Cloud Controlled EV Charging - A Threat to the Power Grid and a Countermeasure

Niklas Goerke
FZI Research Center for Information Technology
Karlsruhe, Germany
Karlsruhe Institute of Technology
Karlsruhe, Germany
goerke@fzi.de

Maximilian Staab
FZI Research Center for Information Technology
Karlsruhe, Germany
staab@fzi.de

Ingmar Baumgart
FZI Research Center for Information Technology
Karlsruhe, Germany
Karlsruhe Institute of Technology
Karlsruhe, Germany
baumgart@fzi.de

Abstract

Electric Vehicle (EV) manufacturers operate cloud infrastructures that can control the charging operations of connected EVs. An attacker gaining control over an EV manufacturer's cloud could perform load-altering attacks (LAAs) on the power grid by simultaneously sending charging commands to the connected EVs. This implies that the clouds of large EV manufacturers are effectively single points of failure to the power grid. This work presents an analysis of the technical implementation of the communication from the EV manufacturer smartphone app to the manufacturer cloud. Based on this analysis, we propose a change to the currently used communication concept: the use of end-to-end integrity protection to reduce the impact of a malicious cloud whilst keeping the impact on user experience to a minimum. Our results indicate that none of the EVs analyzed uses a communication architecture that provides protection in case of a compromised cloud. Through the end-to-end integrity protection scheme, we demonstrate that there is a countermeasure that EV manufacturers can adopt.

CCS Concepts

• **Hardware** → *Smart grid*; *Power networks*; • **Security and privacy** → *Security protocols.*

Keywords

Electric Vehicles, cyberattacks, cloud, power grid, stability, end-to-end integrity

ACM Reference Format:
Niklas Goerke, Maximilian Staab, and Ingmar Baumgart. 2026. Cloud Controlled EV Charging - A Threat to the Power Grid and a Countermeasure. In *ACM Sustainability Week 2026 (ACM Sustainability Week Companion '26), June 22–25, 2026, Banff, AB, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3765611.3815144

1 Introduction

Previous research [14] has shown that attackers can carry out attacks on the power grid if they are able to control the power consumption of large amounts of devices. EVs are examples of such devices that could be miscontrolled not only by attacking EVs themselves but via any authoritative entity that can send start/stop charging commands to the EV (e.g. the manufacturer's cloud). Recent research[7] showed that the amount of EVs that need to be miscontrolled to affect the stability of the power grid is relatively small. In the presented work, we address this issue by answering the following research questions.

(1) What concepts are currently used to provide resilience for successful attacks on clouds that control EV functions?
(2) Can the use of end-to-end integrity protection of control commands improve security without negatively impacting user experience?

The results can be used to reduce the risk of negative impacts on EVs and the stability of the power grid.

2 Related Work

The foundational work on the topic of miscontrolled Internet of Things (IoT) devices was presented by Soltan et al. [14]. The authors identified a new class of attacks called MadIoT (Manipulation of demand through IoT) and showed the potential impact on the simulated 9-bus WSCC grid.

Goerke et al. have calculated the impact of miscontrolled Distributed Energy Resources (DERs) on the European power grid [7]. In [2, 8, 12], the authors present in detail the effect that synchronized start and stop of charging operations can have on the New York and Polish power grids, as well as a local transmission system. Maleki et al. present a recent review of load altering attacks on the power grid using IoT devices [9].

Abazari et al. have analyzed the impact of EV based LAA [1].

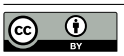
In [13], Sayed et al. have shown that EVs can be used to counteract LAA, suggesting that EVs possess sufficient capacity to influence the stability of the power grid; in their case, beneficially.

Multiple authors [3, 10, 11], have presented work on the use of End-to-end security protocols for IoT devices. They discuss implementation details for a variety of specific protocols, showing that the use of End-to-End security is possible.

Previous works have focused on the feasibility of performing LAA on power grids, the vulnerability of DERs or ideas for IoT and cloud architecture. This work closes a gap in considering a specific attack scenario in which an attacker with control over the cloud of an EV manufacturer can perform an LAA on the power grid.

3 Our Contribution

In this work, we analyze the communication of four EV manufacturer apps to the manufacturer cloud to derive insight on the



This work is licensed under a Creative Commons Attribution 4.0 International License.
ACM Sustainability Week Companion '26, Banff, AB, Canada
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2199-1/26/06
https://doi.org/10.1145/3765611.3815144

protection mechanisms used. We find that the EV manufacturer clouds are in a position to issue charging commands to all connected EVs and thus are a single point of failure to the power grid. We describe a countermeasure based on end-to-end integrity protection that can be implemented with minimal impact on usability.

