



INFORMATION AND COMMUNICATION TECHNOLOGIES COLLABORATIVE PROJECT

“V-Charge”

Autonomous Valet Parking and Charging for e-Mobility

Grant Agreement Number 269916



V-CHARGE

Deliverable Number	5.4
Deliverable Name	Final integration and evaluation
WP Number	5
Date of Preparation	01.07.2015
Date of Submission	31.07.2015
Editor/Beneficiary	Timpner, Rottmann, Wolf / TU BS

Executive summary

In WP5 of the V-Charge project, the binding system and a user interface have been developed. This includes the network infrastructure, server hardware as well as software services. Wireless radio interfaces are needed to communicate with the Electric Vehicles (EVs) in the parking lot. This document describes the final architecture and the evaluation of the V-Charge Server, its setup, components, and interactions.

First, we describe the final version of user interfaces in Section 1, including screenshots, an explanation of what user experiences were taken into account for the final redesign, as well as an additional design study.

In Section 2, we provide the final description of the server, communication protocols, and user interaction.

Sections 3 and 4 present the second versions of the parking spot and charging station management algorithms, respectively. Detailed evaluations are included in each section.

The final security system as well as its evaluation according to Milestone 6 (M6) is presented in Section 5.

Finally, Section 6 describes the final integration and evaluation of the binding system. To this end, the test procedure is specified first. Then, the system is validated in simulation and emulation as well as at the final test site at Mobile Life Campus in Wolfsburg, Germany.

Contents

1	User Interfaces	5
2	Communication Protocols	10
2.1	Handling Communication Failures	12
3	High-Density Parking	13
3.1	Introduction	13
3.1.1	Related Work	13
3.1.2	Our Contribution	14
3.2	k -Stacks	14
3.2.1	Operating k -Stacks	15
3.2.2	Analysis	16
3.3	Experimental Evaluation	17
3.3.1	Test Cases	17
3.3.2	Simulator	18
3.3.3	Parking Lot Layout	18
3.3.4	Results	19
3.4	Conclusion	22
4	Charging Station Scheduling	23
4.1	Introduction	23
4.2	Related Work	23
4.3	Charging Station Scheduling	24
4.3.1	Static Scheduling	25
4.3.2	Dynamic Scheduling	27
4.4	Simulation Setup & Scenarios	28
4.4.1	Simulation Setup	28
4.4.2	Simulation Parameters	29

4.5	Evaluation	33
4.5.1	Metrics	33
4.5.2	Scenario 1: Airport	34
4.5.3	Scenario 2: Downtown	35
4.6	Conclusion	38
5	Security Challenges and Solutions	39
5.1	Related Work	40
5.1.1	Web Security	40
5.1.2	Delay-Tolerant Networking	41
5.1.3	Key Deployment in Vehicular Networks	41
5.2	Web Service Security	42
5.2.1	Authentication and Session Management	43
5.2.2	Transport Layer Security	43
5.2.3	Password Protection	44
5.3	Smartphone-to-Cloud Security	45
5.3.1	OAuth Specification	46
5.3.2	OAuth Server	48
5.3.3	OAuth Client	51
5.3.4	Transport Layer Security for Client Applications	53
5.4	Vehicle-to-Cloud Security	53
5.4.1	Security in Delay-Tolerant Networks	54
5.4.2	Certificate Revocation	54
5.4.3	Implementation	54
5.5	Vehicle Registration and Key Deployment	55
5.5.1	Proposed Registration and Key Deployment Process	55
5.5.2	Implementation	58
5.6	Discussion	58
5.6.1	In-Vehicle Key Storage	59

5.6.2	Attack Scenarios	59
5.6.3	Evaluation	61
6	Final Integration and Evaluation	62
6.1	Experiment 1: Parking Management	62
6.2	Experiment 2: Security Overhead	62

1 User Interfaces

This section describes the second version of user interface devices. The interface design of the mobile Android devices has undergone a significant redesign to resemble experiences gathered by regular users, especially developers, as described in Deliverable 5.3 (D5.3).

In particular, these were:

- Provide “one-glance” visual feedback as to in which phase of the valet parking process the car is (pick-up, drop-off, none).
- Provide visual feedback in case something went wrong.
- Simplify the user interface such that actions do not need to be confirmed if the action is not safety-relevant.
- Some users suggested moving the choice of vehicle option (relevant for people regularly using more than one car / car sharing companies) into persistent “background” options (or “power-user mode”) so that a regular user does not need to repeat this step with every drop-off / pick-up.

The second version of the Android user interface reflects this feedback, as depicted by Figures 1 and 2.

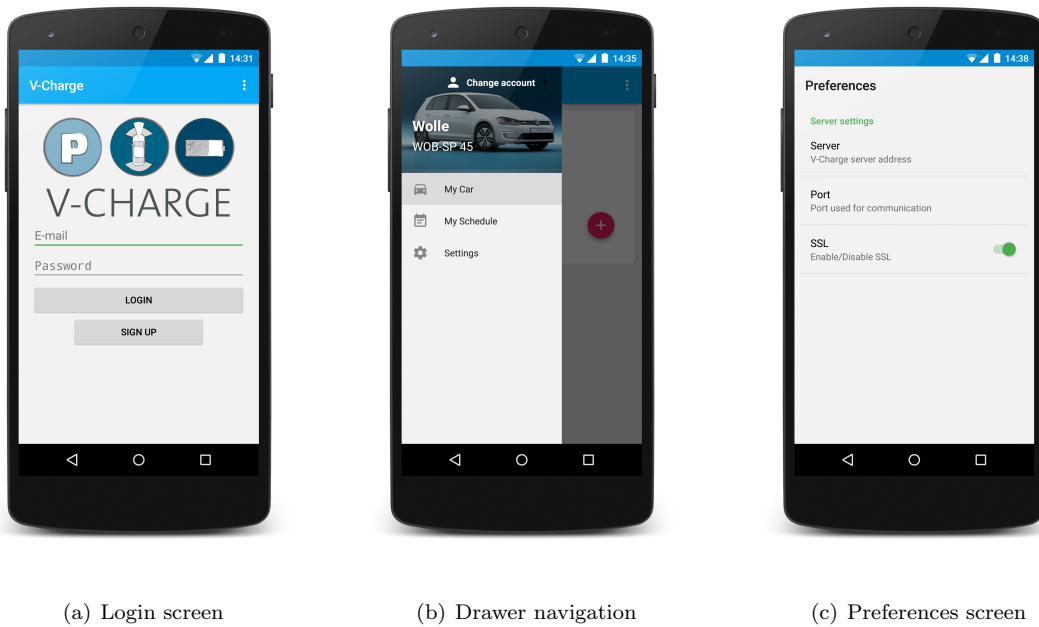
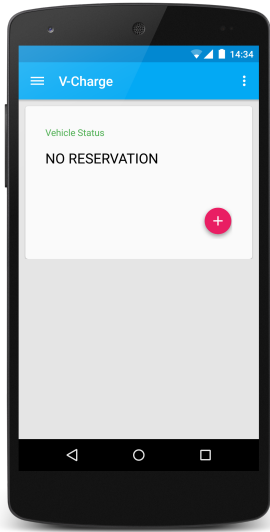
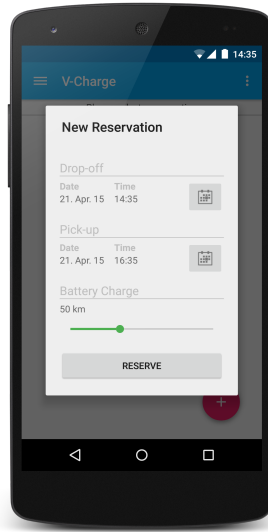


Figure 1: Android user interface

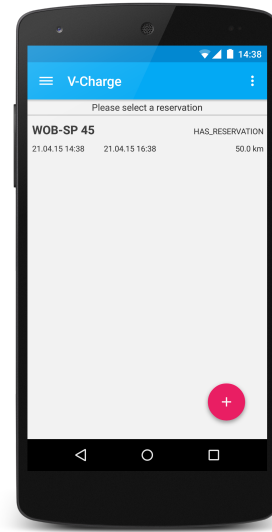
An additional evaluation of the user interface as requested in M6 will not be conducted, as explained in D5.3. In addition to the redesign, though, an independent design study at HSR Hochschule für Technik Rapperswil has been supervised. An excerpt of the results of different development stages, which could become relevant for future commercial developments, are presented in Figures 3 through 5.



