-24).

[212] Kim Wuyts, Dimitri Van Landuyt, Aram Hovsepyan, and Wouter Joosen. “Effective and efficient privacy threat modeling through domain refinements.” In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. 2018, pp. 1175–1178.

[213] Li Xi, Kang Yang, Zhenfeng Zhang, and Dengguo Feng. “DAA-related APIs in TPM 2.0 revisited.” In: *International Conference on Trust and Trustworthy Computing*. Springer. 2014, pp. 1–18.

[214] Chen Xiaofeng and Feng Dengguo. “Direct anonymous attestation for next generation TPM.” In: *Journal of Computers* 3.12 (2008), pp. 43–50.

[215] Koen Yskout, Thomas Heyman, Dimitri Van Landuyt, Laurens Sion, Kim Wuyts, and Wouter Joosen. “Threat modeling: from infancy to maturity.” In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 2020, pp. 9–12.

[216] Fatih Yucel, Kemal Akkaya, and Eyuphan Bulut. “Efficient and privacy preserving supplier matching for electric vehicle charging.” In: *Ad Hoc Networks* 90 (2019), p. 101730.

[217] Daniel Zelle, Markus Springer, Maria Zhdanova, and Christoph Krauß. “Anonymous charging and billing of electric vehicles.” In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. 2018, pp. 1–10.

[218] Tianyu Zhao, Chi Zhang, Lingbo Wei, and Yanchao Zhang. “A secure and privacy-preserving payment system for Electric vehicles.” In: *2015 IEEE International Conference on Communications (ICC)*. IEEE. 2015, pp. 7280–7285.

[219] Liang Zhu, Johanna Amann, and John Heidemann. “Measuring the latency and pervasiveness of TLS certificate revocation.” In: *International Conference on Passive and Active Network Measurement*. Springer. 2016, pp. 16–29.

Appendices

A. EV Charging DFDs

The figures in this section show the DFDs that were used as the basis for the STRIDE- and LINDDUN-based treat analysis in Section 5 (cf. *Step 5* of the threat modeling process as described in Section 2.3.2). The external entities, processes, data flows, and data stores represent the system model as described in Section 4. The trust boundaries represent the adversaries and assumptions as discussed in Section 5.