4 Problem Analysis

The threat of misdirecting EVs through the manufacturer's clouds on the stability of the power grid is based on two issues. First, we describe the vulnerability of power grids to LAA. Afterwards, we analyze the concept employed for remote controlling EV functionalities and show how it could be abused by an attacker who gains control of a manufacturer's cloud.

4.1 Power Grids and Load Altering Attacks

This work uses the European Network of Transmission System Operators for Electricity (ENTSO-E), the pan-European power grid, as an example, but the fundamental mechanics and mechanisms also apply to other power grids. ENTSO-E operates at an AC frequency of 50 Hz. Any imbalance in supply and demand influences this frequency: if too little energy is provided to cover demand, the frequency drops. In contrast, if too much energy is provided, the frequency increases. In ENTSO-E a three-tier operational reserve is used to keep the frequency stable. The frequency containment reserve (FCR) activates automatically and instantly and provides positive or negative balancing power proportional to any frequency deviation thus stabilizing it back to the allowed range (usually between 49.99 and 50.01 Hz). The automatic frequency restoration reserve (aFRR) is activated in case the FCR is needed over a prolonged period. It has a lead-time of up to 30 seconds until the first reaction and a ramp-up time of 5 minutes until the full balancing power needs to be provided. The third tier is provided by the manual Frequency Restoration Reserve (mFRR) which has an even longer lead time of 5 minutes.

If a surge or drop in load occurs within 30 seconds, neither aFRR nor mFRR can activate quickly enough due to their ramp-up times. Currently, 3,000 MW of FCR are being provided in ENTSO-E. Any surge or drop of more than 3,000 MW can only be partially compensated for and will thus result in frequency deviations outside of the allowed range. The European Regulation defines that load shedding should be performed if the frequency drops to 49 Hz [4].

Dabrowski et al. [5] modeled that, attackers only need to control 4,500 MW to push the frequency below 49 Hz. Goerke et al. have shown that this amount of power can be controlled by an attacker who manages to misdirect the behavior of relatively small proportions of DER, such as 4.7% of all EVs, 16.6% of all Battery Energy Storage System (BESS) or 3.9% of all Photovoltaic Inverter (PV-I) predicted for Germany in 2030 [7]. For many of the DERs, there are single manufacturers with a large enough market share to provide these proportions[1]. Thus, an attacker who manages to gain control over all DERs supplied by only one manufacturer could be able to carry out static LAA against the power grid. For EVs, this number must be corrected to reflect only the EVs connected to the power grid at a time. On the other hand, the amount of power is a conservative estimate, as it only considers static LAA and disregards the limited capacity of power lines.

4.2 Remote Controlling EVs

In order to control the power input or output of a DER, an attacker does not need to attack the DER itself. Depending on the DER and the context, there are multiple influential entities that could be attacked. Local entities such as the owner's smartphone or a local Energy Management System (EMS) and central entities such as the manufacturer's cloud.

This work focuses on the specific use case of smartphone applications that an owner can use to send commands to DERs. Due to the limited range of local connections, e.g. Bluetooth, the communication is usually carried out through the internet and routed through the manufacturer's cloud. We find that manufacturer's clouds themselves do not commonly issue control commands themselves but only forward control commands from users' smartphones. We use EVs as a specific class of DERs. Due to the fact that their main use case is transporting people and goods, they need not only fulfill the requirements based on handling electric energy as other DERs do, but additional requirements that arise from standard vehicle usage such as locking, air-conditioning control, etc. We discuss the applicability of our findings on other classes of DERs in Section 8

We have analyzed the currently used implementation of app to cloud communication on the basis of four different EVs:

- Kia EV6
- MG 4
- Tesla Model 3
- Volkswagen ID. Buzz (VW ID. Buzz)

4.2.1 Communication from App to Cloud. In the currently used concept, the user's smartphone sends an authenticated command to the manufacturer's cloud, which in turn forwards it to the EV. For the EVs analyzed in this work, manufacturers implement access control at the cloud level, allowing only authorized users to access their respective EV data and send commands to it.

4.2.2 Analysis for Kia EV6 and VW ID. Buzz. We used a Google Pixel 8a smartphone for the analysis and manually repackaged the Kia EV6 APK to embed the Frida Gadget library[2]. By inserting a load instruction into the main activity's entry point, the reverse engineering toolkit Frida gains runtime instrumentation capabilities without root access on the smartphone. We then dynamically patched the certificate validation within the application's context to enable the Kia EV6 app to trust our self-signed Transport Layer Security (TLS) certificate.

