Securing Electric Vehicle Charging Systems Through Component Binding

Andreas Fuchs, Dustin Kern, Christoph Krauß(✉), and Maria Zhdanova

Fraunhofer SIT, Darmstadt, Germany
christoph.krauss@sit.fraunhofer.de

Abstract. In Vehicle-to-Grid (V2G) scenarios, Electric Vehicle (EV) batteries serve as distributed energy resources that help stabilize power supply through managed (dis)charging. The effective and safe grid integration is only possible when an Electric Vehicle Charging System (EVCS) responsible for the battery management and V2G communication is counterfeit-free and protected against malicious attacks. By manipulating the EVCS, adversaries can cause financial and physical damage and increase the risk of hazardous situations such as fire and traffic accidents. In this paper, we introduce `secEVCS`, a security architecture for EVCSs, which ensures that only a vehicle with a manufacturer-approved charging system can connect to the grid by securely binding all components of the EVCS. Our solution is based on the enhanced authorization functionality of the Trusted Platform Module (TPM) and protects against the installation of counterfeit products and re-use of secret data stored in scrapped EVCSs. We implemented `secEVCS` using a TPM 2.0 chip and the V2G protocol specified in the ISO 15118 standard to show the feasibility and to evaluate the performance of our solution.

Keywords: Electric vehicle charging · Security · TPM 2.0 · ISO 15118

1 Introduction

The worldwide adoption of EVs, i.e., fully battery-powered and plug-in hybrid vehicles, is growing, with the 5 million mark reached in 2018 [8]. The need to charge EV batteries causes an extra load on electric grids, but their storage capacity can be used by V2G services to handle power fluctuations [3]. The V2G technology allows EVs to communicate with the grid to optimize charging profiles, e.g., to limit the charging rate or to feed energy from batteries back to the grid during high demand. For this purpose, V2G communication protocols such as ISO 15118 [10] were developed. Using these protocols, an EV can inform the grid of its preferences (energy amount, departure time, etc.) and negotiate a grid-friendly (dis)charging schedule. The support for bidirectional power transfer services is provided by the vehicle's EVCS with two connected components: an Electric Vehicle Communication Controller (EVCC) and a Battery Management System (BMS) responsible for the V2G session handling and battery management, respectively. As the degradation rate of EV batteries is affected by the charging strategy, ensuring their correct operation during V2G sessions is equally important to servicing the grid [18]. Besides, Li-ion batteries are prone to overheating and can self-ignite due to improper (dis)charging control [1,21].

The possibility to cause safety hazards and the V2G connectivity can provide a strong incentive for malicious attacks aiming to subvert the functioning of the EVCS. For example, if an adversary manipulates the BMS part of the EVCS or replaces it with a tampered one, s/he can damage the battery by deliberately operating it outside of the safe range, which can eventually lead to its failure and the danger of fire or explosion [14,15]. In case several EVCSs connected to the grid are under adversarial control, they can be turned into a botnet of high-wattage devices for a coordinated attack aiming for local power outages or large-scale blackouts [20]. Another critical aspect is the usage of counterfeit components in EVCSs. The growing market and high price of EV batteries attract criminals selling expired or low-quality counterfeit spare parts, which do not meet regulatory standards and are potentially unsafe [17].

In this paper, we propose a new security architecture for EVCSs, further referred to as `secEVCS`, which guarantees that a vehicle participating in V2G services has a manufacturer-approved configuration of EVCC and BMS by securely binding these components. `secEVCS` uses a TPM in the EVCC and the Device Identifier Composition Engine (DICE) [25] in the BMS as security anchors. The general idea is to only allow access to a V2G authentication key, which is required for connecting to the grid, if the binding is successfully verified using the TPM's enhanced authorization functionality. `secEVCS` protects against the installation of counterfeit spare parts and re-use of secrets from scrapped EVCSs. We implemented `secEVCS` using a hardware TPM and ISO 15118 [10] for V2G communication[1] to evaluate `secEVCS` under realistic constraints. To our knowledge, TPMs have not been deployed in this scenario yet and the analysis of the trade-offs is missing. Our work aims to close this gap.

The rest of the paper is structured as follows: First, we introduce the necessary background on TPMs to be able to understand the paper in Sect. 2. Next, we discuss related work in Sect. 3. In Sect. 4, we define our system model and in Sect. 5 analyze safety-related security threats. Security and functional requirements are defined in Sect. 6. In Sect. 7, we introduce `secEVCS` before we describe and evaluate our prototype in Sect. 8. We discuss the applicability of our solution in Sect. 9 and conclude the paper in Sect. 10.

2 Background on Trusted Platform Modules (TPMs)

The TPM 2.0 Library Specification [23] defines a catalog of functionalities that can be used to build the second iteration of the TPM for different platforms.

[1] The ISO 15118 standard series is actively adopted by the industry, e.g., the CharIn network (www.charinev.org). While we focus on the current protocol specification, ISO 15118-2, we consider the 2nd edition draft, ISO 15118-20 [11], whenever relevant.

Accompanying the TPM specification, a TPM Software Stack (TSS) 2.0 specification [26] defines multiple Application Programming Interfaces (APIs) for different application scenarios. A TPM is a microchip designed to provide security-related functionalities, e.g., secure storage and usage of cryptographic keys. A central functionality in our solution is the enhanced authorization functionality. All objects of a TPM, e.g., cryptographic keys, can be authorized with a policy that must be satisfied in order to authorize an action on that object. Policy assertions are sent to the TPM before the command being authorized. Our solution also uses the internal Non-Volatile (NV) memory of a TPM, which retains content even if the power is off. This memory can be used to make keys of the TPM persistent but can also be allocated by applications to create indexed strongly monotonic NV counters. In the following, we list the TPM 2.0 commands and policy assertions relevant for our proposed solution.