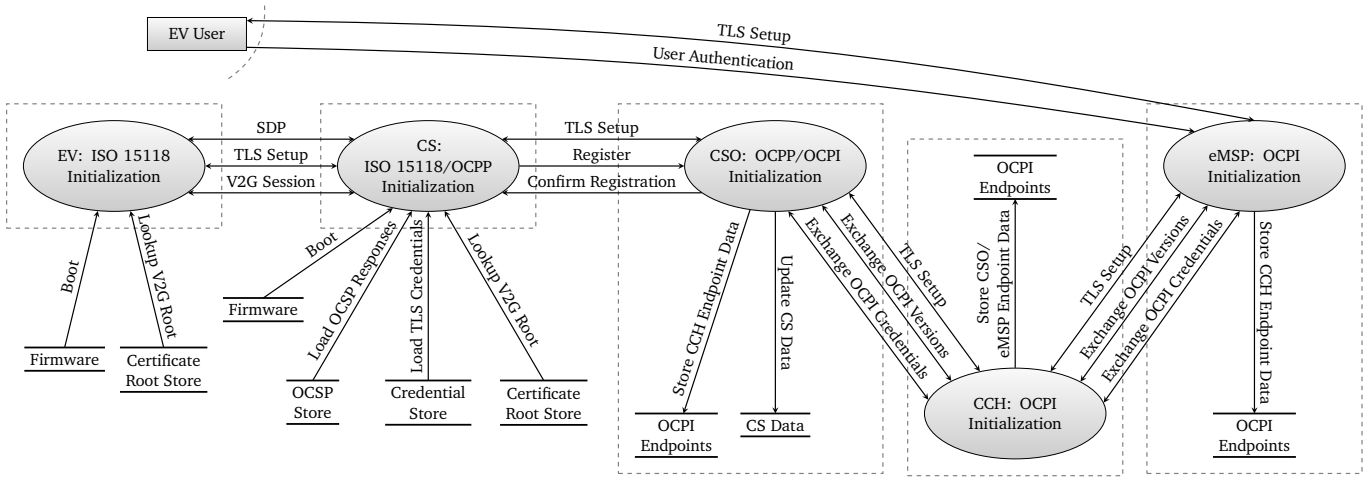


Figure A.1.: EV Charging DFD - Initial Communication Setup

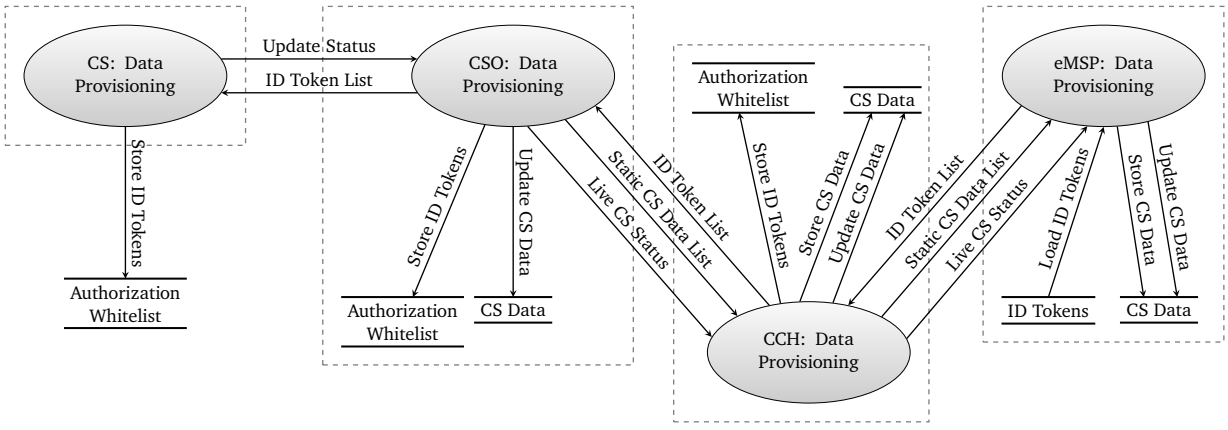


Figure A.2.: EV Charging DFD - Data Provisioning

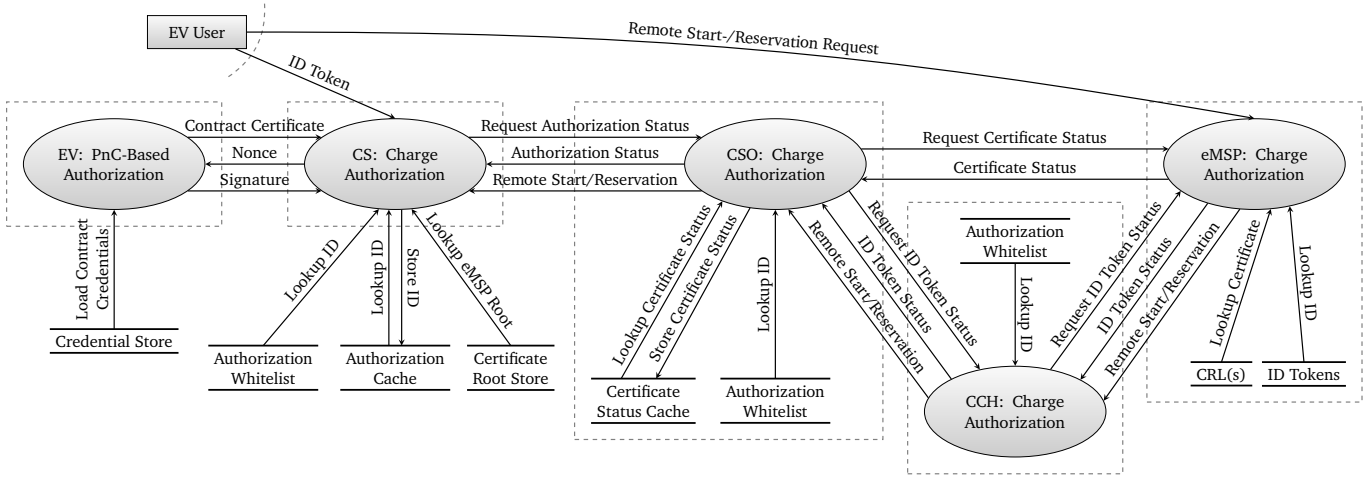


Figure A.3.: EV Charging DFD - Charge Authorization

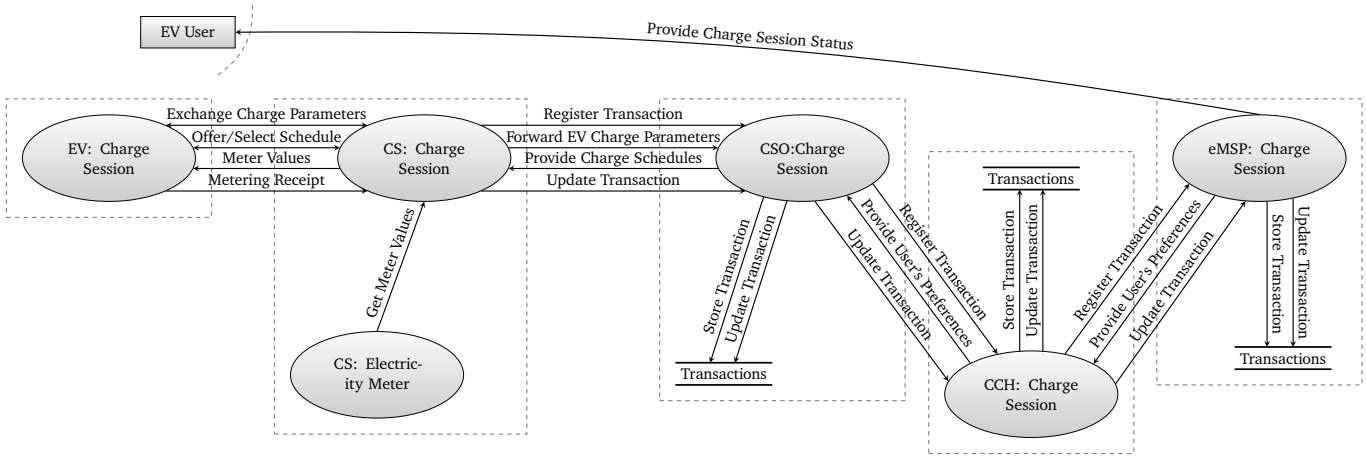


Figure A.4.: EV Charging DFD - Charge Session

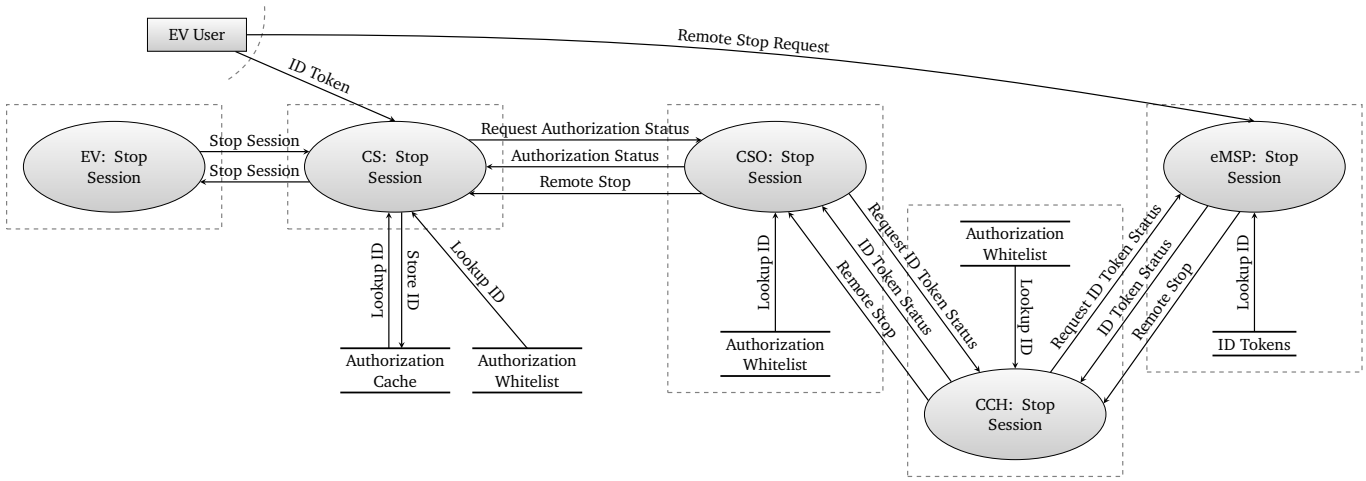


Figure A.5.: EV Charging DFD - Stop Session Authorization

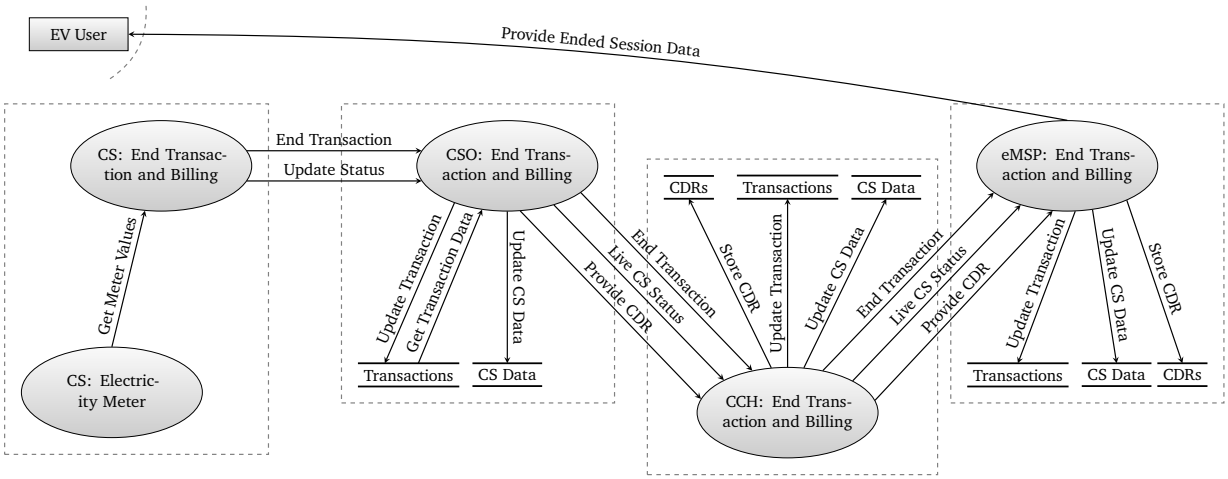


Figure A.6.: EV Charging DFD - End Transaction and Billing

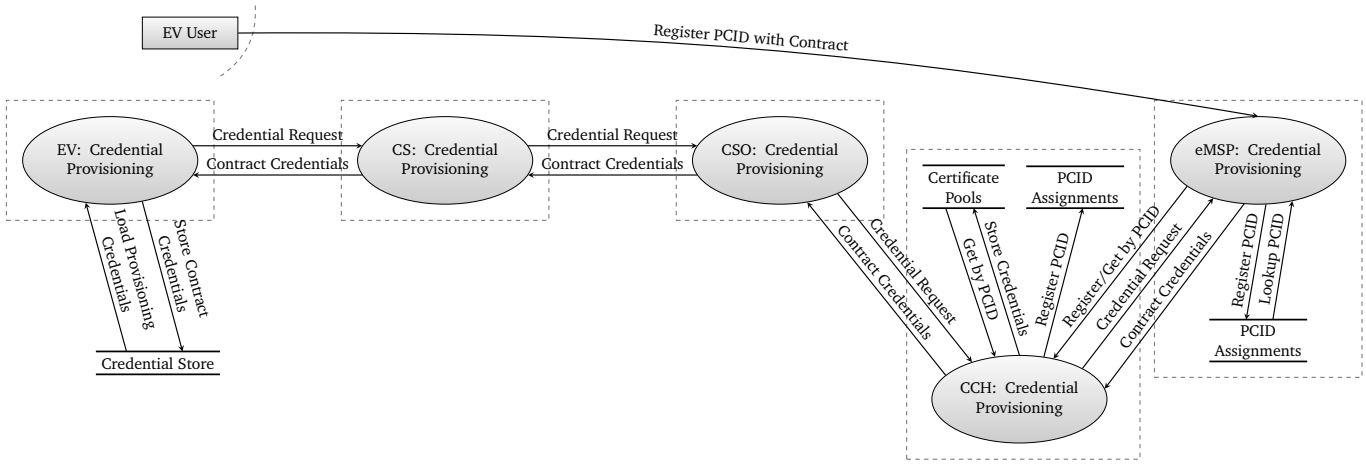


Figure A.7.: EV Charging DFD - User Credential Provisioning

B. ISO 15118 Message Changes

The following listings show the changed ISO 15118 messages that were used for the proof-of-concept implementation (cf. Section 8.1). All definitions are based on ISO 15118 [97]. The inclusion of the certificate installation response data via a *SignedInstallationDataType* element (cf. Listing B.3) is based on ISO 15118-20 [98]. Most changes are implemented as a choice of two sequences, the first being the original message data and the second being the new message data. In Listing B.1, *SACertChain* was added. In Listing B.5, *LocalAuthorizationNonce* was added. In Listing B.6, *EVSEID* and *X2* were added. Listing B.7 shows the new type definitions. The maximum lengths are defined to fit the data and do not necessarily reflect the actual data length.

```
<xs:complexType name="ServiceType">
  <xs:sequence>
    <xs:element name="ServiceID" type="serviceIDType"/>
    <xs:element name="ServiceName" type="serviceNameType" minOccurs="0"/>
    <xs:element name="ServiceCategory" type="serviceCategoryType"/>
    <xs:element name="ServiceScope" type="serviceScopeType" minOccurs="0"/>
    <xs:element name="FreeService" type="xs:boolean"/>
    <xs:element name="SACertChain" type="CertificateChainType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Listing B.1.: Changed ServiceDiscoveryRes (Based on [97])

```
<xs:complexType name="CertificateInstallationReqType">
  <xs:complexContent>
    <xs:extension base="BodyBaseType">
      <xs:choice>
        <xs:sequence>
          <xs:element name="OEMProvisioningCert" type="v2gci_t:certificateType"/>
          <xs:element name="ListOfRootCertificateIDs" type="v2gci_t:
              ListOfRootCertificateIDsType"/>
        </xs:sequence>
        <xs:sequence>
          <xs:element name="EncryptedInstallReq" type="v2gci_t:encryptedInstallReqType"/>
          <xs:element name="DHpublickey" type="v2gci_t:DiffieHellmanPublickeyType"/>
        </xs:sequence>
      </xs:choice>
      <xs:attribute name="Id" type="xs:ID"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Listing B.2.: Changed CertificateInstallationReq (Based on [97])

```
<xs:complexType name="CertificateInstallationResType">
  <xs:complexContent>
    <xs:extension base="BodyBaseType">
      <xs:choice>
        <xs:sequence>
          <xs:element name="ResponseCode" type="v2gci_t:responseCodeType"/>
          <xs:element name="SAProvisioningCertificateChain" type="v2gci_t:
              CertificateChainType"/>
          <xs:element name="ContractSignatureCertChain" type="v2gci_t:CertificateChainType"
              />
          <xs:element name="ContractSignatureEncryptedPrivateKey" type="v2gci_t:
              ContractSignatureEncryptedPrivateKeyType"/>
          <xs:element name="DHpublickey" type="v2gci_t:DiffieHellmanPublickeyType"/>
          <xs:element name="eMAID" type="v2gci_t:EMAIDType"/>
        </xs:sequence>
        <xs:sequence>
          <xs:element name="ResponseCode" type="v2gci_t:responseCodeType"/>
          <xs:element name="SAProvisioningCertificateChain" type="v2gci_t:
              CertificateChainType"/>
          <xs:element name="signedInstallationData" type="v2gci_t:
              SignedInstallationDataType"/>
        </xs:sequence>
      </xs:choice>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Listing B.3.: Changed CertificateInstallationRes (Based on [97])