For the analysis of the VW ID. Buzz app, we used a rooted Android emulator (Android 14, API 34), in which we run Frida Server to inject instrumentation scripts that log function arguments and bypass security checks. The root detection mechanism of Android is bypassed by intercepting package manager queries, system property reads, and shell command executions to mask the presence of our tooling. We managed to bypass TLS certificate validation by hooking into the Android TrustManager implementations and popular HTTP libraries (OkHTTP, Cronet, Apache HTTP).

[1] e.g. there are more than 10 Mio. vehicles registered in Germany that were manufactured by Volkswagen (without subsidiaries), corresponding to a market share of 20% (see https://www.kba.de/DE/Statistik/Fahrzeuge/Bestand/MarkenHersteller/marken_hersteller_node.html

[2] See https://frida.re/

Both of these paths allowed us to intercept all network traffic from the Kia EV6 and VW ID. Buzz app by performing a Machine-in-the-Middle (MITM) attack using mitmproxy[3] on the analysis host. Our analysis shows that for both Kia EV6 and VW ID. Buzz, the communication from the smartphone app to the manufacturer's cloud is protected using only TLS. Both Kia EV6 and VW ID. Buzz use standard OAuth-Tokens to authenticate requests from users. This prevents a network-based Dolev-Yao attacker [6] from reading and manipulating the data communicated. The absence of additional protection mechanisms implies that any data sent from the smartphone app to the EV is protected in transit to and from the manufacturer's cloud, but not *on* the cloud itself.

4.2.3 Analysis for MG 4 and Tesla Model 3. Both MG 4 and Tesla Model 3 require the user to register an account at the manufacturer's cloud. We performed a test in which we log into an account, pair an EV, and verify that the commands from the app are being executed by the EV. Afterwards, we successfully connected to the same account using the "lost password" function on a different smartphone. Upon accessing our account, we were able to send commands to the EV that were executed by it. Because we did not provide any secret (e.g. a password) that could have been used to decrypt any encrypted data stored on the cloud, this implies that all data needed to send commands to the paired EV must be stored on the cloud in a readable format. This also implies that the cloud itself is able to send commands to the EV.

4.3 Problem analysis summary

An attacker who manages to gain control over the electrical power consumption of all DER by a single relevant manufacturer can perform LAA on the power grid and be able to push it into a load-shedding situation. All analyzed EVs route commands through manufacturer clouds without protection from malicious behavior of the cloud itself. Consequently, the cloud infrastructure could send control commands, such as a start-/stop charging command, to any or all EVs of its fleet. This architecture poses a risk to the stability of the power grid if an attacker manages to gain control over a manufacturer's cloud.

5 Analysis of current implementation

Our goal is to design a countermeasure with minimal impact on user experience and without fundamental changes to the current architecture. Thus, we first analyze the current implementation and user experience for the four EVs.

For all manufacturers, a user must register an account with the manufacturer and then link an EV to this account. All manufacturers allow for only one main user with higher privileges; this will usually be the owner of the vehicle. Tesla Model 3 and VW ID. Buzz require the main user to provide legal documents (ID card, vehicle registration certificate) to the manufacturer to prove that they are the owner. Kia EV6 and MG 4 grant main user access to the first user that connects initially or after resetting the infotainment system. MG 4, Tesla Model 3 and VW ID. Buzz allow for additional, lower privileged non-main users.

[3]https://www.mitmproxy.org/

We find that all the EVs analyzed allow the user to control a number of functionalities in the car, such as air conditioning, unlocking the car, polling location information and others. Two features stand out and require special attention:

Start- / Stop charging: This allows the user to start or stop the charging process of the EV if it is connected to a charger. Users can either immediately start and stop the charging process, schedule future charging sessions, and update the State of Charge (SoC) limits. From a power grid stability point of view, this is the critical feature that is the primary focus of this work.

Using the App as a key: This feature allows the user to configure their phone to be used as a vehicle key via Bluetooth, allowing them to use the vehicle on the road. Users activating this feature do not need to carry a physical key and may thus end up relying on the feature. For the EVs analyzed in this work, this feature is available only for the MG 4 and the Tesla Model 3.

We find that four underlying processes are implemented by the EV manufacturers. For each of these processes, we have analyzed the steps needed for each vehicle. We found that all manufacturers implement similar processes that we describe in the following.

5.0.1 Pair a new app. For all vehicles, the user must register an account with the manufacturer cloud service, verifying either their phone number (Kia EV6) or their email address (MG 4, Tesla Model 3, VW ID. Buzz) in the process. Afterwards, the user must prove that they are authorized to pair their manufacturer cloud account with their EV following one of two approaches:

Local pairing is implemented by Kia EV6, MG 4 and VW ID. Buzz and requires the new user to authenticate to the EV, e.g., by presenting a physical key. The user must then scan a QR Code on the vehicle display (MG 4 and VW ID. Buzz) or enter a 6-digit code displayed on the screen (Kia EV6).