- **TPM2_Create.** This command is used to create all kinds of objects for the TPM. This includes cryptographic keys usable for authenticating to external entities. During creation, an (enhanced authorization) policy can be provided that restricts usage of the created object.
- **TPM2_Sign.** This command calculates a signature using a private key created via *TPM2_Create*. These signatures can be used for authenticating a device or for asserting data integrity and origin.
- **TPM2_VerifySignature.** The TPM can verify a signature using a provided public key. This operation by itself is much faster when implemented in software and does not require the secure execution environment of a TPM. This operation is to be used in conjunction with *TPM2_PolicySigned*.
- **TPM2_NV_DefineSpace.** This command is used to define an NV index. Depending on the assigned index number, the NV index is either part of the user-owned (storage hierarchy) or of the platform-/OEM-owned (platform hierarchy) areas of the TPM. For the sake of this paper, only OEM-owned indices are used.
- **TPM2_NV_Increment.** This command is used to increment a TPM NV counter value.
- **TPM2_StartAuthSession.** In order to fulfill any authorization policy, the application needs to start a policy session using the *TPM2_StartAuthSession* command. Then the actual policy statements are subsequently satisfied by invoking the corresponding TPM commands.
- **TPM2_PolicyAuthorize.** This command allows the activation of policies *after* the definition of an object. In order to achieve this, a public key is registered with a policy. This policy element then acts as a placeholder for any other policy branch that is signed with the corresponding private key.
- **TPM2_PolicyNV.** The PolicyNV element provides the possibility to include NV-indices in the evaluation of a policy. Amongst other operations, it can be used to test whether an NV counter index has a certain value or whether it is smaller or greater.
- **TPM2_PolicySigned.** This policy element can be used to validate a signature before granting object usage. A public key is registered with the policy. In order to satisfy a *TPM2_PolicySigned*, a TPM generated nonce (from

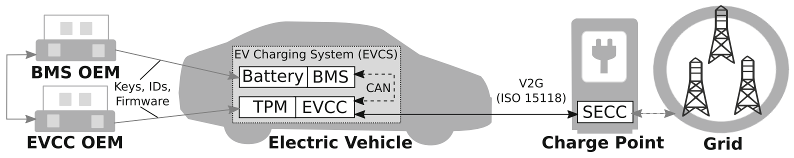


Fig. 1. System overview

TPM2_StartAuthSession) must be signed by the holder of the private key corresponding to the registered public key.

– **TPM2_PolicyTicket.** In order to speed up operations, a *TPM2_Policy Signed* can return a ticket that is valid for a certain amount of time. Instead of having to execute the same sequence of *TPM2_VerifySignature* and *TPM2_PolicySigned*, the ticket can be replayed into a *TPM2_PolicyTicket* command instead. This saves the round trips with the owner of the private key for a predefined amount of time. The expiration time is already denoted in the policy during its creation.

3 Related Work

The use of trusted computing in vehicle communication scenarios is commonly discussed. In [30,31], solutions for privacy-preserving EV charging and billing based on Direct Anonymous Attestation techniques using a TPM are introduced. In [29], the authors use TPM-based remote attestation for identity and integrity verification in the V2X scenario. The authors of [28] propose to use a Mobile Trusted Module for remote attestation in V2G networks. In their system, an EV directly sends its integrity metric to the grid server to verify its trustworthiness. The work in [7] presents a privacy-aware architecture for V2G networks. As part of this solution, a TPM in EV batteries is used for encrypted communication of charging status, accumulation of information in sealed storage and remote attestation. In [19], another security architecture for V2G networks is discussed using a TPM in EV batteries for remote attestation. In [5], the authors propose to include a TPM in a head unit (also called infotainment system) to realize protocols for secure update, remote attestation, and sensitive data protection.

In our previous work [4], we introduce TrustEV, a security architecture for secure provisioning, storage and usage of ISO 15118 credentials in an EV based on TPM 2.0. Similarly, the TPM is used to store and control access to authentication keys of the EVCC. The main feature of TrustEV is the direct import of the ISO 15118 keys created by a third party into the EVCC's TPM. TrustEV can be combined with `secEVCS` to additionally support secure components binding.

4 System Overview

Figure 1 gives an overview of our assumed system. An Electric Vehicle Charging System (EVCS) comprises a Battery Management System (BMS) with an integrated rechargeable battery and an Electric Vehicle Communication Controller (EVCC) with a TPM providing it with security services (cf. Sect. 2). As TPMs are common in modern cars [9], this assumption is not limiting.

The BMS's major function is to maintain the vehicle's battery within its safe operating range, to monitor its state (i.e., state-of-charge, state-of-health, and state-of-function) and to assess the available energy amount [2]. The BMS also controls battery cooling/heating, operates power switches, and exchanges charging-related data via a Controller Area Network (CAN) bus with the EVCC.