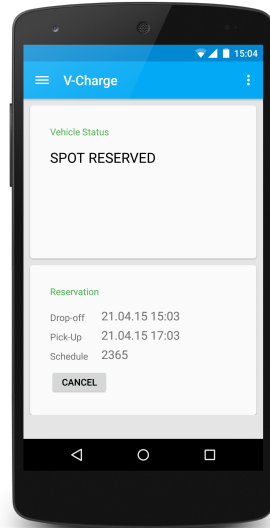
(a) Status screen



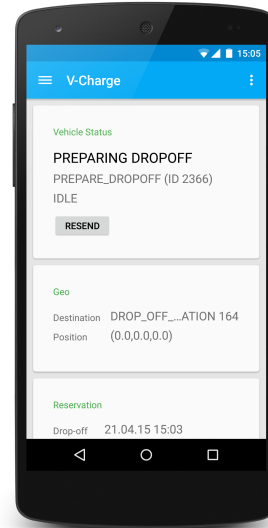
(b) New reservation



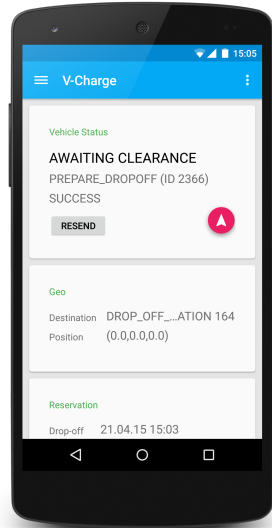
(c) Reservation screen



(d) Spot reserved

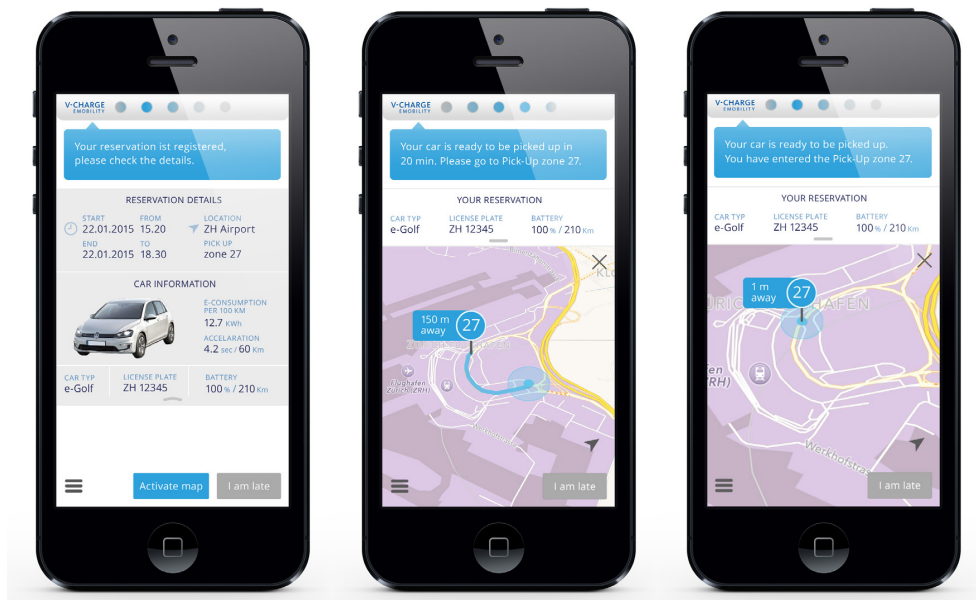


(e) Preparing dropoff

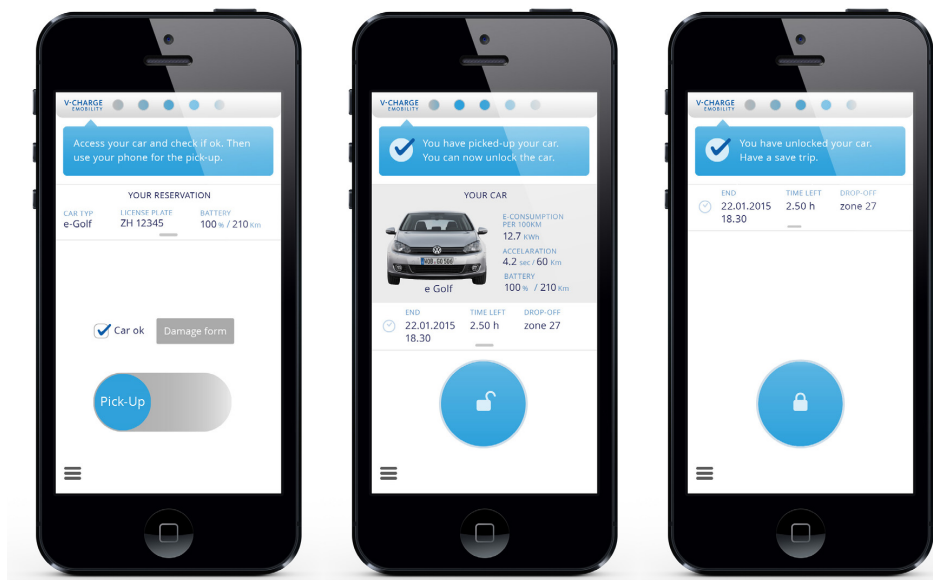


(f) Awaiting user clearance

Figure 2: Android user interface



(a)



(b)

Figure 3: Design prototype developed at HSR Rapperswil.

0 Trademark

V-CHARGE
EMOBILITY

colored on white background

V-CHARGE
EMOBILITY

colored on light grey background

V-CHARGE
EMOBILITY

white on dark grey background

Basic Elements

1 Color Swatches

Primary Colors



Lightblue
rgba 18/163/232
Hex #12a3e8



Skyblue
rgba 0/91/170
Hex #005baa



Navyblue
rgba 17/17/104
Hex #111166

Secondary Colors



White
rgba 255/255/255
Hex #ffffff



Light grey
rgba 170/170/170
Hex #aaaaaa



Middle grey
rgba 90/90/90
Hex #5a5a5a

2 Typography

Open Sans Regular

Open Sans Regular
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
WXYZ12346 7890!»\$%&/()
?;:;_-.-><