App-based delegated pairing (only for MG 4 and Tesla Model 3) requires an already paired main user to generate an invitation code in the manufacturer app and share it with the new user. The new user must enter the code into the manufacturer app to pair their app with the EV. In practice, the invitation code is implemented as a mobile deep link, the new user thus only has to click it.

5.0.2 Send command. Sending a command to the EV is identical for all vehicles, the user must simply click on a button in the app. Physical proximity is only required to remotely control the car or use the app as a key (both only for MG 4 and Tesla Model 3).

5.0.3 Revoke access. All vehicles allow for only one main user. Revoking access for this user requires either resetting the connect module (Kia EV6) or the infotainment (VW ID. Buzz) or simply connecting a new main user (Kia EV6, MG 4, Tesla Model 3, VW ID. Buzz). This process requires physical access for all vehicles except for Tesla Model 3. The access for a non-main user can be remotely revoked by the main user via the app.

5.0.4 Recover access. All vehicles allow the user to follow a very simple process where they only need to install the respective app on any phone and enter their username and password. Optionally, the user can use the "lost password" function to recover access to their account. Access recovery does not require physical proximity for any of the EVs analyzed.

6 End-to-End Integrity Protection

Our goal is to design a security solution that can be easily implemented. We thus use the current functionalities and the current user journey as a reference for the countermeasures we design. To counteract the conceptual security weaknesses of current solutions, as described in Section 4.2.1, we propose the use of end-to-end integrity protection. Similarly to how end-to-end encryption protects the confidentiality of messages on the way from sender to recipient, end-to-end integrity protection uses digital signatures to protect the integrity and authenticity of messages on the way from sender to recipient. Every command sent from the user's smartphone to an EV must be signed using the user's secret key sk_u. Similarly, every message sent from the EV to the user's smartphone must be signed using the EV's secret key sk_{ev}. Every message must be verified using the sender's public key. To prevent the manufacturer's cloud from storing legitimate commands and bulk-sending them at the desired time, each message must be accompanied by a timestamp, and the receiver must verify that the command was sent within a specified time frame, e.g. the last 30 seconds. In addition, the receiver must verify that the sender is authorized to give the command. To do so, all participants must keep a list of known public keys and store the permissions associated with each. Doing so ensures that a command has not been modified in transit (integrity), it initially originates from the sender as claimed (authenticity), and the sender is authorized, so send the specific command (authorization). If and only if the message can be verified, will the receiver act upon it (e.g. the EV executes the command).

Using this solution implies that the user's smartphone and the EV must know the public keys of the respective other. The initial pairing process must therefore be adapted to ensure this.

6.1 Examples for practical implementation

In this section, we describe how the four critical processes identified in Section 5 can be implemented in our solution.

6.1.1 Pair a new app. We present two different approaches to designing the pairing process for a new app. The first is based on app-based delegated pairing from a main user through the app. This process does not require physical proximity to the EV; it is visualized in Figure 1.

(1) The new user registers an account with the manufacturer's cloud and logs into the account on his to-be-paired app.
(2) An already authorized user (e.g. the main user) instructs their app to generate a random secret s. Optionally, the user defines the authorization limits that should be applied to the new user. The user then sends the random secret s together with the EV's public key pk_{ev} to the new user using an out-of-band channel (e.g., via email or instant messenger).
(3) The already authorized app encrypts the secret s using the EV's public key pk_{ev} and declares the authorization limits that should be applied and sends both to the EV using the *send command* process, signaling to the EV that any new user who knows the secret s should be added to the authorized users list given the limits.
(4) The to-be-paired app of the new user generates an asymmetric key pair (sk_{nu} and pk_{nu}) and stores them.

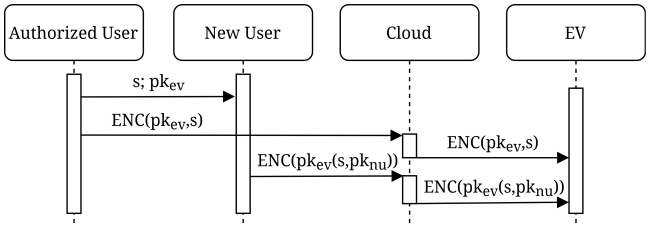


Figure 1: Pairing a new app with app-based delegated pairing from another user's app

(5) The to-be-paired app receives the secret s and the EV's public key pk_{ev}.
(6) The to-be-paired app encrypts the secret s and its own public key pk_{nu} using the EV's public key pk_{ev} and a non-malleable encryption scheme. The generated ciphertext c is then sent to the EV via the manufacturer's cloud.
(7) The EV decrypts the received ciphertext c and verifies that the secret s matches the one it received earlier from the already authorized app.
(8) If the secret s matches, the EV adds the public key of the new user pk_{nu} to the list of authorized users, thus granting the owner of the secret key sk_{nu} permission to send commands to the EV.