The EVCC is responsible for communication with a Supply Equipment Communication Controller (SECC) of a charge point during V2G service sessions and supports automated authentication of EVs (i.e., Plug&Charge). We assume an EV has to identify and authenticate itself against an SECC by means of a so-called *authentication key* stored in its EVCC before it can connect to the grid for charging. In ISO 15118, this key is part of the vehicle's OEM provisioning or contract certificate. When negotiating a charging schedule for a V2G session with the SECC, the EVCC queries the BMS on such parameters as battery state, allowed current and voltage. Together with the expected departure time and other user-defined charging preferences, this information is crucial for demand-side management aimed to improve efficiency and stability of the grid. Grid-friendly charging behavior can be awarded with reduced tariff rates. While charging, the EVCC periodically receives metering receipts from the SECC for signing that may later be used to bill the EV's driver for the charged energy.

The life-cycle of an EVCS and its components includes several stages. The BMS and the EVCC are produced by respective Original Equipment Manufacturers (OEMs) that provide firmware and cryptographic keys. An automotive OEM creates for an EV a unique Vehicle Identification Number (VIN)[2] and authentication key, while a battery OEM provides a BMS with a unique *identity key*. The cryptographic keys are assumed to be created by OEMs in a secure way and not leaked during manufacturing. When deploying a new EVCS in a vehicle, the EV's manufacturer defines a safety-approved configuration, by binding a BMS to an EVCC. Replacing or updating any of the EVCS's components can be carried out in an authorized repair shop, where a new approved configuration will be created. A drained EV's battery can also be replaced with a fully-charged one in a battery swap station operated by a battery swapping company [22]. We assume backend systems of the OEMs and service providers exchange information securely using a common Public Key Infrastructure (PKI).

5 Safety-Related Security Threats

The EVCS of a vehicle is a safety-critical system. The growing number of reports on self-ignition of EV batteries while charging or in driving [21], shows the potential for adversaries not only to damage EVs and their components but to harm their passengers or people in the vicinity with targeted attacks. In [15,16], the

[2] VINs mainly conform to two international standards ISO 3779 and US Standard FMVSS 115; a VIN is always 17 characters long.

authors propose a Security-Aware Hazard and Risk Analysis Method (SAHARA) and use it to identify threats for a BMS and estimate the risk. Threats for BMS and potential effects are also analyzed in [14]. Based on these analyses, we consider the following threat scenarios with their possible safety impact:

Configuration Tampering. An adversary replaces the BMS in the EVCS with one that is not approved by the OEM and/or under full control of the adversary. This could also be done by the EV's driver who aims to extend the range of the vehicle by upgrading the battery [6]. Such action would violate the OEM's warranty. **Impact:** The attack affects the EVCS's integrity and has multiple safety implications. The adversary can modify battery information, e.g., to indicate a larger capacity than given, and control battery functions to, e.g., ignore dangerous operating conditions like overheating in order to damage the battery or cause a fire [18]. Moreover, incompatible EVCS components can incorrectly interpret exchanged data, which can shorten the battery lifetime and lead to hazardous situations. Disrupting demand-side management would also affect the grid.

Charging Contract Hijacking. An adversary uses a scrapped EVCC storing the authentication key of a valid V2G user to charge her/his own vehicle on the uncovered account or to use the access profile to connect to the V2G service. **Impact:** The attack affects the confidentiality of the EV's key and the privacy of the previous user of the scrapped EVCC; the integrity and authenticity of V2G sessions is also breached. The latter can affect grid stability due to unexpected charging behavior or even cause blackouts if the attack is launched in a coordinated manner [20].

Counterfeit BMS. An adversary uses old BMSs with expired or malfunctioning batteries to produce and sell counterfeit products, which can still carry the label of the original manufacturer but are not certified for safe use. **Impact:** The attack affects system integrity and authenticity. Counterfeit batteries often lack required safety protections and can easily catch fire.

6 Security and Functional Requirements

To prevent the threats defined in Sect. 5 with `secEVCS`, we propose to enable access to an EV's authentication key needed to use V2G services, only if its EVCS is original, i.e., only if a verifiable binding between EVCC and BMS exists. This leads to the following security requirements, which must be fulfilled:

SR_1 *Secure private key storage and usage.* Private keys (e.g., authentication keys, identity keys) shall be protected against leakage during their storage and usage.

SR_2 *Restriction of key usage to trustworthy systems (Key usage authorization).* Access to private keys shall only be possible if the EVCS is trustworthy, i.e., the components configurations are approved by the manufacturer and are not manipulated.

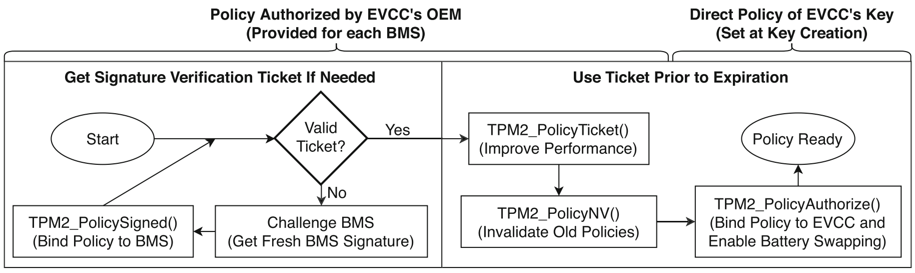

Policy Authorized by EVCC's OEM
(Provided for each BMS)
Direct Policy of EVCC's Key
(Set at Key Creation)
Get Signature Verification Ticket If Needed
Use Ticket Prior to Expiration
Start
Valid Ticket?
Yes
No
TPM2_PolicySigned() (Bind Policy to BMS)
Challenge BMS (Get Fresh BMS Signature)
TPM2_PolicyTicket() (Improve Performance)
TPM2_PolicyNV() (Invalidate Old Policies)
TPM2_PolicyAuthorize() (Bind Policy to EVCC and Enable Battery Swapping)
Policy Ready

Fig. 2. `secEVCS` policy verification steps

SR_3 *Revocation support.* It shall be possible to revoke BMS of an EVCS in case it is removed from an EV, so that it cannot be used in another manufacturer-approved EVCS configuration later on.