```
<xs:complexType name="PaymentDetailsReqType">
  <xs:complexContent>
    <xs:extension base="BodyBaseType">
      <xs:choice>
        <xs:sequence>
          <xs:element name="eMAID" type="v2gci_t:eMAIDType"/>
          <xs:element name="ContractSignatureCertChain" type="v2gci_t:CertificateChainType"
              />
        </xs:sequence>
        <xs:sequence>
          <xs:element name="emspDAACertChain" type="v2gci_t:CertificateChainType"/>
          <xs:element name="RndDAACredential" type="v2gci_t:rndDAACredentialType"/>
          <xs:element name="ContractSessionPublicKey" type="v2gci_t:
              contractSessionPublicKeyType"/>
          <xs:element name="ContractSessionCertifyInfo" type="v2gci_t:
              contractSessionCertifyInfoType"/>
          <xs:element name="ContractSessionCertifySignature" type="v2gci_t:daaSignatureType
              "/>
          <xs:element name="X1" type="v2gci_t:sha256DigestType"/>
        </xs:sequence>
      </xs:choice>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Listing B.4.: Changed PaymentDetailsReq (Based on [97])

```
<xs:complexType name="PaymentDetailsResType">
  <xs:complexContent>
    <xs:extension base="BodyBaseType">
      <xs:sequence>
        <xs:element name="ResponseCode" type="v2gci_t:responseCodeType"/>
        <xs:element name="GenChallenge" type="v2gci_t:genChallengeType"/>
        <xs:element name="LocalAuthorizationNonce" type="v2gci_t:sha256DigestType"/>
        <xs:element name="EVSETimeStamp" type="xs:long"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Listing B.5.: Changed PaymentDetailsRes (Based on [97])

```
<xs:complexType name="AuthorizationReqType">
  <xs:complexContent>
    <xs:extension base="BodyBaseType">
      <xs:sequence>
        <xs:element name="EVSEID" type="v2gci_t:evseIDType" minOccurs="0"/>
        <xs:element name="GenChallenge" type="v2gci_t:genChallengeType" minOccurs="0"/>
        <xs:element name="X2" type="v2gci_t:sha256DigestType" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="Id" type="xs:ID"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Listing B.6.: Changed AuthorizationReq (Based on [97])

```
<xs:complexType name="SignedInstallationDataType">
  <xs:sequence>
    <xs:element name="emspDAACertChain" type="CertificateChainType"/>
    <xs:element name="eMAIDKeyEncrypted" type="encryptedTPMDataType"/>
    <xs:element name="eMAIDEncryptionDHpublickey" type="DiffieHellmanPublickeyType"/>
    <xs:element name="eMAIDKeyPublic" type="publicAreaTPMDataType"/>
    <xs:element name="ActivatCredentialKeyEncrypted" type="encryptedTPMDataType"/>
    <xs:element name="ActivatCredentialEncryptionDHpublickey" type="
       DiffieHellmanPublickeyType"/>
    <xs:element name="DAAContractCredentialEncrypted" type="encryptedTPMDataType"/>
  </xs:sequence>
  <xs:attribute name="Id" type="xs:ID" use="required"/>
</xs:complexType>

<xs:simpleType name="encryptedInstallReqType">
  <xs:restriction base="xs:base64Binary">
    <xs:maxLength value="1000"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="encryptedTPMDataType">
  <xs:restriction base="xs:base64Binary">
    <xs:maxLength value="1000"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="publicAreaTPMDataType">
```

```
27    <xs:restriction base="xs:base64Binary">
28      <xs:maxLength value="1000"/>
29    </xs:restriction>
30  </xs:simpleType>
31
32  <xs:simpleType name="rndDAACredentialType">
33    <xs:restriction base="xs:base64Binary">
34      <xs:maxLength value="280"/>
35    </xs:restriction>
36  </xs:simpleType>
37
38  <xs:simpleType name="contractSessionPublicKeyType">
39    <xs:restriction base="xs:base64Binary">
40      <xs:maxLength value="64"/>
41    </xs:restriction>
42  </xs:simpleType>
43
44  <xs:simpleType name="contractSessionCertifyInfoType">
45    <xs:restriction base="xs:base64Binary">
46      <xs:maxLength value="1000"/>
47    </xs:restriction>
48  </xs:simpleType>
49
50  <xs:simpleType name="daaSignatureType">
51    <xs:restriction base="xs:base64Binary">
52      <xs:maxLength value="1000"/>
53    </xs:restriction>
54  </xs:simpleType>
55
56  <xs:simpleType name="sha256DigestType">
57    <xs:restriction base="xs:base64Binary">
58      <xs:maxLength value="64"/>
59    </xs:restriction>
60  </xs:simpleType>
```

Listing B.7.: New Type Definitions

C. Measurements

The following figures show the timing measurements for the computational overhead of the proof-of-concept implementation as discussed in Section 8.2.2. That is, the figures show the measured times in ms for each of the 100 repetitions ("# of Measurement") per process. The legend entry "Normal Method" refers to the baseline measurements with the default RISE V2G implementation and the legend entry "Proposed Method" refers to the measurements of the proposed privacy-preserving method with the proof-of-concept implementation. The whitelist management processes (cf. Fig C.1) were only measured for the proposed method as mentioned in Section 8.2.2. Note that the scale of the y axis is different per figure.