The second approach is based on physical authentication to the EV. This approach does require physical proximity to the EV it is visualized in Figure 2.

(1) The user registers an account with the manufacturer's cloud service and logs into the account on their to-be-paired app.
(2) The user authenticates to the EV using a physical key.
(3) The EV generates a random secret s.
(4) The EV generates a QR code containing the secret s as well as the EV's public key pk_{ev} and displays it on its screen[4].
(5) The user scans the QR code using the EV's app, thus reading in the secret s and the EV's public key pk_{ev}. The EV's public key pk_{ev} is stored in the app.
(6) The to-be-paired app of the new user generates an asymmetric key pair (sk_{nu} and pk_{nu}) and stores them.
(7) The app generates a Message Authentication Code (MAC) over the user's public key pk_{nu} using s as the secret.
(8) The app sends both MAC and the user's public key pk_{nu} to the manufacturer's cloud.
(9) The manufacturer's cloud forwards both to the EV.
(10) The EV ensures that the pk_{nu} has not been modified in transit by verifying the MAC.
(11) If the MAC can be verified, the EV adds the public key pk_{nu} to its authorized users list, thus granting the owner of the secret key sk_{nu} permission to send commands to the EV.
(12) (optional) the EV limits the permissions of the new user until the main user has approved or a timeout of 7 days has passed.

Depending on the manufacturer, one or both of these options can be implemented.

[4]QR codes can hold up to 2.956 Bytes, whereas an RSA key is typically 3072 bits, which in practice results in around 768 Bytes of data. The length of the random secret depends on the MAC scheme, 256 Bit (32 Bytes) should be sufficient for most schemes.

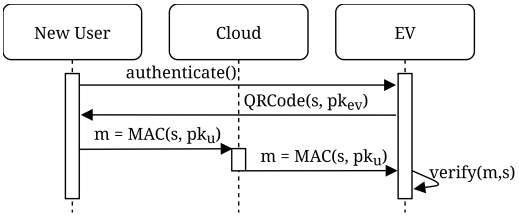


Figure 2: Pairing a new app with local authorization.

6.1.2 Send command.

(1) The user initiates sending a command by clicking on the respective button in the app.
(2) The app generates a message m containing the command and the current timestamp.
(3) The app generates a digital signature s of the message m using its secret key sk_u. $s = Sign_{sk_u}(m)$
(4) The app sends the message m and the digital signature s to the manufacturer's cloud.
(5) The manufacturer's cloud forwards the message m and the signature s to the EV.
(6) The EV verifies the digital signature s using the user's public key pk_u, thus ensuring that
   (a) the message was originally sent by the owner of sk_u, thus authenticating the user, and
   (b) the message has not been modified in transit.
(7) The EV verifies that the timestamp contained in the message m is not older than 30 seconds.
(8) The EV verifies that the user's public key pk_u is marked authorized for the command in the authorized users list.
(9) The EV executes the command if all these checks were successful.

6.1.3 Revoke access. Revocation can be implemented in two ways:

(1) The user α clicks on the respective button in the app to revoke access for user γ.
(2) The app uses the *Send command* process to send the revocation command to the EV.
(3) The EV verifies that the user α is authorized to revoke access for the user γ.
(4) The EV removes the user γ's public key pk_γ from the authorized users list.

or, alternatively:

(1) The user α authenticates to the EV, e.g., by presenting a physical key.
(2) The user α uses the infotainment system to revoke access to the user γ.
(3) The EV verifies that the user α is authorized to revoke the access for the user γ.
(4) The EV removes the user γ's public key pk_γ from the authorized users list.
(5) (optional) the user α resets the infotainment system, thus revoking their own access.

Depending on the manufacturer, one or both of these options can be implemented.

6.1.4 Recover access. If a physical key is present or the main user is available, the *Pair a new app* process can be used to pair a new app with the EV, without the need for the *recover access* process. Otherwise, the following process can be used that uses the manufacturer cloud as a semi-trusted third party:

(1) (Optional) the user uses the *lost password* functionality to regain access to their account on the manufacturer's cloud.
(2) The user logs into their account on the manufacturer's cloud using the manufacturer app.
(3) The user uses the app to initiate the access recovery process.
(4) The app sends a message to the manufacturer's cloud and requests access recovery for the EV.
(5) The cloud service sends a message to the EV that signals that a user is authenticated to the cloud and authorized to use the vehicle and wishes to recover access.
(6) The EV flashes its indicators to signal to the user that a physical-presence check is required.
(7) The user pulls the outside door handle of the driver's door to confirm their physical presence. This has to be performed within 30 seconds, otherwise the recover process is terminated and must be re-initiated.
(8) The EV unlocks the doors and authorizes the user to start the *pairing a new app* process based on physical authentication.