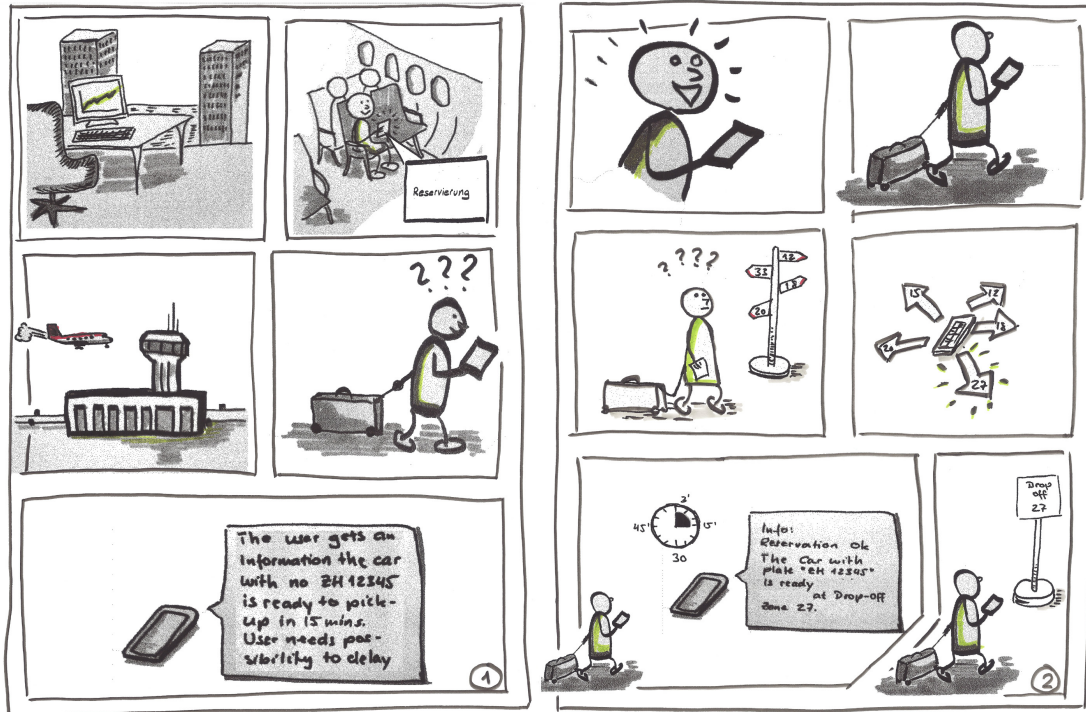
H1 HEADING 12PX

H2 HEADING 10PX

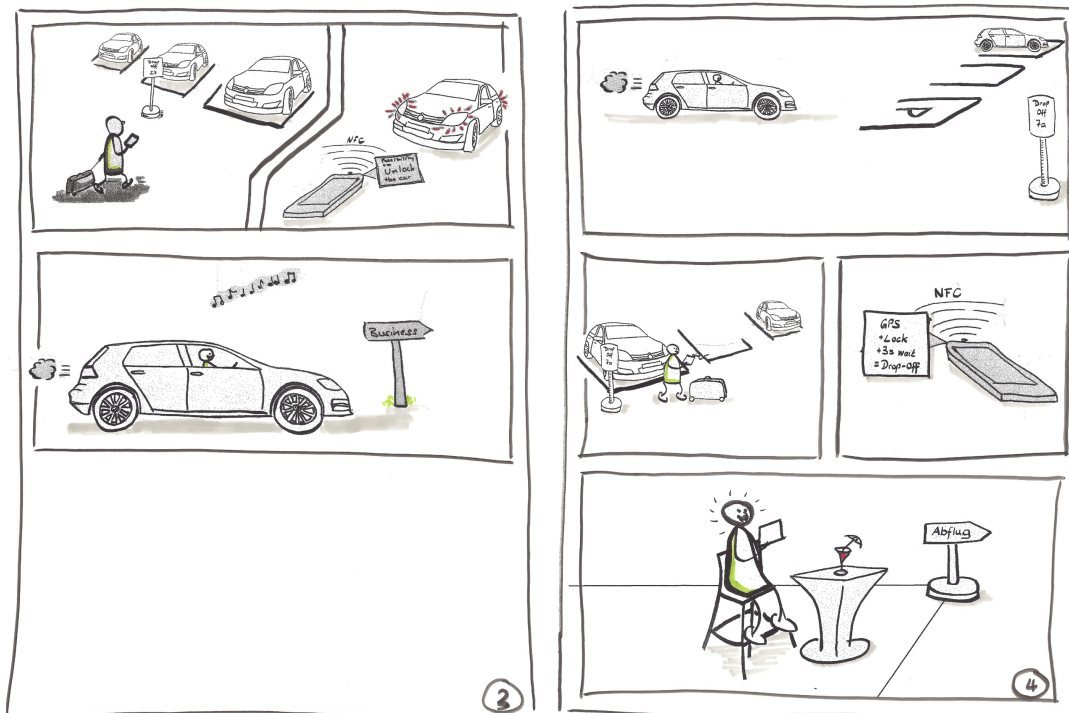
Information 14px

Your car is ready to be picked up.
You have entered the Pick-Up zone 27.

Figure 4: Styleguide developed at HSR Rapperwil.



(a)



(b)

Figure 5: Storyboards developed at HSR Rapperwil.

2 Communication Protocols

Figure 6 gives a high-level overview of the general communication components.

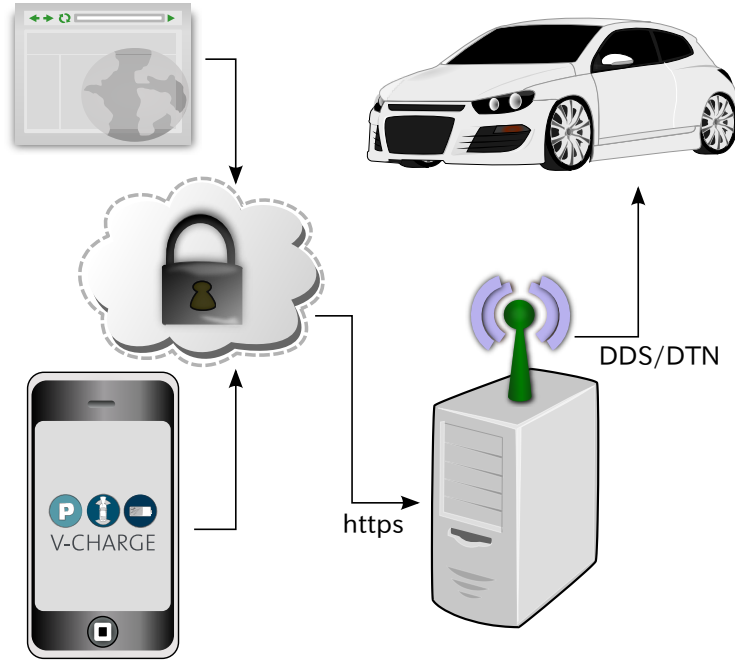


Figure 6: High-level overview of communication components.

The software architecture of the V-Charge backend service ParkingManager (PM) has been described in Deliverable 5.2 (D5.2). Since the basic architecture has basically stayed stable since then we refer the reader to D5.2 for a detailed description instead of repeating it here. Figure 7 shows a more detailed view of the V2X communications architecture. DDS Interface Definition Languages (IDLs) have been described in D5.3.

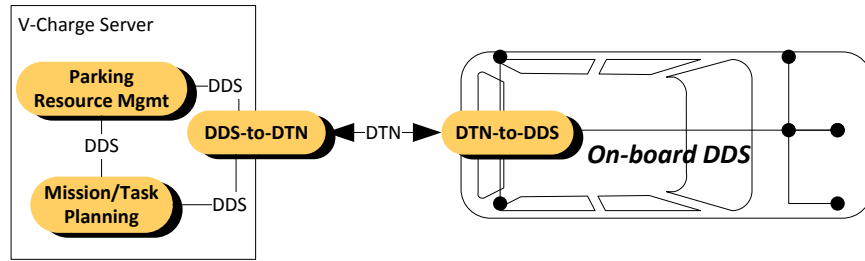


Figure 7: V2X architecture.

Significant updates have been made to the Vehicle Interaction Concept (see Section 2.5 in D5.2), though. The interactions of the user with his vehicle and of the vehicle with the server are of utmost importance for final usability, safety, and user acceptance. The state machine in Figure 8 depicts the final version of this interaction process and the resulting communication protocol.

2.1 Handling Communication Failures

The server only transmits high-level mission goals to the vehicle. As long as no unexpected incidents occur, the vehicle only sends status messages to the server, containing the current position and mission goal ID. As soon as the state changes (destination is reached, a road is blocked, ...), a status message signaling the state (FAILURE, SUCCESS) is being transmitted. While driving, the communication link between backend system and vehicle is not crucial, but if the car gets stuck, the server would not be able to handle the problem.

The outage of the link (missing Heartbeat messages from the other partner) would be recognized by both communication partners, which is the vehicle or backend system. After a timeout, the operator could be notified. He might either steer the vehicle manually or use a mobile WiFi station to transmit a new mission goal. Mobile WiFi stations which might even be battery powered and need no additional cables for Ethernet are a cost-effective, simple solution for handling temporary communication problems.

The possibility of communication failures is drastically reduced by using our DDS/DTN bridge. With these features, parked vehicles can be used as a gateway for multi-hop data transmissions of missions and status messages. Depending on the configuration of the network, a vehicle with working cellular connection is able to forward DTN bundles containing mission information via WiFi to the vehicle with the broken connection.

3 High-Density Parking

3.1 Introduction

Automated valet parking offers significant benefits, not only to the traveler, but for operators of parking lots, fleet managers, or car rental companies as well. In contrast to regular (human only) or mixed-mode (human and robot drivers share the same area) parking scenarios, there is a significant potential for space (and, consequently, cost) savings in fully-automated valet parking. This becomes increasingly important as a large percentage of urban space is already exclusively dedicated to parking, for instance 21 % [Fer+14] in midtown Atlanta, Georgia. Generally, such a scenario can be implemented very cost-effectively even in existing parking lots by closing off a dedicated area to human drivers. Since human passengers can exit the vehicle before it enters the parking area, automated cars do not require ambient space for opening the doors, and can thus be parked much closer than in regular scenarios. A coordinated parking management based on Vehicle-to-Infrastructure (V2I) communication, as implemented by V-Charge [Fur+13], offers the possibility to increase the parking density even further by allowing vehicles to actually block other cars. Obviously, this is not an option in regular or mixed-mode scenarios, where drivers cannot manually move blocking vehicles. In the fully-automated scenario, however, this becomes feasible. The goal is to use the parking area as efficiently as possible while maintaining reasonable pick-up times and limiting vehicle movements (and thus energy consumption and attrition).

In this section, which has been published as a separate paper [Tim+15], we propose a novel parking management approach based on k -stacks that guarantees bounds on the maximum movement of vehicles as well as on pick-up times.