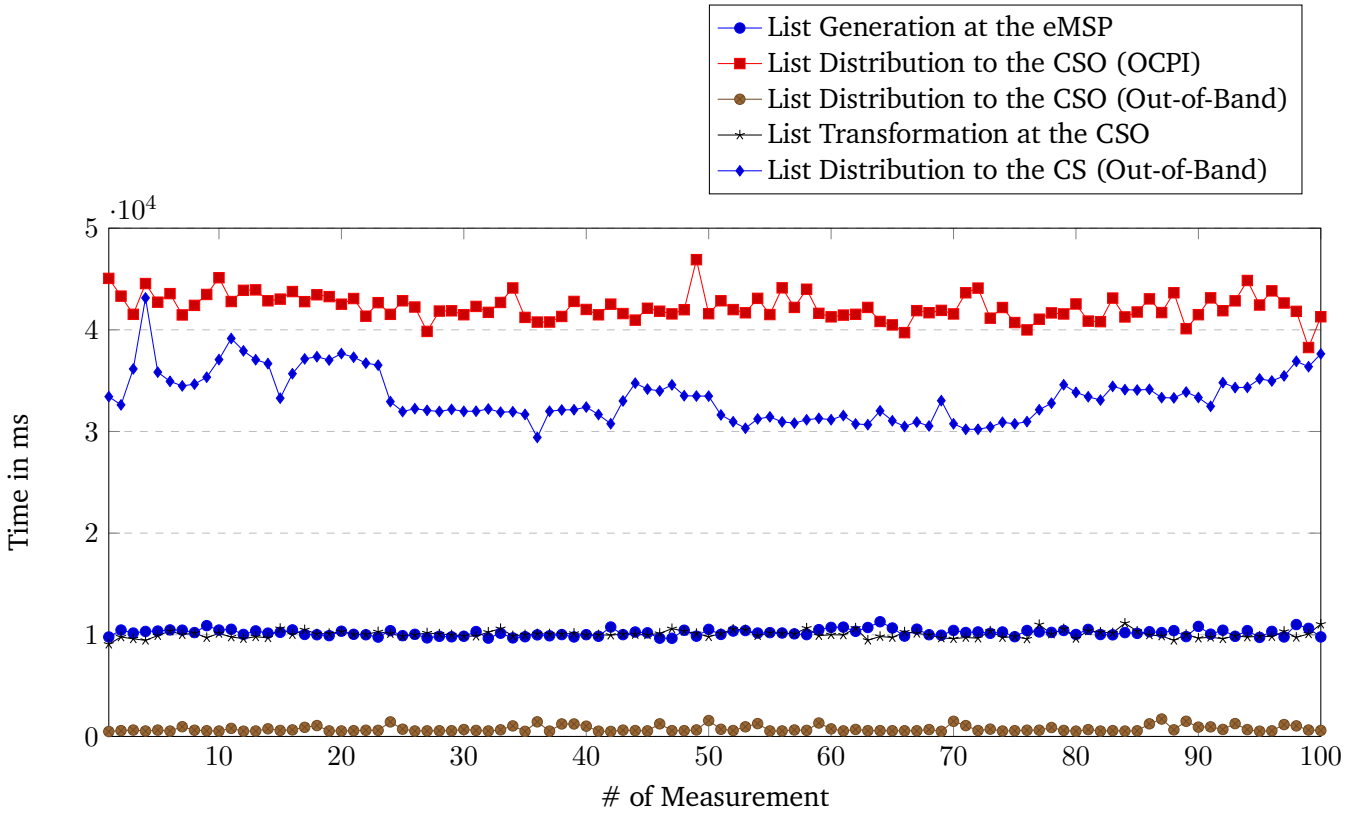


Figure C.1.: Measured Times for Privacy-Preserving Whitelist Management

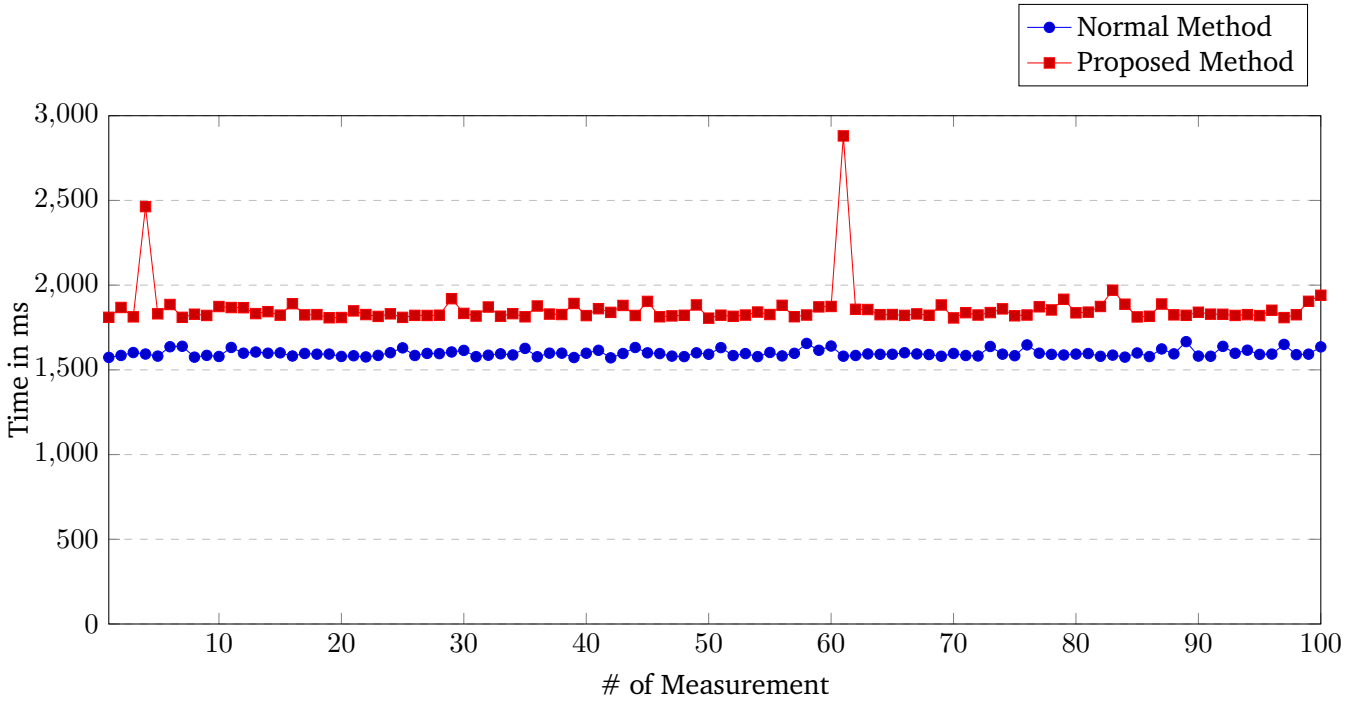


Figure C.2.: Measured Times of *CertificateInstallationReq* Generation

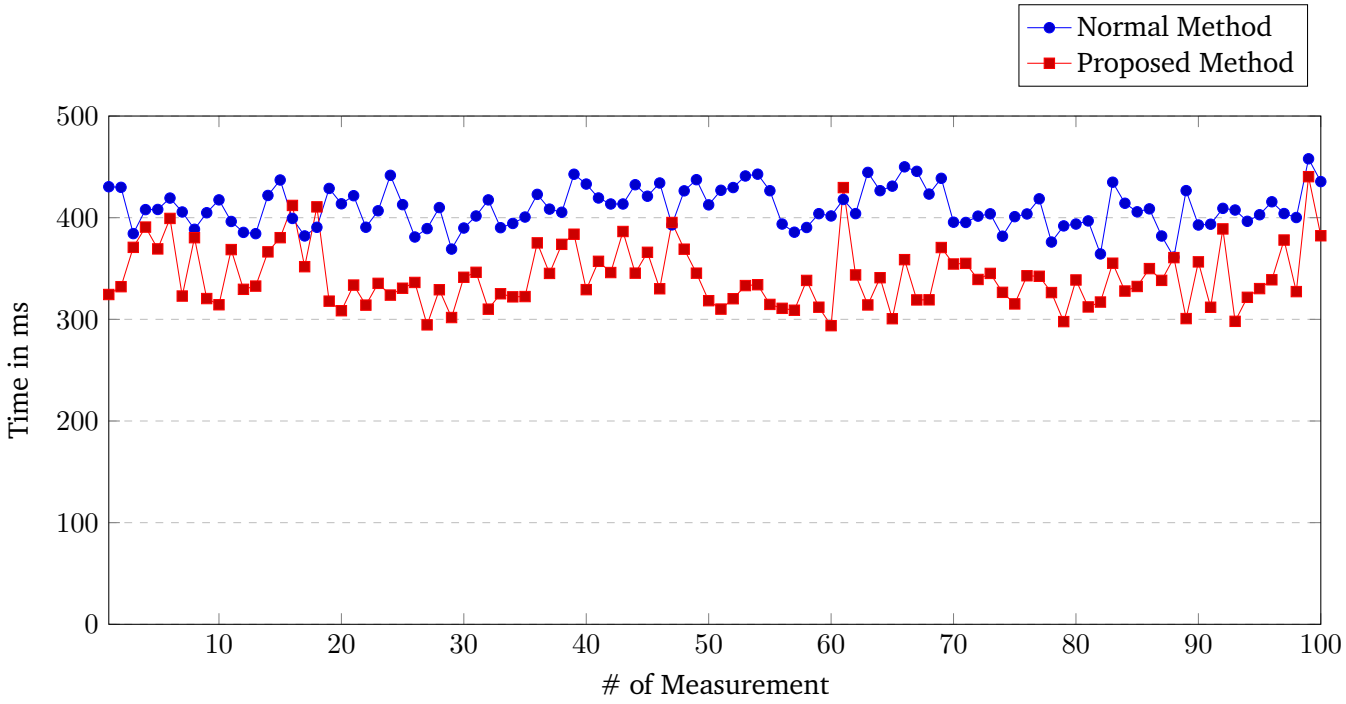


Figure C.3.: Measured Times of *CertificateInstallationReq* Handling

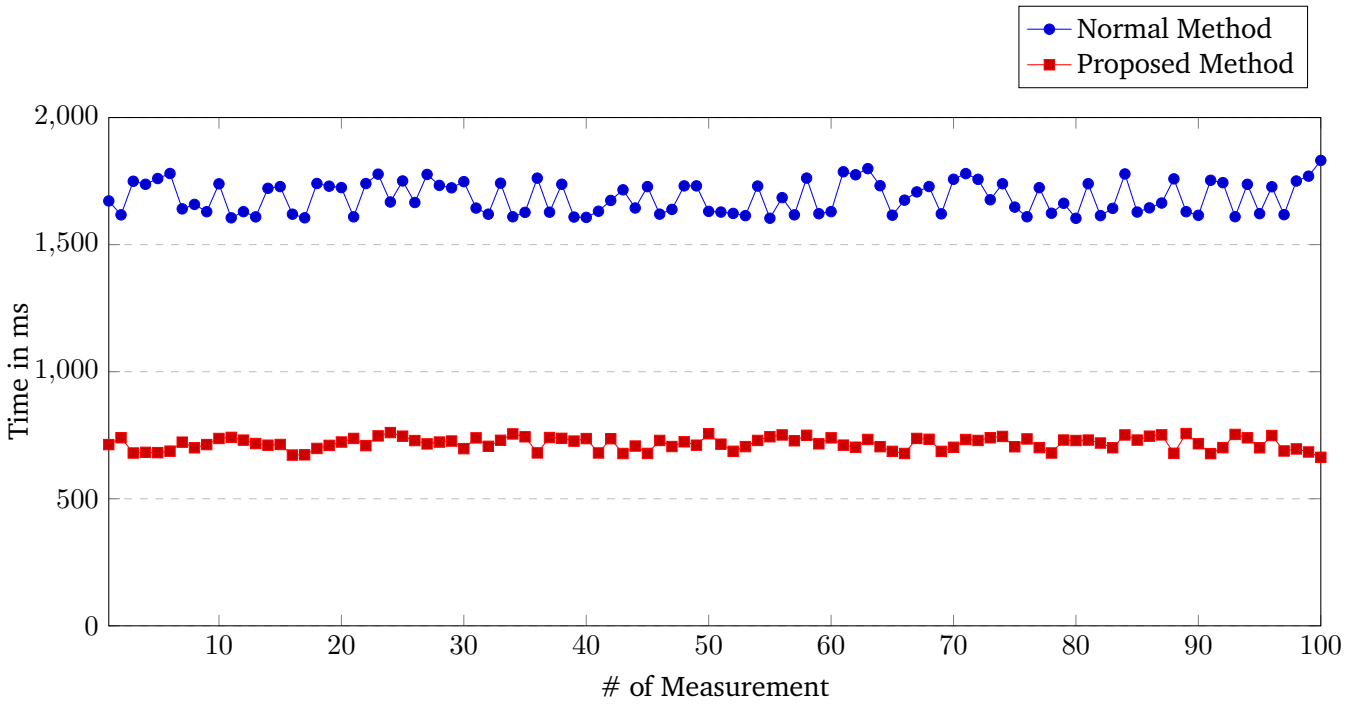


Figure C.4.: Measured Times of *CertificateInstallationRes* Handling

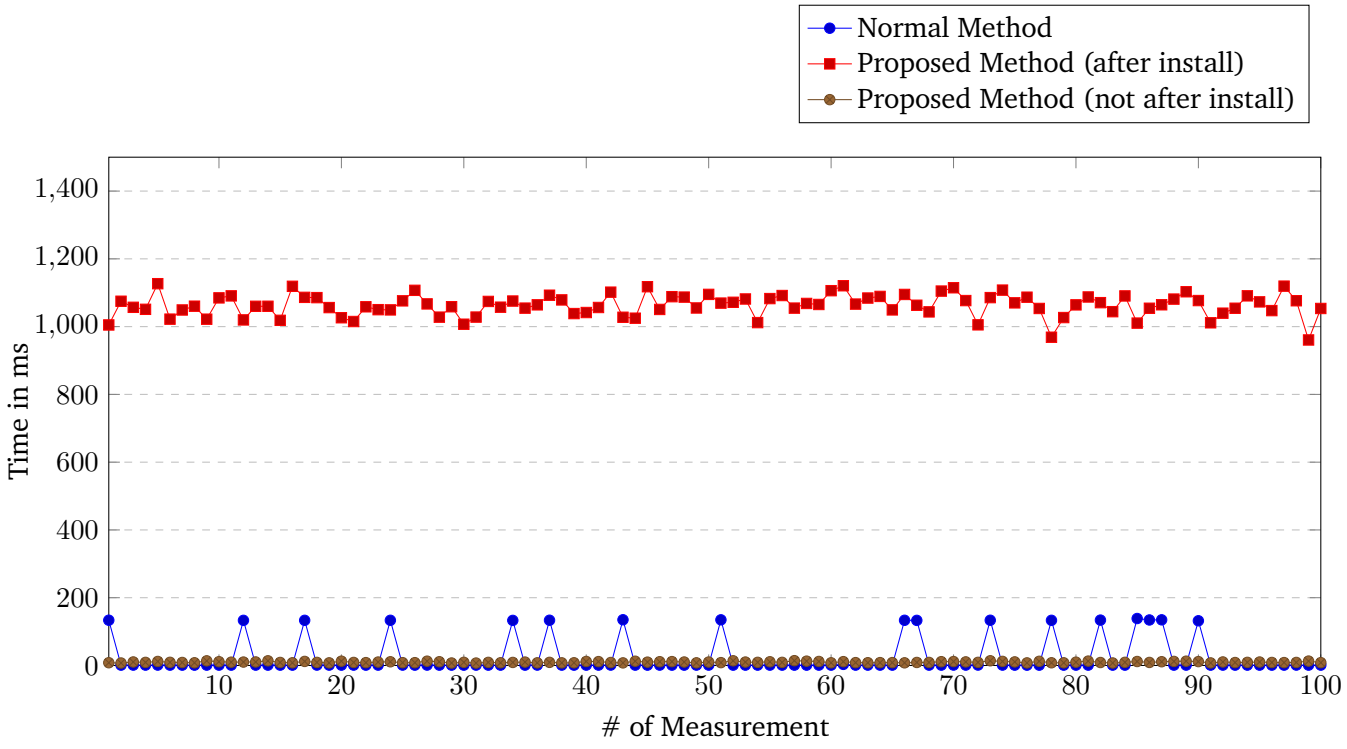


Figure C.5.: Measured Times of *PaymentDetailsReq* Generation

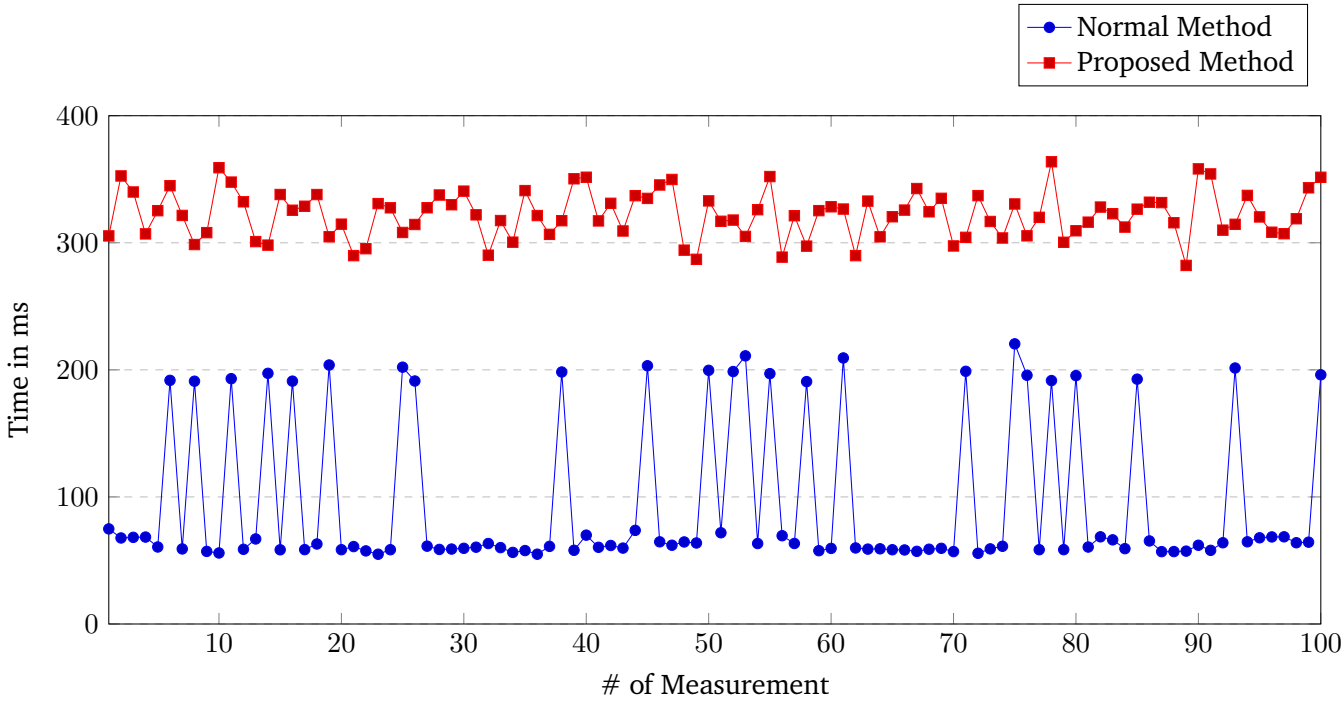


Figure C.6.: Measured Times of *PaymentDetailsReq* Handling

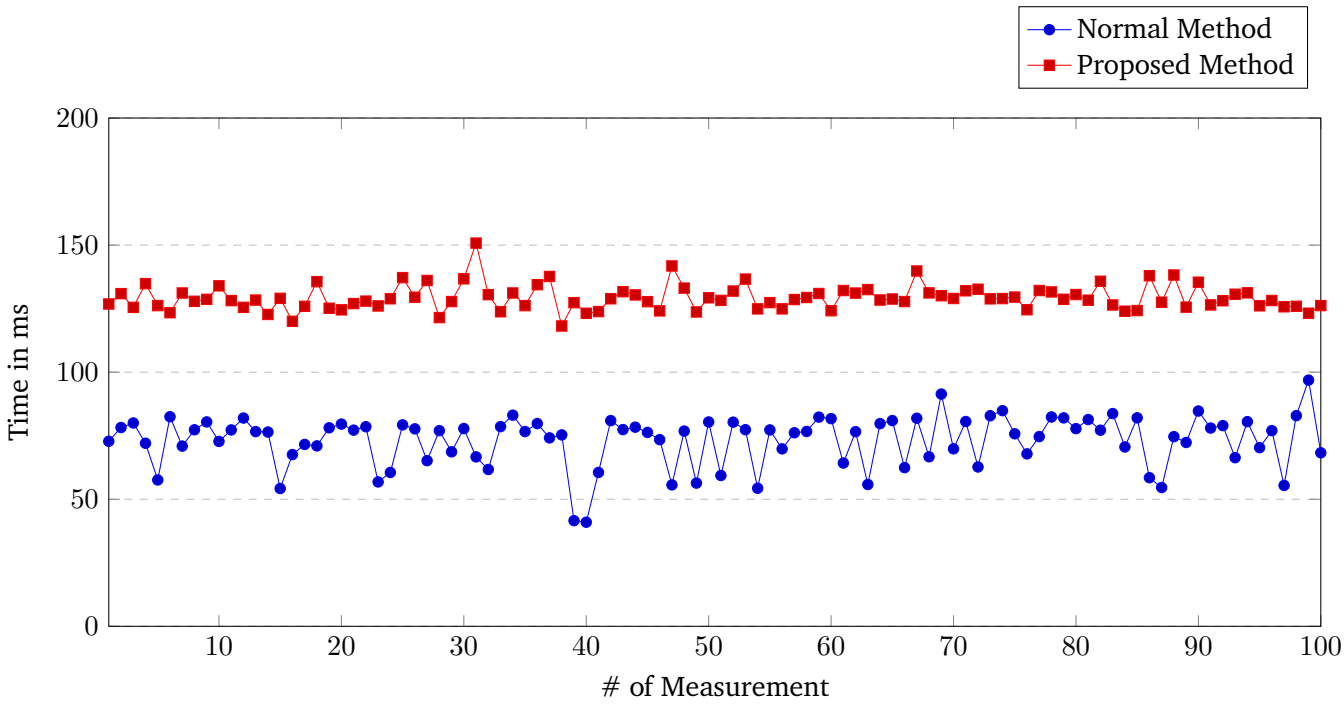


Figure C.7.: Measured Times of *AuthorizationReq* Generation