7 Evaluation

7.1 Security Comparison

This work focuses on the specific security risk of a manipulated manufacturer cloud and the danger that this poses to the power grid. This risk is based on the fact that in the currently used concept, a cloud-based MITM attacker can manipulate or inject malicious control commands and especially send *start charging* commands simultaneously to a large number of EVs. Using the scheme that we have developed, only commands sent a legitimate user's smartphone app will be executed by the EV. The manufacturer's cloud only forwards the commands without being able to manipulate them. Due to the fact that LAA depend on *simultaneous* attacks, simply not forwarding commands does not control enough power to impact the stability of the power grid.

In the end-to-end integrity protection scheme, only the *recover access* process depends on the manufacturer's cloud. This sets a successful MITM attacker in a position to initiate the process for any or all EVs simultaneously. This risk is mitigated by a physical presence check, where the user must pull the door handle of the driver's door to confirm that they are physically present and wish to perform the process. Even if an attacker manages to trick a small number of users into performing this step at the right time, the *recover process* only initiates the *Pair a new app* process that will pair the EV with a *locally present* smartphone app.

7.2 Usability Comparison

To evaluate the impact on the user of our proposed end-to-end integrity protection scheme, we analyzed the steps a user must take for each of the processes described in Section 5 compared to the steps in the end-to-end integrity protection scheme as described in Section 6.1. Only two processes differ; these are described here. A complete comparison of all processes is shown in Appendix A.

Pair a new app using physical authentication for the Kia EV6 requires the user to copy a 6-digit code from the infotainment system into their app. The process in the end-to-end integrity protection scheme requires more data (see Section 6.1) to be transmitted from the EV to the app. Thus, scanning a QR code is required.

Recover Access in cases where no physical key is present and no other user can delegate access, in the end-to-end integrity protection scheme requires the user to perform two additional steps after recovering access to the manufacturer cloud account: They must initiate the access recovery process, wait for the indicators of the EV to flash, and then pull the handle of the driver's door. This also implies that the *recover access* process in the end-to-end integrity protection scheme now requires physical proximity to the EV, which the current process does not. This may be an inconvenience, but on the other hand, this is a process that users rarely need to follow. In the worst case, where the user needs to recover access to be able to use their phone as a key to drive the EV, they are present at the EV anyway.

The daily used *send command* process does not change for the user at all.

8 Discussion

This work focuses on the narrow attack scenario of a MITM attacker in an EV manufacturer's cloud. Such an attack might seem unlikely because one might assume that EV manufacturers use sufficient protection mechanisms to prevent it. However, given the potential impact, manufacturer's clouds are a high-profile target even for state sponsored attackers. The clouds of EV manufacturers have shown vulnerabilities in the past; for example, parts of the Volkswagen cloud were left vulnerable in 2024, exposing location data for 800, 000 cars[5]. In Section 4.1 we have shown, based on previous research, that there are EV manufacturers with a large enough market share that their clouds might pose a single-point-of-failure for the power grid.

We have based our scheme on the concept(s) currently used and only added the end-to-end integrity protection required actions. Our results are thus limited to addressing only the specific problem; other conceptual security issues have not been addressed. Because manufacturers are usually able to ship software updates to the EVs, they are from a technical point of view in a position to ship malicious software updates. It can be argued that a successful attacker in a manufacturer's cloud might be able to manipulate software updates to the manufacturer's app on the user's smartphone or the EV's software. Although it is true that attacks on software development exist, established countermeasures such as code-review, regression testing, protected build pipelines, and offline-signing of software releases can help mitigate the threat.

According to our analysis, EV manufacturers have not yet implemented fundamental security mechanisms against cloud attackers. Thus, it is our impression that additional regulation might be required to ensure that EV manufacturers implement a conceptually secure countermeasure such as the one described in this work.

The research presented was carried out on EVs but there are other DERs and IoT devices that suffer from a similar threat. We have chosen to focus on EVs as we expect large numbers of them from comparably few manufacturers. In addition, there are edge cases, such as using the smartphone as a key, which results in higher complexity for the recovery process.

We have found Information on Tesla moving to a public-key scheme for third party applications[6]. We have found no indication that this might be extended to the smartphone app, nor could we verify that it meets the requirements identified in Section 5.