3.1.1 Related Work

The efficient management of parking lots has received a great deal of attention since the ever increasing proliferation of the automobile and the resultant shortage of parking space. In this paper, we therefore focus on models and systems for optimizing the utilization of parking space. Young [You88] reviews models that help to determine the efficiency of parking lot layouts in order to support design decisions. Amongst others, models for determining the dynamic capacity, i.e., the actual capacity depending on parking and unparking maneuvers, and the delay for unparking vehicles are presented. However, these numerical models can only analyze simple performance and cannot describe the interactions in more complex scenarios as investigated in this work.

Simulations in order to select the best parking layout or pricing structure have been conducted by several authors [HD97; RWB93; YY97]. While some works [RWB93; YY97] also investigate conflicts between unparking vehicles and the pick-up time, which are metrics we are also interested in, these works do not consider alternative (i.e., other than human-only) parking layouts. Abdullah et al. [Abd+12] reason about the most efficient parking layout at a campus area. They compare perpendicular, parallel, and diagonal parking and conclude that perpendicular parking can accommodate the most vehicles per road segment. Munzir, Ikhsan, and Amin [MIA10] use a linear programming approach to maximize the available on-street parking capacity, considering average parking duration, space requirements and overall traffic. As a main result, it was shown that the constraints need to be prioritized and cannot be satisfied simultaneously. Again, none of the abovementioned works takes unconventional parking layouts such as the proposed k -stacks into account, as they are focusing on regular (i.e., human drivers only) scenarios.

Automated valet parking has seen increasing attention lately, with the majority of approaches being based on self-driving vehicles [Fur+13; Lop+13; MC13; Jee+10]. None of these considers the optimization of parking

space, though. An alternative to automated cars are robotic valets that pick up and move the vehicles to a parking spot. Such systems are already commercially available, for instance at Dusseldorf Airport, Germany.¹ Here as well as in literature [NAP13], vehicles are parked such that they block each other, requiring multiple rearrangements to retrieve a specific car. However, the authors do not provide any results on the overhead (e.g., in terms of the pick-up delay) of this approach. In general, robotic valets require high acquisition and maintenance investments and are less flexible than self-driving cars, which is why we focus on the latter.

Ferreira et al. [Fer+14] explore the impact of autonomous vehicles on parking lots, guided by a parking lot manager. They propose a parking lot design for 100 autonomous cars saving 30 % of space as compared to a classical layout. 20 % of space is saved due to smaller parking spots, the remaining 10 % by exploiting the insight that parked autonomous cars may be blocked, as they can always be moved aside by the parking lot controller. Ferreira et al. do not claim to have an optimal parking strategy, in fact, they focus on the new aspect of having a central parking lot controller at all. Hence, they do not provide detailed (worst case) analysis about how many cars have to be moved or total shunting distance or number of operations to access an arbitrary car, instead they present experiments for 100 cars and average case analyses.

To the best of our knowledge, we present the first study of parking space optimization for automated valet parking with an in-depth theoretical analysis of the parking lot properties under various aspects including worst-case extraction time, total shunting distance, and the number of shunting operations (each per car). Most importantly, the proposed model bounds all these values.

3.1.2 Our Contribution

We generalize a widespread parking lot design using k -stacks, see Section 3.2. k -stacks are capable of increasing the parking lot density in purely automated vehicle scenarios. We achieve this by exploiting the capability of automated vehicles to move aside when they block another car.

The increase in parking density comes at the expense of *shunting operations* of parked cars. We show that our model limits the total number of shunting operations as well as the total distance covered during shunting operations per car by $O(k)$ and $O(k^2)$, respectively. Additionally, we can guarantee limited response times, as no car is ever blocked by more than $k - 1$ others.

We evaluate our model using a simulator in four different scenarios, including a realistic scenario based on real-world data from a downtown parking garage.

3.2 k -Stacks

In this section we introduce k -stacks, the basic building block of our high-density parking lot for autonomous vehicles, see Figure 9. k -stacks generalize a typical parking lot layout where cars are parked in boxes perpendicular to the road, compare Figure 9(a). Assuming the width of the road to be equal to the length of a parking box, this pattern allows a maximum parking density of $\frac{2}{3}$. For the sake of simplicity, we assume the width of a road to be equal to the width of the parking spot; while that is not true in practice, our analysis of parking lot density can easily account for that by multiplying with an appropriate constant.

On a conventional parking lot, the density cannot be increased, because additional cars would block other cars' way out. But when dealing with a parking lot exclusively available to autonomous vehicles as described in Section 3.1, blocking cars can be moved out of the way at any time. Hence, blocking vehicles becomes a viable option. In this paper, we show how to exploit that feature to increase parking lot density.

¹<http://serva-ts.com/>

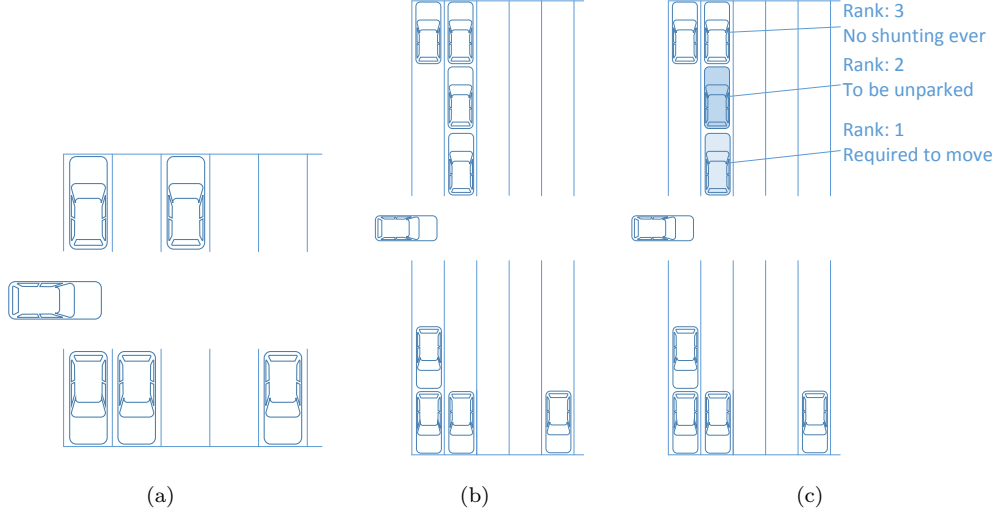


Figure 9: k -stacks generalize the typical pattern of cars parking perpendicular to the road as in (a). We extend the boxes to hold a maximum of k instead of 1 cars, see (b), thus increasing the overall density. Removing the i -th car from a k -stack requires $i - 1$ cars to be moved, compare (c), but increases the ranks of those cars by 1.

However, temporarily moving a car out of the way too often or too far can deplete its battery or gas tank and increases attrition. Another problem is that if a car is hopelessly jammed, it can take a long time to maneuver it out of the parking lot. Our approach avoids either situation. Subsequently, we refer to a parked car temporarily moving to make way for another car as *shunting operation*.

Our objective is to increase the maximum achievable car density of a parking lot while bounding, for each car, (a) the worst-case response (or pick-up) time for maneuvering it out of the parking lot, (b) the number of shunting operations it is involved in, (c) its total distance covered during shunting operations. In Theorems 1, 2, and 3 we show that k -stacks meet the criteria (a), (b), and (c), respectively.

3.2.1 Operating k -Stacks

Instead of placing no more than one car in each parking box, we extend the boxes vertically to hold up to k cars instead, compare Figure 9(b). This way, a parking lot needs less roads for more parking spots and the maximum parking density increases to $\frac{2k}{2k+1}$ (k on either side of the road). Obviously, $k = 1$ yields a classical parking lot layout.

We examine a single k -stack S . Let $1, \dots, k$ denote the parking spots in S , where 1 is the spot just next to the road and k the spot the furthest in the back. We say that a car located on the i -th spot of S has *rank* i and *blocks* all parking spots of S ranked strictly higher than i .

The invariant is that no free spot is blocked: If the i -th spot is occupied the spots $i + 1, \dots, k$ are occupied as well; and if the i -th spot is free the spots $1, \dots, i - 1$ also are.

A car c parking on S undergoes 3 phases, see Figure 9.

1. At first, if there is a free spot in S , it is parked. c directly moves to the free spot with the highest

rank, requiring no shunting operations, because every free spot can be reached directly.

2. In the second phase, c remains parked, but needs to perform shunting operations if cars of higher rank leave S .
3. In the third and last phase, c is removed from S . Suppose c has attained rank i , then removal works as follows: (1) The at most $i - 1$ cars behind c temporarily move on the road. (2) c leaves. (3) All temporarily moved cars enter S .