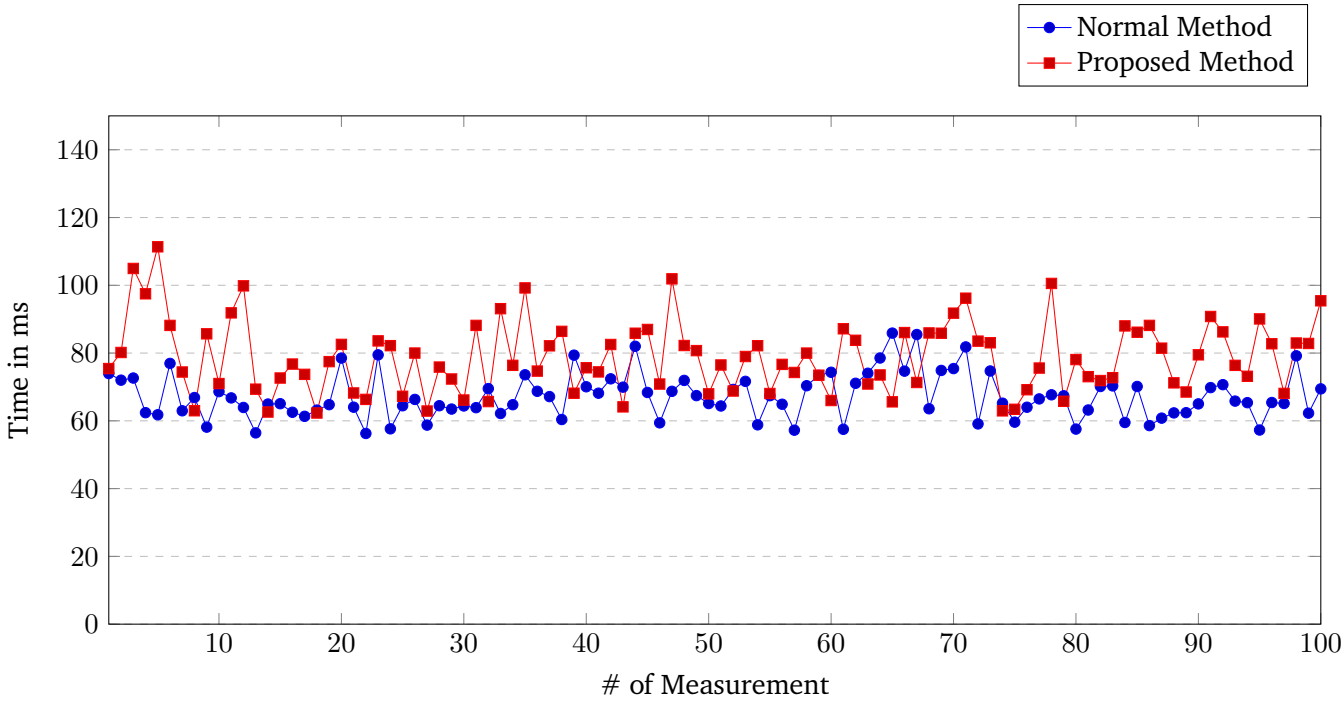


Figure C.8.: Measured Times of Offline Authorization

D. Agreement During the Join

In order to validate the security properties of the DAA join-based credential installation method, the symbolic model from [205][1] was adapted to reflect the proposed changes. For the symbolic analysis in [205], the Tamarin prover [131][2] was used. With Tamarin, protocols are specified using multiset rewriting rules and properties are specified using guarded fragment of first-order logic. By default, the adversary is a Dolev-Yao adversary [47] with control over the network. Tamarin enables the automated construction of proofs or disproofs (by counterexample) for a protocol's desired security properties (specified as trace properties in the form of lemmas).