A security solution for EVCSs should bring clear benefits to the automotive and EV battery industry and consumers without unnecessary restricting legitimate usage scenarios. This results in the following functional requirements:

FR_1 *Minimal performance overhead.* A solution shall not cause undesirable delays in EV charging and shall meet timing constraints of standard V2G protocols. In ISO 15118, e.g., charging may be delayed by two uses of the EVCC's key and key use is bound by strict timing requirements (detailed in Sect. 8.2).

FR_2 *Support of legitimate component exchange.* Only legitimate entities shall be able to replace or swap the battery (including BMS) and/or EVCC while maintaining the manufacturer-approved EVCS configuration.

7 Solution

The general idea of `secEVCS` is to bind EVCC and BMS of an EV and to allow access to an authentication key only if this binding can be verified. The authentication key is securely stored and used in the EVCC's TPM and access is only possible if a TPM enhanced authorization policy is fulfilled. This policy includes the result of challenge-response protocol between EVCC and BMS.

`secEVCS` consists of an initial EVCS preparation phase for initializing and binding EVCC and BMS (cf. Sect. 7.1), the EVCS usage phase (during the lifetime of the EV) supporting the authorization of charging sessions and the swapping of batteries (cf. Sect. 7.2), and performance optimizations (cf. Sect. 7.3). Figure 2 shows the enhanced authorization policy verification steps as the central part of `secEVCS`, which are described in more detail below.

7.1 EVCS Preparation

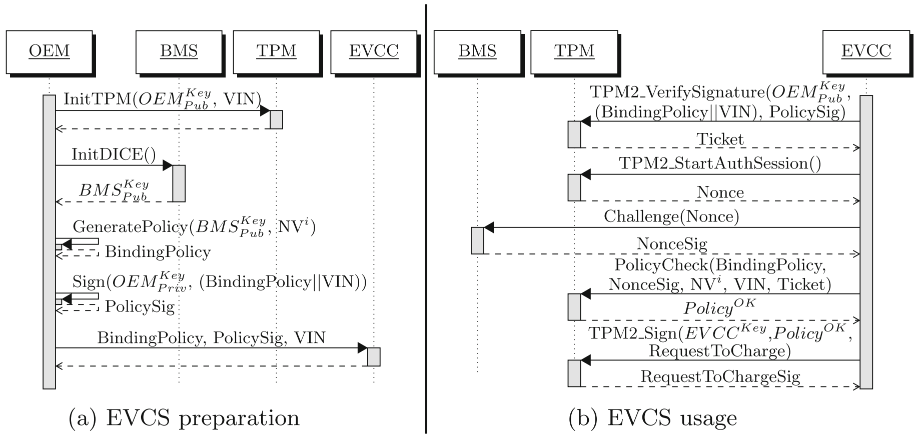
EVCC Preparation. During manufacturing, the OEM generates an authentication key on the EVCC's TPM. This authentication key comes with an authorization policy (*TPM2_PolicyAuthorize()*) that refers to an OEM public key (cf. Fig. 2 on the right) for policy statements. To use the authentication key, the EVCC software needs to present a policy to the TPM that was authorized (signed) by the OEM. Thus, the newly created key cannot be accessed directly after its generation. The OEM needs to explicitly issue (and sign) a policy statement that describes, under which conditions the authentication key can be used. We use the EV's VIN as policy reference value of the authentication key's policy. This way, this key can only be accessed if a policy is fulfilled that was authorized by the OEM with the corresponding VIN denoted during key generation, i.e., a signed policy addresses the intended EVCC only and cannot be copied to other EVCCs. The EVCC preparation process is represented by InitTPM() in Fig. 3a. If key generation on the TPM is not possible, keys can be also generated outside and imported into the EVCC's TPM (e.g., in ISO 15118 using TrustEV [4]).

BMS Preparation. The BMS is equipped with a DICE [25] (cf. InitDICE() in Fig. 3a), as a cheap alternative to a TPM suitable for highly constrained embedded systems [12]. DICE generates a unique device Identifier (ID) based on a globally unique secret and a measurement of the device's first mutable code using a cryptographically secure one-way function. Hence, any persistent attack to the BMS results in the generation of a different device ID. As the DICE is trusted and has exclusive access to the unique secret, it is impossible for an attacker to recover the secret or generate a valid device ID after a persistent attack. The BMS can use the DICE-generated ID to secure its identity key (e.g., by using the ID as seed to a Key Derivation Function (KDF) and using the resulting key to encrypt the identity key before it is stored). This way, the BMS's identity key is also protected from persistent attacks. With this key, the BMS can authenticate itself using a public key signature. As the BMS' public identity key is required for the binding between EVCC and BMS (see next paragraph), the key is read out by the BMS' OEM and passed to the EVCC's OEM (cf. OEM in Fig. 3a).