3.2.2 Analysis

We analyze some properties of cars parked on a single k -stack. The following theorem begins our analysis and states that no car is blocked by more than $k - 1$ other cars. This ensures that the response times of maneuvering a car out of a k -stack are bounded.

Theorem 1. *Accessing a car c of rank i in a k -stack requires no more than $i - 1$ shunting operations. Thus, no more than $k - 1$ shunting operations are required to access an arbitrary car in a k -stack.*

Proof. Refer to Figure 9(c). No more than $i - 1$ cars can block the way of c , as it has rank i . Those of higher rank are not moved. $\square$

The following theorem bounds the number of shunting operations (and thus of energy-intensive engine starts) per car by $k - 1$.

Theorem 2. *Between entering and leaving a k -stack, a car performs no more than $k - 1$ shunting operations.*

Proof. Let c be a car of rank i on the k -stack S . c only performs a shunting operation when another car of higher rank j , with $i < j < k$, leaves S . After that, the evading cars are parked on S again, but due to the new free spot, their ranks increase by 1. Thus, c attains a new rank of $i + 1$. A car of rank k never blocks any other car, so this can happen at most $k - 1$ times, as claimed. $\square$

We want cars in the parking lot to cover as little distance as possible in order to reduce fuel consumption and attrition. A natural baseline for the driven distance is the transit in a conventional parking lot, of course. No matter how long a car stays in a k -stack, the following theorem concludes our theoretical analysis by bounding that distance with $O(k^2)$.

Theorem 3. *Between entering and leaving a k -stack, a car moves no further than $O(k^2)$ during shunting operations.*

Proof. Whenever the car c temporarily makes way for a blocked car, it has to move out of the k -stack and into a temporary queue on the road. Each is possible within the order of two stack lengths, i.e., with a distance of $O(k)$. By Theorem 2 this happens at most $k - 1$ times, so the total distance covered is in $O(k^2)$. $\square$

Note that a parking lot entirely composed of k -stacks can achieve a car density of up to $\frac{2k}{2k+1}$ (assuming a road width equal to the box length for one car): One road in the middle and k -stacks to the left and the right, compare Figure 9(b). Theoretically, for any density $0 < \rho < 1$ we can choose $k = \arg \min_{k \in \mathbb{N}} \frac{2k}{2k+1} \geq \rho$, although the traffic obviously suffocates under its own weight if k is too large. Simulations in Section 3.3.4 indicate $k = 3$ to be a reasonable choice.

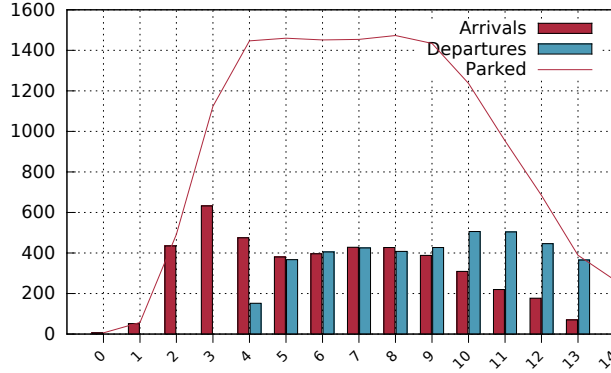


Figure 10: The number of cars arriving/departing as well as parked cars per simulation hour in the real-world test case.

3.3 Experimental Evaluation

We have run extensive simulations to evaluate the performance of the k -stacks approach. In this section, we first define four test cases. We then describe the simulator and the parking lot layout, before presenting our evaluation results.

3.3.1 Test Cases

We present four test cases modeling different arrival and departure processes of cars at a standardized parking lot, which we define in Section 3.3.2. Scenarios 3.3.1 and 3.3.1 are as realistic as possible, while 3.3.1 and 3.3.1 are constructed to be worst- and best-case examples for our approach, respectively.

Real-World For a realistic evaluation, we resort to a dataset from a real-world downtown parking structure in Braunschweig (BS, in the following), Germany, that is both used by shoppers for short-term parking as well as by commuters during their working hours. The parking lot has a total of 1440 spots. Figure 10 shows the number of arrivals and departures for a period of 14 hours. We start with an empty parking lot configuration and simulate it for a period of 14 hours, in which it is completely filled once, see Figure 10.

Random The Random test case (RAND, in the following) is based on probability distributions from the literature; we model the parking duration for a shopping scenario according to Lohse and Schnabel [LS11], using a negative binomial distribution for the probability W_k of the parking duration class k :

$$W_k = \binom{k + p_2 - 2}{k - 1} (p_1)^{k-1} (1 - p_1)^{p_2}, \quad (1)$$

with $(0 < p_1 < 1; p_2 > 0; k = 1, 2, \dots)$. The parameters p_1, p_2 allow to calibrate the distribution with empirical data. Lohse and Schnabel use the opening hours to this end, resulting in the values in Table 1. The underlying arrival process is modeled with a Poisson distribution with $\lambda = 3.5$.

Table 1: Parameters of parking duration distribution for a shopping scenario with opening hours from 9am to 6pm.

Arrival	p_1	p_2	Arrival	p_1	p_2
08–09 am	0.96	3	1–2 pm	0.98	6
09–10 am	0.95	3	2–3 pm	0.89	6
10–11 am	0.93	4	3–4 pm	0.90	5
11–12 am	0.90	6	4–5 pm	0.94	4
12–01 pm	0.87	6	5–6 pm	0.98	4

FIFO In the First In First Out (FIFO) test case, cars are picked up in the order of arrival at their respective k -stack. At first, the parking lot is completely filled with vehicles. Then, all cars with rank $i = k$ are ordered to the pick-up zone simultaneously. After a car has left its stack, the ranks of the remaining cars increase by 1. All cars with the new rank $i = k$ are ordered to the pick-up zone after all previous shunting operations have been completed. This repeats until all vehicles have left the parking lot. For the FIFO test case we expect long pick-up times and large shunting distances due to a maximum of shunting operations.

LIFO The Last In First Out (LIFO) test case represents the opposite of the FIFO case with regard to the order in which cars are picked up. Similarly to FIFO, the parking lot is first filled completely. Then, the car that arrived last (which, naturally, has rank $i = 1$) is ordered to the pick-up zone. When it has left the parking lot, the car that arrived second to last leaves. This process repeats for all cars. Since the LIFO case requires the least shunting operations, we expect very good pick-up times and small shunting distances.

3.3.2 Simulator

We implement a discrete event simulation for a grid-based parking lot layout. Every simulator event, in the following referred to as a *tick*, corresponds to 0.9 s of real time. The grid is made up from $2.5\text{ m} \times 2.5\text{ m}$ *tiles*, on which the cars move with a velocity of 10 km/h or about 2.778 m/s. Each vehicle requires 2 tiles of space (i.e., $5\text{ m} \times 2.5\text{ m}$), which cannot be shared with other vehicles at the same time. Accordingly, each parking spot has the same dimensions. Streets have a width of 1 tile. The simulator sets waypoints for the cars which they follow autonomously, meaning that they determine whether they are allowed to drive onto the next tile (if it is not blocked). After the last car leaves the parking lot the simulation terminates. Our simulator provides a visual mode as shown in Figure 11. Whenever a car parks in or unparks from a k -stack, the stack gets *locked* for other vehicles, whose parking/unparking events will be queued. When a car with rank i unparks, all vehicles with ranks $j < i$ back out to the lowest rank in the stack. Now, the simulator checks if enough road space is available to allow the vehicles to exit their stack and, if so, temporarily blocks the space off both to vehicles trying to unpark from a neighboring stack and to cross traffic. In Figure 11, this is indicated by the red-colored parts of the road. When the shunting operations of this stack have been completed, the road is unblocked again. Arriving vehicles are always parked in the least filled k -stack that is closest to the exit. A video of the RAND simulation is available online.²

3.3.3 Parking Lot Layout

We generalize a typical parking layout of single parking spots (aligned perpendicular to a street as in Figure 9(a)) to k -stacks as shown in Figure 9(b). Moreover, we define our layout to comprise $q = 3$ parallel