Due to time constraints, the symbolic analysis in this thesis is limited to the modifications to the join procedure from [205]. The main changes compared to the model from [205] are:

- The changed model allows the setup of multiple issuers (eMSPs).[3]
- The changed model includes rules to model the role of the CPS.
- The changed model includes rules for a secure channel between eMSP and CPS in order to represent their TLS connection.
- The changed model does not require the host (EV) to know their issuer (eMSP) before the join.
- The changed model includes rules for the creation of the EV's provisioning credential PC key pair in the TPM. The TPM-level design of the rule follows the design of the existing key creation rules from [205].
- The `Host_Store_DAA_Key` rule is changed in order to incorporate the creation and encryption of $CertReq$ for the CPS. EC_{pk} and PC_{pk} (instead of only EC_{pk}) are provided to the CPS securely via `EK_FOR_ISSUER` in order to model the inclusion of these keys in the EV's provisioning certificate PC_{cert}.
- The `CPS_Verify_Credential_Request` rule is added to model the decryption and verification of the EV's $CertReq$ by the CPS.
- The `CPS_Sign_Credential_Response` rule is added to model the CPS' signature over the certificate installation response data from the eMSP. The result is directly sent to the EV via an insecure channel, i.e., access by a manipulated CS is model via Tamarin's Dolev-Yao adversary.

[1] The model from [205], which was used as the basis for the modifications, can be found at `https://github.com/tamarin-prover/tamarin-prover/blob/522dda8ef7d91b0fe552a7c8e27ee864cbf807a1/examples/asiaccs20-eccDAA/CERTIFY/TPM_DAA_JoinCertify_with_fix_noBSN.spthy` (visited on 2020-12-14).

[2] `https://tamarin-prover.github.io/` (visited on 2020-1-8).

[3] Note that the action fact `OnlyOnce` is only used by [205] for the correctness lemma, i.e., it does not restrict the eMSP initialization in any other case.

Listing D.1 shows the modified join part of the symbolic model from [205]. As the model is based on [205], the modeling choices from [205] are adopted, i.e., TPM and Host are modeled as separate entities with a one to one relationship, TPM and Host communicate over a secure channel, and issuer and verifier are modeled as independent entities (cf. [205] for details). The equational theories for the DAA computations from [205] are omitted in Listing D.1 for brevity. With the modified symbolic model, the injective agreement during the join property (verified with the lemma shown in Listing D.2) still holds.[4] That is, based on the authentication definitions of Lowe [126] and their transfer to the DAA join use case by [205] (as mentioned in Section 2.5.4), whenever a signer (identified by its EC_{pk}) completes a join with an issuer, then the issuer has been running the join in the issuer role with the signer, and both the issuer and the signer agree on the exchanged credentials, and each join of the signer corresponds to a unique run of the issuer. Based on [205], only the compromise (`KeyReveal`) of one of the entities running the protocol (marked as `Honest` at `i`, i.e., at the time of the `Commit` in the `Host_JoinComplete` rule) is excluded.

Note that, as discussed in Section 9.2, injective agreement to the issuer with the signer on the credential (join) request cannot be provided by the proposed solution. In fact, only non-injective agreement (with no guarantee of uniqueness) [126] with regard to credential requests as shown in Listing D.3 holds. That is, whenever a CPS accepts a credential join from an EV, then the EV has been running the join with the CPS, and both the CPS and EV agree on the request data. Note that the non-injective agreement requires that no CPS was compromised (which is assumed by the adversary model; cf Section 5.1). If this condition does not hold, then the proposed solution does not provide the described non-injective agreement property as the leaked private key of a CPS could be used to decrypt the credential request of an EV and re-encrypt it for a different CPS. This possibility could be avoided by including the CPS' public key (from its certificate) in the EV's request signature such that non-injective agreement with regard to credential requests can be provided under a stronger adversary, i.e., only requiring that no `KeyReveal` occurred for one of the entities running the protocol.

```
theory TPM_DAA_JoinCertify_with_fix_noBSN_thesis
begin
builtins:   asymmetric-encryption, symmetric-encryption, signing

functions:  accept/0, MAC/2, KDF_AES/1, KDF_EK/1,KDF_a/3, QPub/2, QName/2, certData/2, DAAKeyID/1,
             multp/2, plus/2, minus/2, len16/1,
             H_SHA256/1,  H_k_1/1, H_k_2/2, H_k_4/4, H_k_6/6, H_n_2/2, H_n_2/2, H_n_8/8, H_6/1,
       curlyK/1, E/2, E_S/2, L_J/2, RB/2, RD/2,
       calcE/1,
       calcE_S_cert/4, calcL_J_cert/4,
       calcRB/1, calcRD/1, Nonce/1,
       PkX/2, PkY/2, verifyCre1/4, verifyCre2/5,verifyCre3/4,verifyCre4/5,
       BSN/1, F1/1, F2/1, H_p/1,PointG1/2, Message/1, Q_K/1, certify/1, publicData/1

// Protocol Restrictions (Axioms)
restriction equality:        "All #i    x y    .  Eq( x, y ) @ i ==> x = y"

//Modification: removed restriction for single issuer initialisation
//a host should only initialise itself once
restriction single_host_init:
  "All Host #i #j . ((Host_Init(Host)@i & Host_Init(Host)@j) ==> (#i=#j)) "

//a TPM should only be initialised once (and hence there is only one aes key and one TPM_EK_SEED):
restriction single_tpm_init:
  "All PS #i #j. ((TPM_Init(PS)@i & TPM_Init(PS)@j) ==> (#i=#j))"

//Modification: adjusted restrictions for ID uniqueness to new entity definitions
restriction every_id_must_be_one_of_tpm_host:
```