BMS and EVCC Binding. At this step, the EVCC's OEM needs to issue (sign) a respective policy (cf. BindingPolicy and PolicySig in Fig. 3a). The policy consists of a *TPM2_PolicySigned()* containing the public key of the BMS. To fulfill this condition, a nonce generated by the TPM must be signed with the BMS' private key and the signature validated by the TPM. Additionally, the policy contains a *TPM2_PolicyNV()* statement that links this policy to a monotonic counter inside the TPM. If a BMS binding needs to be revoked in the future, the OEM can increment the TPM's counter. The signature over this policy by the OEM also includes the VIN as policy reference value as mentioned above. This binding can happen in conjunction with the initial key generation or at a later stage.

7.2 EVCS Usage

Charging Authorization. Access to the authentication key is only possible if the BMS and EVCC binding is successfully verified. This process is shown in Fig. 3b (PolicyCheck() aggregates all policy-related validations). The EVCC first loads the authorized policy (i.e., the BMS binding policy) and policy reference


OEM
BMS
TPM
EVCC
InitTPM(OEM_{Pub}^{Key}, VIN)
InitDICE()
BMS_{Pub}^{Key}
GeneratePolicy(BMS_{Pub}^{Key}, NV^i)
BindingPolicy
Sign(OEM_{Priv}^{Key}, (BindingPolicy||VIN))
PolicySig
BindingPolicy, PolicySig, VIN
BMS
TPM
EVCC
TPM2_VerifySignature(OEM_{Pub}^{Key}, (BindingPolicy||VIN), PolicySig)
Ticket
TPM2_StartAuthSession()
Nonce
Challenge(Nonce)
NonceSig
PolicyCheck(BindingPolicy, NonceSig, NV^i, VIN, Ticket)
$Policy^{OK}$
TPM2_Sign($EVCC^{Key}$,$Policy^{OK}$, RequestToCharge)
RequestToChargeSig

(a) EVCS preparation (b) EVCS usage

Fig. 3. EVCS sequence diagrams

value, i.e., VIN, and verifies the signature using the OEM's public key. The result is a so-called signature validation ticket. Then, the EVCC starts a policy session (*TPM2_StartAuthSession()*) and sends the session's nonce as a challenge to the BMS. The BMS signs the nonce with its private key and returns the signature and its public key. The EVCC extends its session with a validation of the BMS' signature (*TPM2_PolicySigned()*) and the comparison of the NV-counter (*TPM2_PolicyNV()*). The BMS binding is authorized using the signature validation ticket (*TPM2_PolicyAuthorize()*). After this, the policy session is in a state that grants access to the EVCC's key operations and the EVCC can issue a *TPM2_Sign()* operation to authenticate itself against the charge point.

Battery Swapping. A battery swapping company needs to maintain a backend connection to the OEMs and perform the above BMS binding process. To invalidate the binding to the old BMS, the OEM increments the TPM's counter and then issues (signs) a new policy for the new BMS and the new counter value.

7.3 EVCS Enhancements for Better Performance

The process for key usage described in Sect. 7.2 requires the EVCC to challenge the BMS and perform the policy session assertions for each access to the authentication key. This can lead to undesirable delays when trying to charge an EV (e.g., in our tests it took on average 2.4 s; cf. Sect. 8.2). This can be easily avoided by sending challenges to the BMS independent of the charging sessions and pre-calculating the entire policy session. For example, the EVCC can send a challenge whenever the charging port lid is opened. This way, a correct policy session is always available before the authentication key is to be used.

Another issue is that the EVCC has to send a new challenge to the BMS each time it needs to use the authentication key. This can delay communication protocols between EVCC and SECC using this key not only for charge authorization, but also, e.g., to sign metering receipts. A low performance of the BMS Electronic Control Unit (ECU)[3] and a low throughput of a CAN bus[4], this process (estimated to about 5.8 s) may exceed timing constraints of the protocol.

We address this issue by using a shortcut in the *TPM2_PolicySigned()* command. The command can output a ticket upon validation, which can be used in future policy sessions (within the expiration time) using the *TPM2_PolicyTicket()* command as a replacement (cf. Fig. 2 on the left). This expiration time should not be too short (to gain a speedup) and neither too long (to restrain attacks). An expiration time of 5 min is a good starting point to give a user enough time to initiate charging, while still preventing potential attacks. These 5 min provide enough time to start a second policy session from the beginning.

8 Implementation and Evaluation

In this section, we evaluate our proposed solution. We use ISO 15118-2 [10] as communication protocol between EVCC and SECC. We describe the implemented prototype and evaluate the added overhead. A minimal overhead is important for the usability of `secEVCS` in terms of compliance to the timing constraints of ISO 15118 on EVCC signature creation as well as for user convenience. Additionally, ISO 15118-20 [11], the upcoming successor of [10], allows for even tighter timing constraints, which are also considered in the evaluation.

8.1 `secEVCS` Implementation

Our prototype was implemented using three Raspberry Pi 3 Model B running Linux kernel 4.14 to simulate the EVCC, BMS, and SECC. The EVCC-Pi is equipped with an Infineon Iridium 9670 TPM 2.0. EVCC and BMS communicate over regular Ethernet, while SECC and EVCC communicate over power-line communication (PLC) Stamp micro 2 EVBs (similar to PLC over a charging cable). Our test-bed is shown in Fig. 4.

To execute TPM commands, we use the TPM2-TSS[5] and as ISO 15118 implementation we use RISE-V2G[6], integrated with the TrustEV implementation from [4] for *EVCC Preparation* (cf. Sect. 7.1). The challenge-response communication between EVCC and BMS is implemented using the Secure Shell (SSH) protocol [13] to simulate any added security means on the automotive bus.