²<http://youtu.be/pCzI-18tsPY>

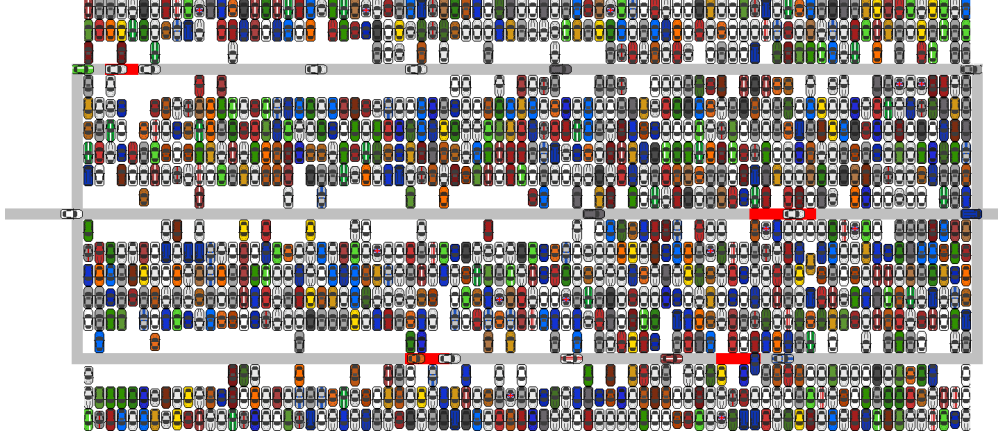


Figure 11: Example of a relatively full parking lot with $k = 3$. Red-colored street sections are blocked for shunting operations.

streets of length l with k -stacks that are connected with both each other and the entry/exit points via cross streets, as shown in Figure 11. In our default layout, there is exactly one entry and one exit to the parking lot. However, in the evaluation we also consider a layout with three exits (one per road) in order to determine (and eliminate) the potential bottleneck of a single exit on the evaluation metrics. We model the total number of parking spots $n = 1440$ after the real-world parking structure described in Section 3.3.1. To provide comparable simulation settings, n, q are invariant for all test cases, k is the investigated parameter. The length l of the parallel streets with k -stacks is thus defined as $l = \frac{n}{2kq}$. The area A of the parking lot (without entry/exit ways; A_p being the area of a single spot) is $A = n \cdot A_p + q(l + 2) + 2(q - 1) \cdot 2k \cdot A_p$.

3.3.4 Results

In this section we present simulation results, verifying the validity of our model and evaluating reasonable parameters.

Exit Time Figure 12 shows the exit time, which is arguably the most important metric for customers as it is the time they have to wait for their car when they order it to the pick-up zone. It can be seen that both the BS and the RAND scenario perform similarly well as the LIFO scenario, which is almost a best case scenario, as its lion's share of exit time is driving time (as opposed to shunting time; see also Section 3.3.4). The average exit time for $k = 1$ in all three scenarios ranges between 1.3 min to 2 min, and decreases for larger k . Eventually it levels off at around 1 min, which is almost exclusively the time required to drive through the parking lot, as described in the following sections. These results show that increasing k leads to excellent exit times, as larger k imply a more compact layout and thus shorter travel distances (which are not absorbed by the additional shunting distance). As expected, exit time for FIFO is catastrophic, as too many cars try to leave simultaneously, blocking each other. Yet, larger k improve the situation considerably again. In the FIFO case, a single exit is a bottleneck, which is why three exits (one per road) effectively divide the exit time by 3 (not depicted). In the other cases, there is no bottleneck effect.

Shunting Time Figure 13 subtracts the minimum driving time from the total exit time (see Figure 12), showing the resulting time required for shunting operations. The minimum driving time is the time it takes to drive from the entry to the k -stack (not into it, though), and from the stack (not out of it) to the exit.

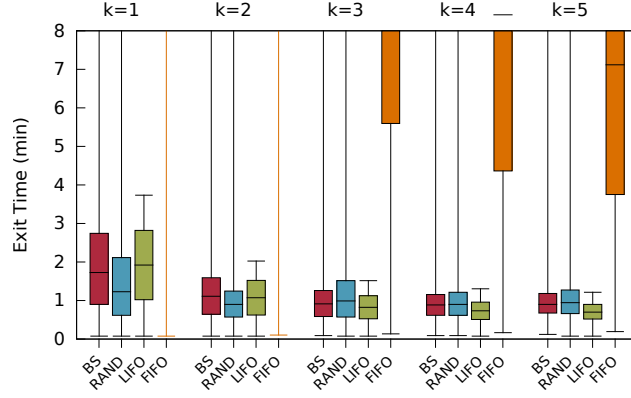


Figure 12: Exit time in minutes per test case and per k .

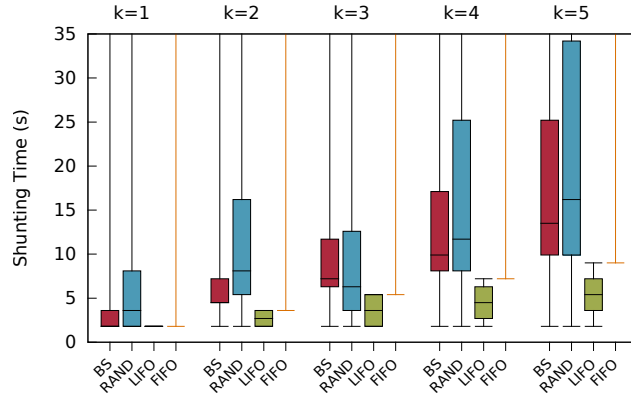


Figure 13: Shunting time (as the difference between total exit time and the minimum driving time through the parking lot) in seconds per test case and per k .

In the LIFO case, shunting time is about 2s for $k = 1$, which is exactly how long it takes to back out of the parking spot. This increases for larger k , as the “depth” of the stack and thus the driving distance/time increases. The RAND and BS cases work similarly, yet add waiting time, caused by vehicles being blocked during their shunting and driving operations and having to wait until the street is unblocked again. As larger k result in more shunting operations, the shunting time increases. Yet, on average it is still only about 17s in the RAND case for $k = 5$, which is negligible compared to the driving time, as was shown in Section 3.3.4. Again, FIFO shunting time is outside of the normal range of measurement—since all vehicles are simultaneously unparking and blocking each other, the average shunting time for $k = 1$ is about 30 min, which decreases significantly (almost by factor 2) for larger k .

Shunting Distance For $k = 1$ the shunting distance is minimal and equal for every car in all scenarios—they only need to back out of their respective parking spot once. As k increases, each car has (a) to drive further into/out of the stack, and (b) to perform shunting operations, as described in Section 3.3.4. Due to space constraints, this is not depicted. We note that, as expected, the shunting distance is limited by Theorem 3.

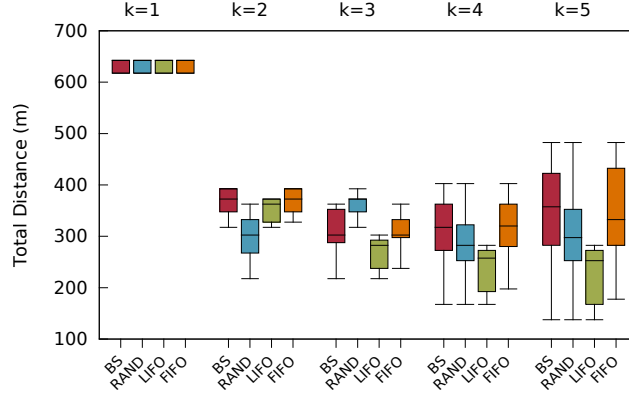


Figure 14: Total travel distance in the parking lot per car in meters.

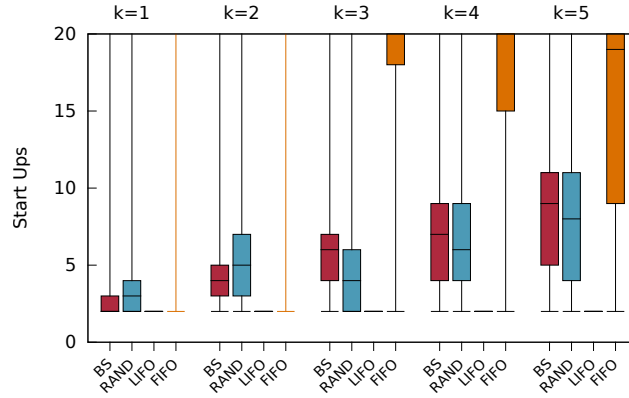


Figure 15: Start-ups per car.

Total Distance The total travel distance in the parking lot per car, as depicted by Figure 14, is the same for all cases if $k = 1$, because it equals to the distance between entry and exit (with minimal intra-case differences for vehicles on different roads, explaining the box size). For larger k , the parking lot's width in x direction decreases, but increases in y direction. Thus, the travel distance from entry to exit decreases significantly, as Figure 14 shows. However, the positive effect of a less wide parking lot is stopped (and eventually absorbed) for $k \geq 3$, because of the increased stack length which increasingly requires more shunting operations (and results in longer cross streets, too). This is generally true for FIFO, BS and RAND. The LIFO case is different in that no shunting operations are required and thus the positive trend continues longer.