[4] Verified on one of the VMs as mentioned in Section 8.1.1. The other lemmas of the DAA join (with certify without basename) model from [205] (for correctness, credential secrecy, and unforgabiltiy) also still hold but are not shown here as the main goal is to show the security of the modified join procedure.

```
  "All Ent1 Ent2 #i #j .
  (Host_Init(Ent1) @ i & TPM_Init(Ent2) @ j)
  ==>
  (not(Ent1=Ent2))"

restriction every_id_must_be_one_of_issuer:
  "All Ent1 Ent2 #i #j .
  (Issuer_Init(Ent1) @ i & CPS_Init(Ent2) @ j)
  ==>
  (not(Ent1=Ent2))"

//a host_init, tpm_init and issuer_init must have different identities
restriction every_id_must_be_one_of_tpm_host_issuer:
  "All Ent1 Ent2 Ent3 Ent4 #i #j #k #l.
  (Host_Init(Ent1) @ i & TPM_Init(Ent2) @ j & Issuer_Init(Ent3) @ k & CPS_Init(Ent4) @ l)
  ==>
  (not(Ent1=Ent2) & not (Ent1=Ent3) & not (Ent1=Ent4)
  & not (Ent2=Ent3) & not (Ent2=Ent4)
  & not (Ent3=Ent4))"
/*
We need a secure channel between the TPM aka the Principal Signer (PS)
and its host aka the Assistant Signer (AS). We refer to the combination
of a PS and AS as a Platform.
*/

/* Secure Channel rules

   Communication between the Host or Assistant Signer (AS) and the TPM
   or Principal Signer (PS) is done over a 'Secure Channel'. This means
   that an adversary can neither modify nor learn messages that are
   sent over the channel. Sec( A, B, x ) is a linear fact modelling
   that the adversary cannot replay on this channel. Secure channels
   have the property of being both confidential and authentic.
   Communication between the AS and PS is constrained by the channel
   invariant !F_Paired, such that two arbitrary roles cannot communicate
   over this channel.
*/
rule ChanOut_S [colour=ffffff]:
    [ Out_S( $A, $B, x ), !F_Paired( $A, $B ) ]
  --[ ChanOut_S( $A, $B, x ) ]->
    [ Sec( $A, $B, x ) ]

rule ChanIn_S [colour=ffffff]:
    [ Sec( $A, $B, x ) ]
  --[ ChanIn_S( $A, $B, x ) ]->
    [ In_S( $A, $B, x ) ]


/* Modification: Added Secure Channel rules for backend communication
  Secure TLS Channel between backend actors.
  Channel is confidential and authentic.
*/
rule ChanOut_S_Backend:
        [ Out_S_B($A,$B,x) ]
      --[ ChanOut_S_B($A,$B,x) ]->
        [ !SecB($A,$B,x) ]

rule ChanIn_S_Backend:
        [ !SecB($A,$B,x) ]
      --[ ChanIn_S_B($A,$B,x) ]->
        [ In_S_B($A,$B,x) ]

/*
Issuer set-up:
Modification: Allow multiple issuer (eMSP) set-ups
*/
rule Issuer_Init:
```

```
    let
      pkX=PkX(~x,'P2')
      pkY=PkY(~y,'P2')
    in
    [Fr(~x),
     Fr(~y)]
    --[Issuer_Init($I)
       , OnlyOnce('Issuer_Init')]->
    [
    !Ltk($I,~x, ~y), !Pk($I, pkX,pkY),
    Out(<pkX,pkY>),
    !Issuer_Initialised($I)
    ]


// simple key reveal rule for the issuer's secret key pair
rule Issuer_KeyReveal:
  [!Ltk(I, ~x, ~y)]
  --[KeyReveal('Issuer_KeyReveal', I)]->
  [Out(<~x,~y>)]

/*
Modification: Added CPS set-up
*/
rule CPS_Init:
    [Fr(~cps)]
    --[CPS_Init($CPS_I)
       , OnlyOnce('CPS_Init')]->
    [
    !LtkCPS($CPS_I,~cps), !PkCPS($CPS_I, pk(~cps)),
    Out(pk(~cps)),
    !CPS_Initialised($CPS_I)
    ]

// simple key reveal rule for the CPS' secret key pair
rule CPS_KeyReveal:
  [!LtkCPS($CPS_I, ~cps)]
  --[KeyReveal('CPS_KeyReveal', $CPS_I)]->
  [Out(~cps)]

/*
Platform set-up:
For a platform we need a TPM (the principal signer) and a Host (the assistant signer)
before binding them together in a platform.

Modification: Removed need for the host to know the issuer before the join.
*/
rule TPM_INIT:
  let
  //!Assumption that the aes key is derived by a KDF_AES key derivation function
  aes_key=KDF_AES(~TPM_AES_Seed)
  in
    [Fr(~TPM_AES_Seed),
     Fr(~TPM_EK_Seed)]
    --[TPM_Init($PS)
       , OnlyOnce('TPM_INIT')]->
    [!TPM_AES_Key($PS, aes_key),
     TPM_EK_SEED($PS,~TPM_EK_Seed),
     TPM_Initialised($PS)]

//simple rule to allow the TPM's aes key to leak
rule TPM_AESReveal:
  [!TPM_AES_Key(PS, aes_key)]
  --[KeyReveal('TPM_AESReveal', PS)]->
  [Out(aes_key)]

rule Host_Init:
```

```
  []
  --[Host_Init($AS)
    , OnlyOnce('Host_Init')]->
  [Host_Initialised($AS)]

//This rule binds an $PS and an $AS to one another.

rule Platform_Setup:
  [ TPM_Initialised($PS)
  , Host_Initialised($AS)
  ]
   //Action label used to ensure there is a one-to-one correspondence between AS and PS
   --[ Bind($PS,$AS)
    ,OnlyOnce('Platform_Setup')
     ]->
   [ Out_S($AS, $PS, < 'createPrimary'>)
    , !F_Paired($AS,$PS)
    , !F_Paired($PS,$AS)
    ]

//The TPM executes this in response to a request by the host
//Note this should only be executed by a TPM once!
rule TPM2_CreatePrimary:
  let
  e=KDF_EK(~TPM_EK_Seed)
  E_PD=<'EK_public_data',pk(e)>
    in
  [ In_S($AS, $PS, < 'createPrimary'>)
  , TPM_EK_SEED($PS,~TPM_EK_Seed)]
  --[ TPM2_EK_Created($PS, $AS, pk(e))
    , OnlyOnce('TPM2_CreatePrimary')
    ]->
  [Out_S($PS,$AS, < E_PD, 'returnEK'>),
  !TPM_ENDORSEMENT_SK($PS, e, pk(e)),
  !TPM_ENDORSEMENT_PK($PS,E_PD),
  Out(pk(e))]

//simple rule to reveal the TPM's endorsement key
rule TPM_EKReveal:
  let
  e=KDF_EK(~TPM_EK_Seed)
    in
  [!TPM_ENDORSEMENT_SK(PS, e, pk(e))]
  --[ KeyReveal('TPM_EKReveal_tpm', PS)
    , KeyReveal('TPM_EKReveal_pke', pk(e))
    ]->
  [Out(e)]

//The Host should store the public endorsement key
rule Host_Store_EK:
  let
  E_PD=<'EK_public_data', pk(KDF_EK(~TPM_EK_Seed))>
  in
  [ In_S($PS,$AS, < E_PD, 'returnEK'>) ]
  --[ Store_EK($PS, $AS)
    , OnlyOnce('Host_Store_EK')
    ]->
  [Out_S($AS,$PS, < pk(KDF_EK(~TPM_EK_Seed)), 'createPCKey'>)]

/*
Modification: Added the generation of the provisioning key pair.
  The rule is designed based on the existing TPM2_CreatePrimary and TPM2_Create rules
*/
rule TPM2_CreatePC:
  let
  PC_PD=<'PC_public_data',pk(~pc)>
  PC_SD=senc(~pc,aes_key)
```

```
  pke=pk(KDF_EK(~TPM_EK_Seed))
    in
  [ In_S($AS, $PS, < pke, 'createPCKey'>)
   , !TPM_AES_Key($PS, aes_key)
   , Fr(~pc) //pc secret key
   ]
  --[ TPM2_PC_Created($PS, $AS)
    , DerivePCKey($PS, $AS, pke, ~pc)
    , OnlyOnce('TPM2_CreatePC')
    ]->
  [Out_S($AS,$PS, < pke, PC_SD,PC_PD, 'createDAAKey'>),
   Out_S($PS, $AS,< PC_SD,PC_PD, 'returnPCKey'>),
   !TPM_PC_SK($PS, pk(KDF_EK(~TPM_EK_Seed)), ~pc),
   Host_State_02($PS, $AS, pk(KDF_EK(~TPM_EK_Seed)), pk(~pc)),
   Out(pk(~pc))
  ]

//simple rule to reveal the TPM's pc key
rule TPM_PCReveal:
  [!TPM_PC_SK(PS, pk(KDF_EK(~TPM_EK_Seed)), ~pc)]
  --[ KeyReveal('TPM_PCReveal_tpm', PS)
    , KeyReveal('TPM_PCReveal_pke', pk(KDF_EK(~TPM_EK_Seed)))
    ]->
  [Out(~pc)]

/*
This rule will create a DAA key
Note that unlike the TPM2_CreatePrimary rule, this rule can be executed
multiple times resulting in a new DAA key
This is obviously not sensible but allowed.
*/
rule TPM2_Create:
  let
  Q=multp(~f, 'P1')
  Q_PD=<'DAA_public_data', Q>
  Q_SD=senc(~f,aes_key)
  pke=pk(KDF_EK(~TPM_EK_Seed))
  PC_PD=<'PC_public_data',pk(~pc)>
  PC_SD=senc(~pc,aes_key)
  in
  [In_S($AS, $PS, < pke, PC_SD,PC_PD, 'createDAAKey'>)
   , !TPM_AES_Key($PS, aes_key)
   , Fr(~f) //our secret key
   ]
  --[ TPM2_DAA_Created($PS, $AS)
    , DeriveDAAKey($PS, $AS, pke, ~f)
    , OnlyOnce('TPM2_Create')
    ]->
  [ Out_S($PS, $AS,< Q_SD,Q_PD, 'returnDAAKey'>),
   !TPM_DAA_SK($PS, pk(KDF_EK(~TPM_EK_Seed)), ~f),
   Out(Q)
  ]

//simpe rule to leak the DAA key:
rule TPM_DAAReveal:
  [!TPM_DAA_SK(PS, pk(KDF_EK(~TPM_EK_Seed)), ~f)]
  --[ KeyReveal('TPM_DAAReveal_tpm', PS)
    , KeyReveal('TPM_DAAReveal_pke', pk(KDF_EK(~TPM_EK_Seed)))
    ]->
  [Out(~f)]

/*
The host needs to store the keys on behalf of the TPM as it has
limited memory. The host then sends the endorsement key and the public DAA key to issuer
for verification

Modification: Included the generation and encryption of the EV's CertReq
```

```
*/

rule Host_Store_DAA_Key:
  let
  Q=multp(~f, 'P1')
  Q_PD=<'DAA_public_data', Q>
  aes_key=KDF_AES(~TPM_AES_Seed)
  Q_SD=senc(~f,aes_key)
  PC_PD=<'PC_public_data',pk(~pc)>
  PC_SD=senc(~pc,aes_key)
  m=<pk(KDF_EK(~TPM_EK_Seed)),pk(~pc), Q_PD, 'join_Issuer_1'> //EV's CertReq with OEM prov cert and public DAA key
  signed_m=<m> // The signature over CertReq; signed_m=<m, pk(~cps)> for inclusion of CPS' public key
  in
  [In_S($PS, $AS, < Q_SD,Q_PD, 'returnDAAKey'>)
  , In_S($PS, $AS, < PC_SD,PC_PD, 'returnPCKey'>)
  , Host_State_02($PS, $AS,  pk(KDF_EK(~TPM_EK_Seed)),  pk(~pc))
  , !PkCPS(CPS_I, pk(~cps)) // from Charge Station but authentic due to certificate
  ]
  --[ Store_DAA($PS, $AS)
    , PlatformSendKeys($PS, $AS, pk(KDF_EK(~TPM_EK_Seed)), Q_PD, pk(~pc))
    , Alive($AS)
    , Role('Platform')
    , Honest( $PS )
    , Honest($AS)
    , Honest( pk(KDF_EK(~TPM_EK_Seed)) )
    , Honest( CPS_I )
    , RunningEV( pk(KDF_EK(~TPM_EK_Seed)) , CPS_I, signed_m )
    , OnlyOnce('Host_Store_DAA_Key')
    ]->
  [ !EK_FOR_ISSUER(<pk(KDF_EK(~TPM_EK_Seed)),pk(~pc)>) //a TPM created the pk(e) and pk(pc), both certified by OEM
  , Out(<CPS_I, aenc(<sign(signed_m,~pc),m>,pk(~cps))>) //CPS_I is included as CS and CSO know chosen CPS
  , Host_State_05($PS, $AS,  pk(KDF_EK(~TPM_EK_Seed)), Q_PD, Q_SD, pk(~cps), CPS_I)
  ]

//Modification: Added verification of the certificate request by the CPS
rule CPS_Verify_Credential_Request:
  let
    //inputs
    Q=multp(f, 'P1')
    Q_PD=<'DAA_public_data', Q>
    m=<pk(KDF_EK(TPM_EK_Seed)),pk(pc), Q_PD, 'join_Issuer_1'>
    //decrypt and parse request
    aenc_m=aenc(<sign(signed_m_in,pc),m>, pk(~cps))
    sig_and_m=adec(aenc_m,~cps)
    sig=fst(sig_and_m)
    m_in=snd(sig_and_m)
    signed_m=<m_in> //signed_m=<m_in, pk(~cps)>
  in
     [ In(<CPS_I, aenc_m>)
    , !EK_FOR_ISSUER(<pk(KDF_EK(TPM_EK_Seed)),pk(pc)>)  //a TPM created the pk(e) and pk(pc), both certified by OEM
    , !PkCPS(CPS_I,pk(~cps))
    , !LtkCPS(CPS_I,~cps)
    , !Issuer_Initialised(I)
   ]
   --[ Eq(verify(sig,signed_m,pk(pc)), true)
    , Check_Ek(pk(KDF_EK(TPM_EK_Seed)))
    , Check_QPD(Q_PD)
    , Honest ( CPS_I )
    , Honest ( pk(KDF_EK(TPM_EK_Seed)) )
    , CPSReceivedReq(CPS_I, pk(KDF_EK(TPM_EK_Seed)), pk(pc), Q_PD)
    , CommitCPS(CPS_I, pk(KDF_EK(TPM_EK_Seed)), signed_m)
    , OnlyOnce('CPS_Verify_Credential_Request')
    ]->
```

```
   //secure channel to EMSP
   [Out_S_B(CPS_I, I, <m_in, 'CPS_Fwd_Req'>) ]

//Issuer: verify the curlyK and the signature before issuing the proper credentials
//Modification: Changed to model new behavior of eMSP issuer, i.e., interaction with CPS and inclusion
    of EMSP_Cert.
rule Issuer_Issue_Credentials:
  let
    //inputs
    Q=multp(f, 'P1')
    Q_PD=<'DAA_public_data', Q>
    m=<pk(KDF_EK(TPM_EK_Seed)),pk(pc), Q_PD, 'join_Issuer_1'>

    //inputs from Issuer PK
    pkX=PkX(~x,'P2')
    pkY=PkY(~y,'P2')

    //new values to be calculated
    A=multp(~r,'P1')
    B=multp(~y,A)
    C=plus(multp(~x,A),multp(multp(multp(~r,~x),~y),Q))
    D=multp(multp(~r,~y),Q)

    R_B=RB(~l,'P1')
    R_D=RD(~l,Q)

    u=H_n_8('P1', Q, R_B, R_D, A, B, C, D)
    j=plus(~l,multp(multp(~y,~r),u))

    Q_N=<'SHA256',H_SHA256(Q_PD)>     //the name of the DAA key
    k_e=KDF_a(~s_2,'STORAGE',Q_N)
    k_h=KDF_a(~s_2,'INTEGRITY','NULL')
    curlyK_2=curlyK(~K_2)
    curlyK_2_hat=senc(curlyK_2,k_e)
    curlyH=MAC(<len16(curlyK_2_hat),curlyK_2_hat, Q_N>,k_h)
    s_2_hat=aenc(~s_2,pk(KDF_EK(TPM_EK_Seed)))
    C_hat=senc(<A,B,C,D,u,j>,curlyK_2)

    EMSP_Cert=<I,pkX,pkY>
  in
     [ In_S_B(CPS_I, I, <m, 'CPS_Fwd_Req'>)
    , !Pk(I,pkX,pkY)
    , !Ltk(I,~x,~y)
    , Fr(~r)
    , Fr(~l)
    , Fr(~s_2)
    , Fr(~K_2)
   ]
   --[ Running(I, pk(KDF_EK(TPM_EK_Seed)), <A, B, C, D>)
    , Alive(I)              //the issuer is "alive" in the protocol here
    , Honest ( I )
    , Honest ( pk(KDF_EK(TPM_EK_Seed)) )
    , IssuerReceivedKeys(I, pk(KDF_EK(TPM_EK_Seed)), pk(pc), Q_PD)
    , OnlyOnce('Issuer_Verify_Challenge')
    ]->
   [Out_S_B(I, CPS_I, <EMSP_Cert, curlyH, len16(curlyK_2_hat), curlyK_2_hat, s_2_hat, C_hat, '
    Host_CompleteJoin'>) ]

//Modification: Added signature over the certificate response by the CPS
rule CPS_Sign_Credential_Response:
  let
    //inputs
    m=<EMSP_Cert, curlyH, len16(curlyK_2_hat), curlyK_2_hat, s_2_hat, C_hat,'Host_CompleteJoin'>
    //sign response
    sig_m=sign(m,~cps)
  in
     [ In_S_B(I, CPS_I, m)
```

```
    , !PkCPS(CPS_I,pk(~cps))
    , !LtkCPS(CPS_I,~cps)
   ]
   --[ CPSReceivedRes(CPS_I)
    , Honest ( CPS_I )
    , OnlyOnce('CPS_Sign_Credential_Response')
    ]->
   [Out(<m, sig_m, 'CPS_Fwd_Res'>) ]

//Modification: Added verification of the CPS' signature
rule Host_Passthrough_2:
  let
    //inputs
    s_2_hat=aenc(~s_2,pk(KDF_EK(~TPM_EK_Seed)))
    m=<EMSP_Cert, curlyH, len16(curlyK_2_hat), curlyK_2_hat, s_2_hat, C_hat,'Host_CompleteJoin'>
  in
  [ In(<m, sig_m, 'CPS_Fwd_Res'>)
    , Host_State_05($PS, $AS,  pk(KDF_EK(~TPM_EK_Seed)), Q_PD, Q_SD, pk(cps), CPS_I)
   ]
  --[ Eq(verify(sig_m,m,pk(cps)), true)
    , Passthrough_ActivateCred2($PS, $AS)
    , OnlyOnce('Host_Passthrough_2')
    ]->
  [ Out_S($AS,$PS,< Q_SD, Q_PD, curlyH, len16(curlyK_2_hat), curlyK_2_hat, s_2_hat, '
    TPM2_ActivateCredentials_2'>)
  , Host_State_06($PS, $AS,  pk(KDF_EK(~TPM_EK_Seed)), Q_PD, C_hat)
  , !Host_Store_EMSP_Cert($PS, $AS,  EMSP_Cert, CPS_I) // CPS_I for Honest condition
  ]

rule TPM2_ActivateCredential_2:
  let
    //unwrap the inputs where needed
    curlyK_2_hat=senc(curlyK(K_2),k_e)
    s_2_hat=aenc(s_2,pk(KDF_EK(~TPM_EK_Seed)))
    Q=multp(~f, 'P1')
    Q_PD=<'DAA_public_data', Q>

    //recompute
    s_2_rec=adec(s_2_hat,e)           //retrieve s
    Q_N_rec=<'SHA256',H_SHA256(Q_PD)>     //calculate Q_N_rec which should be the same as Q_N
    k_e_1=KDF_a(s_2,'STORAGE',Q_N_rec)    //calculate k_e_1 which should be the same as k_e
    k_h_1=KDF_a(s_2,'INTEGRITY','NULL')   //calculate k_h_1 which should be the same as k_h
    curlyH_1=MAC(<len16(curlyK_2_hat),curlyK_2_hat,Q_N_rec>,k_h_1)
    curlyK_2_rec=sdec(curlyK_2_hat,k_e_1)
  in
  [  In_S($AS,$PS,< Q_SD, Q_PD, curlyH, len16(curlyK_2_hat), curlyK_2_hat, s_2_hat, '
    TPM2_ActivateCredentials_2'>)
    , !TPM_AES_Key($PS, aes_key)
    , !TPM_ENDORSEMENT_SK($PS,e, pk(KDF_EK(~TPM_EK_Seed)))
  ]
  --[
    Eq(curlyH_1,curlyH)
    , Eq(k_e, k_e_1)
    , CurlyK2_recomputed($PS, $AS)
    , OnlyOnce('TPM2_ActivateCredential_2')
    ]->
  [ Out_S($PS,$AS, < curlyK_2_rec, 'ret_TPM2_ActivateCredentials_2'>) ]

rule Host_JoinComplete:
  let
    //unwrap the inputs where needed
    curlyK_2_rec=curlyK(K_2_rec)

    //inputs from the issuer
    EMSP_Cert=<I,pkX,pkY>
    //pkX=PkX(x,'P2')
    //pkY=PkY(y,'P2')
```

```
490
491     //input from Host_State
492     Q=multp(~f, 'P1')
493     Q_PD=<'DAA_public_data', Q>
494     C_hat=senc(<A,B,C,D,u,j>,curlyK_2) //we decrypt these credentials by checking that curlyK_2_rec =
        curlyK_2
495
496     //recompute the hash
497     R_B_dash=calcRB(minus(multp(j,'P1'), multp(u,B)))
498     R_D_dash=calcRD(minus(multp(j,Q),multp(u,D)))
499     u_dash=H_n_8('P1',Q,R_B_dash,R_D_dash, A, B, C, D)
500   in
501   [ In_S($PS,$AS, < curlyK_2_rec, 'ret_TPM2_ActivateCredentials_2'>)
502     , !Host_Store_EMSP_Cert($PS, $AS,  EMSP_Cert, CPS_I)
503     , Host_State_06($PS, $AS,  pk(KDF_EK(~TPM_EK_Seed)), Q_PD, C_hat)
504   ]
505    --[
506        Eq(curlyK_2, curlyK_2_rec) //this allows C_hat to be decrypted
507     , Eq(u,u_dash)
508     , Eq(verifyCre1(A,pkY,B,'P2'),accept)
509     , Eq(verifyCre2(A,D,pkX,C,'P2'),accept)
510     , JoinCompleted($PS, $AS, pk(KDF_EK(~TPM_EK_Seed)))
511     , Commit(pk(KDF_EK(~TPM_EK_Seed)), I, <A, B, C, D>)
512     , Role ('Platform')
513     , Secret(pk(KDF_EK(~TPM_EK_Seed)), I, <A, B, C, D> )
514     , Honest ( $PS )
515     , Honest ( $AS )
516     , Honest ( pk(KDF_EK(~TPM_EK_Seed)) )
517     , Honest ( I )
518     , Honest ( CPS_I )
519     , OnlyOnce('Host_JoinComplete')
520     ]->
521    [ !Host_State_07($PS, $AS, pk(KDF_EK(~TPM_EK_Seed)), A, B, C, D)
522     , Host_Org_Creds($PS, $AS, pk(KDF_EK(~TPM_EK_Seed)), A, B, C, D)
523    ]
524
525 // The following rule allows a host to leak the credentials
526 rule Host_CredentialsReveal:
527   let
528     pke=pk(KDF_EK(TPM_EK_Seed))
529   in
530   [Host_Org_Creds($PS, $AS, pke, A, B, C, D)]
531   --[ KeyReveal('Host_OrgCred_Reveal', $AS)
532      , KeyReveal('TPM_OrgCred_Reveal', $PS)
533      , KeyReveal('PKE_OrgCred_Reveal', pke)
534     ]->
535   [Out(<A, B, C, D>)]
536
537 //================================================================
538 // Join Completed
539 //================================================================
```