The expiration time of BMS signatures is set to 5 min. Challenges are sent to the BMS 5 s before start of ISO 15118 communication (to simulate the

[3] It can take an ARM Cortex-M0+ without performance optimizations up to 3649 ms to create a signature using the algorithm and parameters defined by ISO 15118 [27].

[4] Transmitting 16 byte nonce, 64 byte EC public key, and 64 byte ECDSA signature in 18 extended CAN frames (16 bytes each with 8 bytes data and 7 bits inter-frame spacing) with 125 kbps Low-Speed CAN takes about 20 ms under optimal conditions.

[5] TPM2-TSS: https://github.com/tpm2-software/tpm2-tss.

[6] RISE-V2G: https://github.com/V2GClarity/RISE-V2G.

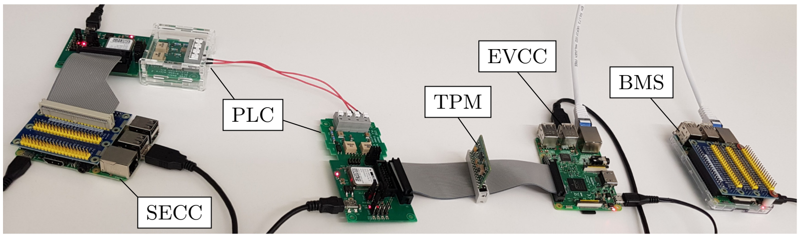


Fig. 4. Test-bed setup

time from opening the charging port lid to plugging in the charging cable) and a minute before the current signature expires. After receiving a signature, *TPM2_PolicySigned()* is called to retrieve the verification ticket. The ticket is used by two processes to pre-calculate multiple policy sessions concurrently. When the authentication key is used in ISO 15118, one of these pre-calculated policies is consumed. We only use two pre-calculation processes along with the ticket generation process, to not exceed three concurrent authorization sessions. While a TPM must be able to support 64 active sessions, it must only be able to hold 3 of those in RAM at a time [24]; hence, exceeding this limit would decrease performance on TPMs that only support the minimums from [24].

8.2 Performance Evaluation

During performance evaluation, we measured the computational overhead created by our prototype from Sect. 8.1 compared to the default RISE-V2G implementation. All measurements were repeated 100 times each using Java's *System.nanoTime()*. During a charging loop, the EVCC alternates between sending charging status and signed metering receipt messages. It tries to send them as fast as possible, reaching 121.9 ms between consecutive receipts. For our measurements, the EVCC sent 10 metering receipts for each of the 100 charging loops.

The time for signing ISO 15118 messages with default RISE-V2G was 15.7 ms and with `secEVCS` 469.8 ms. For comparison, without the parallel pre-calculated policies, the average signature time was 1119.8 ms, and without any of the performance optimizations for `secEVCS`, i.e., with an on-demand challenge to the BMS for each key usage and no policy pre-calculation, the time for signing ISO 15118 messages was 2437.8 ms. Our measurements for `secEVCS` are shown in Fig. 5 (signature #0 is for charge authorization and #1–10 for metering receipts).

With our setup, the time from sending a challenge to the BMS until receiving a signature was 277.6 ms. In Sect. 7.3, we discussed a more realistic device configuration. Extrapolating our measurements to low power ECUs and CAN bus, the measurements for BMS signatures would increase to 3669 ms, leading to ISO 15118 message signing time of 5829.28 ms for `secEVCS` without optimizations. This correlates to the head time used for pre-calculation of sessions and the use of the improvements proposed in Sect. 7.3.

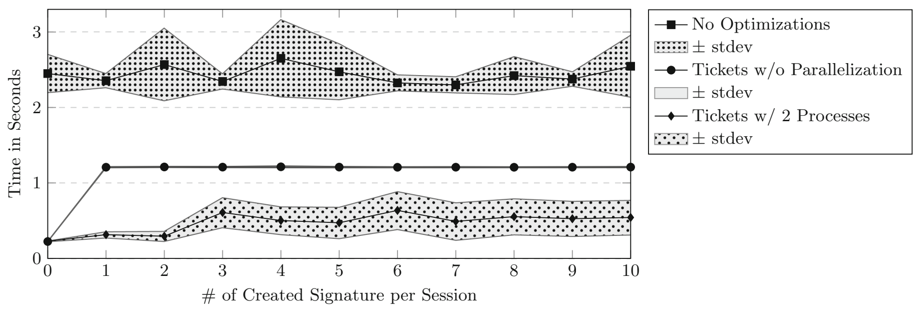


Fig. 5. Mean times of signature creation in a charging session

It is worth noting that with `secEVCS` there was a significant difference in the times for signing charge authorization requests compared to metering receipts. The former was on average 228.8 ms, whereas the latter – 493.9 ms. Also, the mean time for signing the first 2 receipts of each charging loop was 304.3 ms and the mean time for the 3rd to 10th was 541.3 ms. This is because we have only two processes to pre-calculate policy sessions. Hence, when starting a charging session, there are two policies ready to use and if more than two signatures need to be created, new policy sessions need to be calculated at run-time.