9 Conclusion and Future Work

Previous research has shown that miscontrolling large numbers of IoT devices can have a colossal impact on grid stability [7]. In this work, we have analyzed current concepts for communicating control commands from user's smartphone apps to the EV for four different EVs. Answering our first research question, we found that in all EVs analyzed, we found that data is secured in transit using TLS but that no additional security mechanisms are employed that provide resilience in case of a successful attack on the manufacturer's cloud. This implies that the manufacturer cloud is capable of manipulating or injecting control commands such as a *start charging* command for large numbers of EVs. Depending on the amount of EVs from the specific manufacturer, this could make a manufacturer's cloud a single point of failure to the power grid.

Based on this finding, we have developed an end-to-end integrity protection scheme that extends the currently used concepts and uses digital signatures to ensure that the manufacturer's app and the EV can communicate using an end-to-end integrity protected channel. We have analyzed the critical processes in the currently used schemes as well as for the end-to-end integrity protection scheme and compared them from a user's perspective. Our analysis shows that users do not need to change their behavior for the daily used process of sending commands to the EV. The impact on the app pairing process as well as the access recovery process are minimal. Answering our second research question, we show that it is possible to improve security against cloud-based MITM attackers with only minimal impact on user experience.

In the future, this research could be extended by analyzing other countermeasures such as using rate-limited Hardware Security Modules (HSMs) in the manufacturer's cloud to prevent attackers from sending too many control commands simultaneously. The proposed countermeasure should undergo formal verification before being implemented. The results of this research could also be applied to different kinds of DER or IoT devices, such as PV-I, Building Heat Pump (BHP) or BESS.

Acknowledgments

This work was partly funded by the Topic Engineering Secure Systems of the Helmholtz Association (HGF) and supported by KASTEL Security Research Labs, Karlsruhe.

References

[1] Ahmadreza Abazari, Mohammad Mahdi Soleymani, Saba Marandi, Mohsen Ghafouri, Danial Jafarigiv, Ribal Atallah, and Chadi Assi. 2024. Electric Vehicle-Based Load-Altering Attacks and Their Impacts on Power Grids Operations. *IEEE Reliability Magazine* 1, 4 (Dec. 2024), 79–89. doi:10.1109/MRL.2024.3451429

[5] See https://www.spiegel.de/netzwelt/web/volkswagen-konzern-datenleck-wir-wissen-wo-dein-auto-steht-a-e12d33d0-97bc-493c-96d1-aa5892861027 (in German)

[6] See https://developer.tesla.com/docs/fleet-api/virtual-keys/developer-guide

[2] Samrat Acharya, Yury Dvorkin, and Ramesh Karri. 2020. Public Plug-in Electric Vehicles + Grid Data: Is a New Cyberattack Vector Viable? *IEEE Transactions on Smart Grid* 11, 6 (Nov. 2020), 5099–5113. doi:10.1109/TSG.2020.2994177

[3] Reinhard Behrens and Ali Ahmed. 2017. Internet of Things: An end-to-end security layer. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*. IEEE, Paris, France, 146–149. doi:10.1109/ICIN.2017.7899405

[4] The European Commission. 2017. establishing a network code on electricity emergency and restoration. https://eur-lex.europa.eu/eli/reg/2017/2196/oj

[5] Adrian Dabrowski, Johanna Ullrich, and Edgar R. Weippl. 2017. Grid Shock: Coordinated Load-Changing Attacks on Power Grids: The Non-Smart Power Grid is Vulnerable to Cyber Attacks as Well. In *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, Orlando FL USA, 303–314. doi:10.1145/3134600.3134639

[6] D. Dolev and A. Yao. 1983. On the security of public key protocols. *IEEE Transactions on Information Theory* 29, 2 (March 1983), 198–208. doi:10.1109/TIT.1983.1056650

[7] Niklas Goerke, Alexandra Märtz, and Ingmar Baumgart. 2024. Who Controls Your Power Grid? On the Impact of Misdirected Distributed Energy Resources on Grid Stability. In *The 15th ACM International Conference on Future and Sustainable Energy Systems*. ACM, Singapore Singapore, 46–54. doi:10.1145/3632775.3661943

[8] Dustin Kern and Christoph Krauß. 2021. Analysis of E-Mobility-based Threats to Power Grid Resilience. In *Computer Science in Cars Symposium*. ACM, Ingolstadt Germany, 1–12. doi:10.1145/3488904.3493385