Start-ups Figure 15 shows the start-ups per car, i.e., how often a vehicle has to accelerate from a standstill, which is an important metric from an energy efficiency point of view. In LIFO, each car has 2 start-ups independent from k —one to back out of the stack, and one to drive forward to the exit. In BS and RAND, more start-ups are required because of shunting operations (which are limited by Theorem 2) and stop-and-go traffic on transit roads.

3.4 Conclusion

In this paper, we have presented a novel parking management approach for automated valet parking based on k -stacks. k -stacks exploit autonomy of vehicles to significantly increase the parking lot density, resulting in considerable space and cost savings in urban environments. Our model limits the total number of shunting operations as well as the total shunting distance per car by $O(k)$ and $O(k^2)$, respectively. Additionally, it guarantees limited pick-up times, as no car is ever blocked by more than $k - 1$ others. We have verified the theoretical promises of our model in four different simulated scenarios, one of which is based on real-world data from a downtown parking garage. Simulations show that very good pick-up times of about 1 min are possible with very little overhead in terms of shunting distance and time. Moreover, the number of start-ups per car is limited, which is crucial for energy efficiency and for minimizing attrition. It turns out that $k = 3$ provides a very good trade-off between parking lot density, pick-up time and travel distance. For future work, we plan to explore provably optimal strategies for car placement to further improve the performance of our approach.

4 Charging Station Scheduling

4.1 Introduction

EVs still have a limited cruising range of only 100–200 kilometers so far, requiring frequent recharging. Yet, the number of charging stations at large parking areas will be limited; due to cost reasons, not every parking spot will be equipped accordingly, requiring an efficient management of the scarce charging station resource. In this section, which has been published as a separate paper [TW14], we present Charging Station (CS) scheduling strategies that provide a central aspect of this system, i.e., the efficient management and assignment of charging stations to connected electric vehicles. The developed software components are evaluated in a simulation environment with real-world parking statistics to learn the suitability of the implemented scheduling algorithms for different usage scenarios.

The remainder of this section is structured as follows. In Section 4.2, we study related work on charging station scheduling. Section 4.3 proposes five different scheduling algorithms. In Section 4.4, we use real-world statistics to obtain an airport and a downtown parking scenario. These scenarios are used in Section 4.5 to evaluate the scheduling algorithms. Finally, we summarize and draw conclusions in Section 4.6.

4.2 Related Work

In [Kim+10], a reservation-based scheduling algorithm for charging stations is proposed. In our scenario, drivers can make reservations at a charging station in advance, using a vehicular network. The authors define a linear rank function that takes the vehicle’s estimated time of arrival, the required amount of energy, and the waiting time tolerance, i.e., the time a vehicle can wait until it is served, into account. The computation of the rank function can be done in $\mathcal{O}(n)$ and provides a means to decide whether or not a charging request can be served. In addition, it provides the service order of all charging requests. In contrast to our work, though, this model only supports a single charging station, and the authors miss the opportunity to evaluate the scheme’s performance in simulations.

A large-scale network of collaborating charging stations with an emphasis on minimizing waiting times for drivers is presented in [QZ11]. In this work, the authors assume that a system of spatially distributed charging stations is installed at highway exits. They introduce a scheduling scheme, in which charging stations communicate with each other as well as with EVs. The charging stations exchange information about existing reservations in order to refine their estimation of current and future charging waiting times for themselves and other stations. In this way, stations can help vehicles requesting a reservation to select the station with the minimal waiting time. This scheme depends on spatial distribution, i.e., on charging stations at different highway exits. Our focus, on the contrary, is on centralized parking and charging scenarios, such as an airport parking structure, in which the abovementioned scheme is not applicable. Moreover, we take several more evaluation metrics into consideration, as compared to only the waiting time.

To the best of our knowledge, most other studies in the field of charging station scheduling focus on avoiding charging during critical peak periods, lest the power grid experiences overall or partial failure due to overdemand. In addition, the price of electricity is usually the highest during peak hours. In [EH10], a prediction-based charging scheme is proposed that predicts the market prices based on information exchanged with the grid, and is thus able to choose a charging period with low cost. Similarly, [GTL11] proposes decentralized charging schemes that seek to “fill the overnight valley” [GTL11] of demand for electricity. [JTG13] provides optimal solutions for scheduling EV charging with energy storage in a smart grid system in order to maximize the aggregator’s revenue. In [Fre+10], the impact of different charging schemes on the Portuguese power grid are discussed and evaluated. Further, the authors introduce a vehicle-to-grid

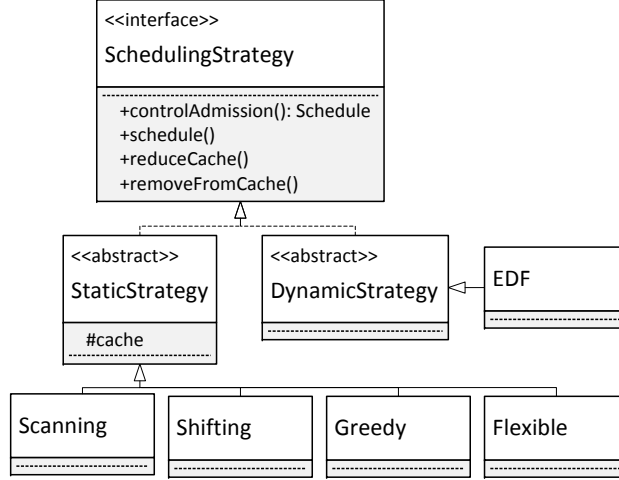


Figure 16: Overview of implemented schedulers.

(V2G) scheme, which allows EVs to basically absorb abundant energy from the grid over night, and put it back on the grid during peak periods.

Daily load forecasting for charging stations is examined in [Xie+11]. The authors propose different models based on daily load data of a battery switch station for public transport buses in Beijing. This approach obviously has different constraints than a parking lot scenario in which personal cars are being charged conductively or inductively.

An approach that uses EV batteries to support the grid is introduced in [SKK12]. To this extent, different CSs are connected to the same distribution node, called a multi charging station (MCS), enabling an easy and controllable flow of power. Moreover, a framework for handling state of charge (SOC) and charging/discharging rate is proposed.

As opposed to our contribution, however, none of the abovementioned works [EH10; JTG13; GTL11; Fre+10; Xie+11; SKK12] take the specific and often contradicting requirements (parking time, required energy, etc.) of drivers, who are competing for scarce charging resources, into account. Instead, they focus on the impact of EV charging on the power grid. Moreover, the use of EV batteries to stabilize the power grid by charging during low-demand hours or even discharging batteries during high-demand often contradicts driver requirements in the use cases we address in this paper.

A Quality of Service (QoS)-aware admission control scheme for plug-in hybrid EVs is presented in [ESM12]. In this scheme, EV owners who are willing to pay more can charge faster. This requires changes to the charging infrastructure, though, which is why we do not take this type of service differentiation into account.

The specific requirements for scheduling schemes applicable to the V-Charge scenario have been presented in [TW12c].

4.3 Charging Station Scheduling

In this section, we propose several static and one dynamic scheduling algorithms for CS assignment, which have been previously described in D5.1, though in an earlier version. These take an arrival time, and optionally, an energy demand and a departure time as input parameters. The arrival parameter can be up

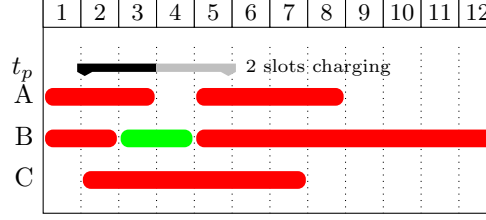


Figure 17: Example schedule illustrating the Scanning strategy.