In our setup with 2 parallel policy sessions (n_p), 121.9 ms between metering receipts (t_m), and the signature time of 228.8 ms (t_s) there are only 472.6 ms ($= n_p \times t_m + (n_p - 1) \times t_s$) for policy pre-calculation. Anything above that will increase the signature time. With an average time for policy calculation of 737.4 ms, i.e., an overrun of 264.8 ms, this gives about 500 ms for signatures without full policy pre-calculation. While this should lead to alternating signature times (after a signature with $t_s = 500$, the available time for pre-calculation is 743.8 ms which should allow for a fast signature), we did not experience this effect. Instead, as a result of the parallelization, the times for the 3rd metering receipt signature onward were much less predictable with a standard deviation of 231 ms compared to the times for the first 2 with a standard deviation of 56.4 ms and the time for the authorization signature with a standard deviation of 4.8 ms.

Since in ISO 15118 at most the first two signatures are time-critical, i.e., signing a request for a new authentication key and signing the charge authorization request afterwards can delay the charging start, we argue that the achieved results are acceptable for the use-case. Regarding ISO 15118 compatibility, the only requirements affected by the increased EVCC signature time are the *V2G_EVCC_Sequence_Performance_Time* of 40 s (time for the EVCC to send its next request after a response from the SECC) and the *V2G_SECC_Sequence_Timeout* of 60 s (timeout of the SECC for waiting on the next EVCC request). Even without the performance optimizations, `secEVCS` stays well within the relevant limits. However, in the 2nd edition ISO 15118-20 [11], the timeout mechanism for metering receipts was changed. The SECC may define its own arbitrary timeout in seconds. Hence, a minimal timeout of 1 s is possible and only the optimized `secEVCS` would be able to meet this minimum (cf. Fig. 5).

9 Requirements Coverage Discussion

To prevent safety-related threats from Sect. 5, `secEVCS` verifies the binding between the EVCS components prior to charging. In Sect. 6, we defined the requirements that need to be satisfied by a secure and usable solution. Below we informally discuss how these requirements are covered by `secEVCS`.

In `secEVCS`, the EVCC's authentication key is generated and stored in the controller's TPM and can only be accessed by this TPM and used only in its shielded location. Thus, this private key is protected from any attacks that read keys from the memory. As the binding between EVCC and BMS is validated based on a signature by the BMS's identity key, secure storage of this private key is essential for the overall security, too. To protect this key, `secEVCS` uses DICE. As our solution does not require the BMS to use the TPM's advanced security functions, a smaller security architecture with less hardware requirements was chosen as more appropriate. Due to the relatively high cost of a TPM, it is also desirable to limit their number to externally facing ECUs. Thus, `secEVCS` meets the security requirement *Secure private key storage and usage (SR_1)*.

The TPM always verifies, whether the EVCC is in a trustworthy state and whether the BMS defined in the configuration provided by its OEM is present, before allowing access to the authentication key. Thus, if an attacker has manipulated or replaced the BMS, or uses a scrapped controller, the EVCC will not be able to authenticate itself for using V2G services. This corresponds to the security requirement *Restriction of key usage to trustworthy systems (SR_2)*.

The security requirement *Revocation support (SR_3)* is fulfilled by the validation of the value of a monotonic counter inside the EVCC's TPM, which can be incremented each time an expired or malfunctioning BMS is exchanged in a repair shop. This way, it will not be possible to use this BMS together with the EVCC for charging because it is not part of the approved configuration anymore.

The functional requirement *Minimal performance overhead (FR_1)* is met as explained in detail in Section 8.2. Requirement *Support of legitimate component exchange (FR_2)* is also fulfilled since only an authorized OEM can register a new BMS with an EVCC's TPM by sending an updated policy.

10 Conclusion

Recent studies indicate fire and traffic incidents caused by manipulated EVCSs as a major safety concern for EVs. In this paper, we proposed `secEVCS`, a new security architecture aiming to prevent harmful situations by allowing only vehicles with manufacturer-approved charging systems to (dis)charge electric energy at charge points. This guarantee is achieved through securely binding the components EVCC and BMS responsible for charging authentication and management using the enhanced authorization feature of the EVCC's TPM. This binding is verified each time the EV wants to use its authentication credential, which turned out to be challenging with regard to user convenience and communication timeouts. In order to evaluate our solution within realistic constraints, we implemented `secEVCS` using a TPM 2.0 chip and ISO 15118-2 [10] as a V2G protocol. Also, the upcoming ISO 15118-20 [11] with harder timeouts was considered in our evaluation. While the performance overhead is acceptable for the use case and within the timing constraints of ISO 15118-2, a straightforward approach of TPM-based component binding cannot meet the new requirements. With the new edition, conformance to the standard can only be guaranteed if all proposed `secEVCS` performance optimizations are in place.

Our results provide a useful reference for future work that can address the shown limitations (e.g., timing conditions or runtime attacks on EVCS) or adopt `secEVCS` as a security anchor in broadened scenarios, e.g., secure load management. We also plan a collaboration with industry as part of technology transfer.

Acknowledgments. The work was partly funded by the Federal Ministry for Economic Affairs and Energy (BMWi) under the project “LamA-connect” (01MZ19005A) and the TALENTA program of the Fraunhofer-Gesellschaft.