[9] Sajjad Maleki, Shijie Pan, Subhash Lakshminarayana, and Charalambos Konstantinou. 2025. Survey of Load-Altering Attacks Against Power Grids: Attack Impact, Detection, and Mitigation. *IEEE Open Access Journal of Power and Energy* 12 (2025), 220–234. doi:10.1109/OAJPE.2025.3562052

[10] Bidyut Mukherjee, Songjie Wang, Wenyi Lu, Roshan Lal Neupane, Daniel Dunn, Yijie Ren, Qi Su, and Prasad Calyam. 2018. Flexible IoT security middleware for end-to-end cloud–fog communication. *Future Generation Computer Systems* 87 (Oct. 2018), 688–703. doi:10.1016/j.future.2017.12.031

[11] Shahid Raza, Tómas Helgason, Panos Papadimitratos, and Thiemo Voigt. 2017. SecureSense: End-to-end secure communication architecture for the cloud-connected Internet of Things. *Future Generation Computer Systems* 77 (Dec. 2017), 40–51. doi:10.1016/j.future.2017.06.008

[12] Mohammad Ali Sayed, Ribal Atallah, Chadi Assi, and Mourad Debbabi. 2022. Electric vehicle attack impact on power grid operation. *International Journal of Electrical Power & Energy Systems* 137 (May 2022), 107784. doi:10.1016/j.ijepes.2021.107784

[13] Mohammad Ali Sayed, Mohsen Ghafouri, Ribal Atallah, Mourad Debbabi, and Chadi Assi. 2023. Protecting the future grid: An electric vehicle robust mitigation scheme against load altering attacks on power grids. *Applied Energy* 350 (Nov. 2023), 121769. doi:10.1016/j.apenergy.2023.121769

[14] Saleh Soltan, Prateek Mittal, and H. Vincent Poor. 2018. BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 15–32. https://www.usenix.org/conference/usenixsecurity18/presentation/soltan

A Usability Comparison of all Processes

	Current Process	E2E Integrity protected process	Difference
Pair a new app using physical authentication	The new user must: (1) Register account with manufacturer cloud service (2) Authenticate to EV, e.g. by presenting a physical key (3) Follow process on infotainment screen, including (4) scanning a QR Code or entering a 6-digit code into the app	The new user must: (1) Register account with manufacturer cloud service (2) Authenticate to EV, e.g. by presenting a physical key (3) Follow process on infotainment screen, including (4) scanning a QR Code	The process on the infotainment screen requires scanning a QR code. Entering a 6-digit code is obsolete.
Physical proximity	Required	Required	
Pair a new app by delegation from main user	(1) New user: Register account with manufacturer cloud service (2) Existing main user: generate an invite code[2] in the manufacturer app (3) Existing main user: send code to new user using a secure channel (4) New user: input data into manufacturer app	(1) New user: Register account with manufacturer cloud service (2) Existing main user: generate an invite code[2] in the manufacturer app (3) Existing main user: send code to new user using a secure channel (4) New user: input data into manufacturer app	None
Physical proximity	Not required	Not required	
Send command	(1) Click on respective button in the app.	(1) Click on respective button in the app.	None
Physical proximity	Not required[1]	Not required[1]	
Revoke Access resetting part of the EV	The user must: (1) authenticate to the EV, e.g. by presenting a physical key (2) follow an on-screen procedure to reset parts of the vehicle	The user must: (1) authenticate to the EV, e.g. by presenting a physical key (2) follow an on-screen procedure to reset parts of the vehicle	None
Physical proximity	Required	Required	
Revoke Access by connecting a new main user	The new main user must: (1) follow the respective process for connecting a new main user	The new main user must: (1) follow the respective process for connecting a new main user	None
Physical proximity	Depending on sub-process	Depending on sub-process	
Revoke Access of a lower privileged user	The main user must: (1) click on respective button in the app	The main user must: (1) click on respective button in the app	None
Physical proximity	Not required	Not required	
Recover Access	The user must: (1) (optional) reset the password of their manufacturer cloud account using the "lost password" function (2) (optional) install the manufacturer's app on a new device (3) login to their manufacturer cloud account using the manufacturer app on any device	The user must: (1) (optional) reset the password of their manufacturer cloud account using the "lost password" function (2) login to their manufacturer cloud account from any device (3) initiate the access recovery process (4) wait for the indicator of the EV to flash, then pull the outside handle of the driver's door	In the end-to-end integrity protection process, the user must perform two more trivial steps. The user must be physically present at the EV.
Physical proximity	Not required	Required	

1: Physical Proximity might be required for special operations such as remote controlling the car.

2: Using mobile deep links for this process enables the mobile operating system to open specific links with specified apps.

Table 1: Comparison of the steps a user must take in the currently used communication concept vs. the end-to-end integrity protection improved concept.