Listing D.1.: Modified Symbolic Model of DAA Join (Based on [205])

```
1  lemma auth_injective_agreement_host_issuer:
2  "
3  All Iss pke n #i .
4    (
5      (
6        // For all commited JOIN sessions running between a platform (identified by is endorsement public
         key pke) and issuer on the term(s) n
7        Commit( pke, Iss, n ) @ i
8      )
9      ==>
10     (
11       (
12         // Implies there exists a running issuer on the same term
```

```
        (Ex #j .
          (
            Running( Iss, pke, n ) @ j
            &
            (#j<#i)
            &
            // And each commited JOIN session corresponds to a unique issuer run
            ( not(
              Ex Iss2 pke2 #i2 . ( Commit( pke2, Iss2, n) @ i2 & not(#i2=#i) )
              )
            )
          )
        )
      )
      |
      (Ex RevealEvent Entity #kr . KeyReveal(RevealEvent, Entity) @ kr & Honest(Entity) @ i)
    )
  )
"
```

Listing D.2.: Injective Agreement During the DAA Join (Based on [205])

```
lemma oracle_auth_non_injective_agreement_CPS_EV1:
"
All CPS pke n #i .
  (
    (
      // For all commited JOIN requests running between a platform and cps on the term(s) n
      CommitCPS( CPS, pke, n )  @ i
    )
    ==>
    (
      // Implies there exists a running platform on the same term
      (Ex #j . RunningEV( pke, CPS, n )  @ j)
      |
      (Ex RevealEvent Entity #kr . KeyReveal(RevealEvent, Entity) @ kr & Honest(Entity) @ i)
      |
      (Ex Entity #kr . KeyReveal('CPS_KeyReveal', Entity) @ kr & kr < i) // CPS key reveal out of scope
    )
  )
"
```

Listing D.3.: Non-Injective Agreement with Regard to Credential Requests