References

1. Blum, A.F., Long, R.T.J.: Hazard assessment of lithium ion battery energy storage systems. Final report, February 2016
2. Brandl, M., et al.: Batteries and battery management systems for electric vehicles. In: Design, Automation Test in Europe Conference Exhibition (DATE) (2012)
3. Clement-Nyns, K., Haesen, E., Driesen, J.: The impact of vehicle-to-grid on the distribution grid. Electr. Power Syst. Res. **81**(1), 185–192 (2011)
4. Fuchs, A., Kern, D., Krauß, C., Zhdanova, M.: TrustEV: trustworthy electric vehicle charging and billing. In: Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing (2020)
5. Fuchs, A., Krauß, C., Repp, J.: Advanced remote firmware upgrades using TPM 2.0. In: Hoepman, J.-H., Katzenbeisser, S. (eds.) SEC 2016. IAICT, vol. 471, pp. 276–289. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33630-5_19
6. Gaton, B.: NZ company offers solution to Nissan Leaf owners wanting a bigger battery. The Driven (2019). thedriven.io/2019/08/13/nz-company-offers-solution-to-nissan-leaf-owners-wanting-a-bigger-battery/
7. Ghosh, D.P., Thomas, R.J., Wicker, S.B.: A privacy-aware design for the vehicle-to-grid framework. In: 2013 46th Hawaii International Conference on System Sciences, pp. 2283–2291. IEEE (2013)
8. IEA: Global EV outlook 2019 (2019). www.iea.org/reports/global-ev-outlook-2019
9. Infineon: Volkswagen relies on TPM from Infineon (2019). www.infineon.com/cms/en/about-infineon/press/market-news/2019/INFATV201901-030.html
10. ISO/IEC: Road vehicles - Vehicle-to-Grid Communication Interface - Part 2: Network and application protocol requirements. ISO Standard 15118–2:2014 (2014)
11. ISO/IEC: Road vehicles - Vehicle-to-Grid Communication Interface - Part 2: Network and application protocol requirements. ISO/DIS 15118–2:2018 (2018)
12. Jäger, L., Petri, R., Fuchs, A.: Rolling DICE: lightweight remote attestation for COTS IoT hardware. In: Proceedings of the 12th International Conference on Availability, Reliability and Security. ARES (2017)

13. Lonvick, C.M., Ylonen, T.: The Secure Shell (SSH) Transport Layer Protocol. RFC 4253, January 2006. https://rfc-editor.org/rfc/rfc4253.txt
14. Lopez, A.B., Vatanparvar, K., Deb Nath, A.P., Yang, S., Bhunia, S., Al Faruque, M.A.: A security perspective on battery systems of the internet of things. J. Hardware Syst. Secur. (2017)
15. Macher, G., Sporer, H., Berlach, R., Armengaud, E., Kreiner, C.: SAHARA: a security-aware hazard and risk analysis method. In: 2015 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 621–624, March 2015
16. Macher, G., Höller, A., Sporer, H., Armengaud, E., Kreiner, C.: A combined safety-hazards and security-threat analysis method for automotive systems. In: Koornneef, F., van Gulijk, C. (eds.) SAFECOMP 2015. LNCS, vol. 9338, pp. 237–250. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24249-1_21
17. Peresson, S.: Counterfeit automotive parts increasingly putting consumer safety at risk. WTR (2019). www.worldtrademarkreview.com/anti-counterfeiting/counterfeit-automotive-parts-increasingly-putting-consumer-safety-risk
18. Sagstetter, F., et al.: Security challenges in automotive hardware/software architecture design. In: 2013 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 458–463, March 2013
19. Saxena, N., Grijalva, S., Chukwuka, V., Vasilakos, A.V.: Network security and privacy challenges in smart vehicle-to-grid. IEEE Wirel. Commun. **24**(4), 88–98 (2017)
20. Soltan, S., Mittal, P., Poor, H.V.: BlackIoT: IoT botnet of high wattage devices can disrupt the power grid. In: 27th USENIX Security Symposium (USENIX Security 18), pp. 15–32. USENIX Association, Baltimore, August 2018
21. Sun, P., Bisschop, R., Niu, H., Huang, X.: A review of battery fires in electric vehicles. Fire Technology (2020)
22. Tillemann, L., McCormick, C.: The faster, cheaper, better way to charge electric vehicles. WIRED (2018). www.wired.com/story/the-faster-cheaper-better-way-to-charge-electric-vehicles/
23. Trusted Computing Group: Trusted Platform Module Library Specification, Family 2.0, Level 00, Revision 01.16 edn., October 2014
24. Trusted Computing Group: PC Client Platform TPM Profile (PTP) Specification, Family 2.0, Revision 00.43 edn., January 2015
25. Trusted Computing Group: Hardware Requirements for a Device Identifier Composition Engine. Specification Family 2.0 - Level 00 Revision 78, March 2018
26. Trusted Computing Group: TCG TSS 2.0 Overview and Common Structures Specification, Version 0.90 Revision 03 edn., October 2019
27. Tschofenig, H., Pégourié-Gonnard, M.: Performance investigations. IETF Proceeding 92 (2015)
28. Wang, S., Wang, B., Zhang, S.: A secure solution of V2G communication based on trusted computing. In: 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID), pp. 98–102. IEEE (2019)
29. Xu, C., Liu, H., Li, P., Wang, P.: A remote attestation security model based on privacy-preserving blockchain for V2X. IEEE Access **6**, 67809–67818 (2018)
30. Zelle, D., Springer, M., Zhdanova, M., Krauß, C.: Anonymous charging and billing of electric vehicles. In: Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES, pp. 22:1–22:10. ACM (2018)
31. Zhao, T., Zhang, C., Wei, L., Zhang, Y.: A secure and privacy-preserving payment system for electric vehicles. In: 2015 IEEE International Conference on Communications (ICC), pp. 7280–7285. IEEE (2015)