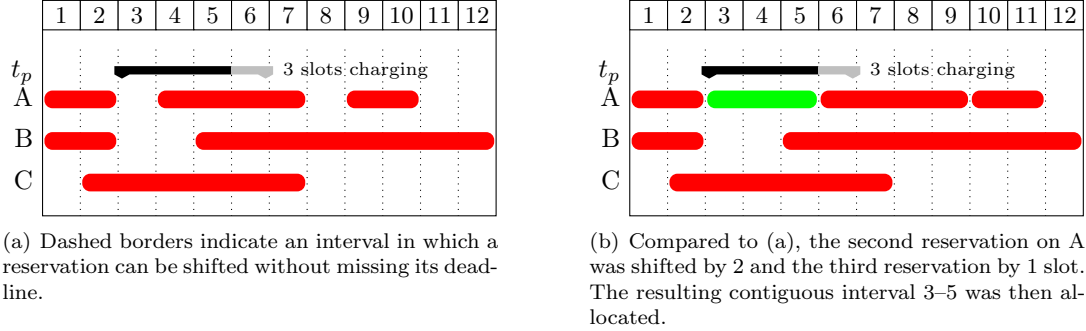


Figure 18: Example schedule illustrating the Shifting strategy before (a) and after (b) allocation.

to several months in the future or be the present if a driver drops off his vehicle without having reserved a spot in advance. The driver can specify his planned departure time or leave it undefined, if he does not know or have a departure deadline. Drivers can request a specific energy the vehicle will be recharged with; otherwise the system will optimize the vehicle's battery level depending on the actual parking time. An overview of our schedulers is given in Figure 16, which indicates that each scheduler has to implement the `SchedulingStrategy` interface, which defines methods to control admission of a vehicle in the context of a check-in request, to remove/retract reservations, and to schedule vehicles. Schedulers can be divided into two groups: static and dynamic. These groups and their representatives are detailed on in the following. All schedulers generally try to at least partially accept user requests. Only if the available resources fall under a configurable lower bound (5 minutes charging time in our case), the request is rejected. All strategies can further be distinguished by the number of CSs they allocate for a single vehicle: Scanning and Shifting only use one charging station, while Greedy, Flexible and EDF are able to split the charge between an arbitrary number of stations. If vehicles are assigned more than one CS and thusly have to move between the stations, an overhead for the additional driving and the charging setup is caused. Though this overhead might be significant in certain urban parking scenarios, we mitigate this problem by generally trying to minimize CS switches and enforcing a minimum time per CS constraint.

4.3.1 Static Scheduling

Static scheduling algorithms make charging station assignments upon a vehicle check-in, and can thus guarantee a certain amount of energy that the vehicle will be provided. While the selection of charging stations and charging times is specific to each algorithm, they share common features. Each static scheduler keeps an in-memory cache of reservations per charging station in order to quickly and efficiently find available charging times.

Each static scheduling strategy is illustrated through an example, such as Figure 17. The x-axis shows

abstract time slots that exemplarily stand for twelve consecutive hours of a day. On the y-axis, one can see different charging stations, labeled A through C. Red bars (filled, solid border) in charging station rows indicate that the CS is occupied by another vehicle, whereas unmarked (white) slots mean that it is available at that time. Green bars (filled, dashed border) indicate time slots that are being allocated for the vehicle that is currently checking in and requests charging.

Consider now that a new check-in request is processed. The time interval in which the user wants to park is indicated by the parking time t_p . The black fraction of t_p stands for the amount of time the vehicle needs to be charged in order to meet the user's energy requirement. In Figure 17 this means that the user wants to park in interval 2–5, and two of the interval's slots are the required charging time. The charging time can be arbitrarily allocated throughout the parking interval, i.e., 2–3, 3–4, or 4–5. Obviously, the vehicle's total parking time exceeds its charging time (4 slots instead of 2). Thus, and in order to make the CS available to other vehicles as soon as possible (ASAP), the vehicle is going to be parked on a regular parking spot when not charging. For the assignment of regular parking spots, different strategies, e.g., shortest path to a vacant spot, can be applied.

Scanning This scheduler is the simplest static algorithm: for a check-in request, it successively scans all charging stations to find contiguous time slots that can accommodate the vehicle's charging requirement during its parking interval. If it cannot find a sufficient charging interval, it will automatically allocate the largest interval available. Either way, a single reservation on one charging station that matches the initially requested charging time as well as possible will be created. A reservation request will only be rejected if no time slot larger than the abovementioned lower bound is available.

In Figure 17, charging station A's only free slot in the interval 2–5 is slot 4. The vehicle requires a charging time of 2 slots, though. Charging station B, on the contrary, is available from 3–4 and therefore this interval is reserved. If several options exist to make a reservation, the earliest one will be selected, following an ASAP charging strategy which is beneficial for a user picking up his vehicle earlier than originally estimated. With an ASAP strategy, his vehicle will have been charged as much as possible already.

Shifting This scheduler is more advanced than Scanning in the sense that it is able to shift reservations (on the same charging station) in order to create a time interval large enough to accommodate a check-in request. It therefore not only stores a reservation's start time in its cache when the vehicle is admitted, but also the latest point in time the reservation needs to start in order to comply with the anticipated departure time. Because the scheduler will try to find a reservation starting as early after the vehicle's arrival as possible, this will create an interval in which each reservation can be shifted without missing its deadline. In Figure 18(a), the only existing reservation on CS C is in interval 2–7. As the dashed border (unfilled) indicates, however, it can be shifted backward by four time slots.

The scheduler iterates over all charging stations' reservations within the parking interval and checks if the time slot before or after each reservation in the interval accommodates the parking requirement, if necessary by recursively shifting reservations backward. If it cannot find such an interval despite its effort to shift reservations, it will automatically allocate the largest interval available. Figure 18(a) shows the CS reservations in the requested interval. Charging station A can meet the required charging time of 3 slots starting at slot 3. Yet, in order to do so, the subsequent reservation needs to be shifted by 2 slots. This creates a conflict with the third reservation (slots 9–10). This is resolved by shifting the third reservation by 1 slot, resulting in the situation depicted in Figure 18(b).

Greedy The Greedy scheduler is more complex in that it is able to distribute charging time among several charging stations, thus composing multiple reservations to accommodate the vehicle's check-in request. It

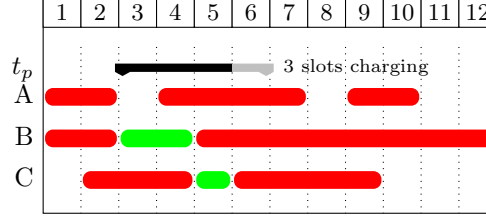


Figure 19: Example schedule illustrating the Greedy strategy.

does so by scanning all charging stations to find and greedily reserve the largest free interval available within the parking interval. It then scans all stations again to find the next largest interval. This scheme is iteratively repeated until the entire charging time is covered or it becomes clear that the request cannot be accommodated. In that case, the scheduler will reduce the requirement and automatically allocate the longest possible charging time. In order to avoid station hopping, in which a lot of small intervals are allocated, resulting in high overhead due to frequent moving from one station to another, an interval's size has to fulfill a minimum charging time per station constraint.

As mentioned above, Greedy does not require reservations to be on the same charging station. Instead, they can be distributed among all stations. However, this algorithm is limited in that it requires allocated intervals to seamlessly tie in with previous intervals. That means that if one interval was found and reserved, the next interval has to start immediately after the first one ends—there cannot be a pause between two CS reservations. This avoids that a vehicle alternately charges at a CS and parks on a regular spot, which would induce a bigger overhead for moving the vehicle. In Figure 19, the largest interval that starts simultaneously with the parking time is on CS B. The next largest interval that is adjacent to the first reservation is on CS C. The combined charging time of both intervals meets the requirement of 3 slots.

Flexible The Flexible scheduler is the most advanced among the implemented static algorithms. It builds upon the Greedy scheduler, but overcomes Greedy's limitation of reservations having to seamlessly tie in with each other, meaning it allows vehicles to alternately charge on a CS and park on a regular spot. It selects the largest intervals available across all charging stations and reserves them to accommodate the required charging time as depicted by Figure 20. During slot 5, i.e., between its CS reservations, the vehicle will be parked on a regular parking spot. If the initially requested charging time cannot be fulfilled, this scheduler will allocate the longest possible charging time, too. Station hopping is also avoided by having intervals to fulfill a minimum charging time per station constraint.

4.3.2 Dynamic Scheduling

This scheduling algorithm implements Earliest-Deadline-First Utilization-Separation with separator $m/(2m-1)$, in short EDF-US($m/(2m-1)$) [AAJ03]. It is a fully dynamic, earliest deadline first priority-based scheduling scheme, that admits jobs (i.e., vehicles) as long as an admission counter is less than a static capacity bound. The capacity bound is initialized during startup as $m/(2m-1)$, with m being the number of charging stations in the system. The admission counter A is initialized to zero.

For each check-in request, a new job i is created with the utilization $u_i = t_c/t_p$, with t_c and t_p being the charging time and parking time, respectively. If the condition $u_i/m + A \leq m/(2m-1)$ holds, the vehicle is admitted and guaranteed to meet its deadline (i.e., the departure time). Each admitted job i increases A by u_i/m . If all CSs are idle, A is reset to zero. For each job i that ends, A is decreased by u_i/m , if i arrived after or at the time of the last reset. If the abovementioned condition does not hold, the job would typically

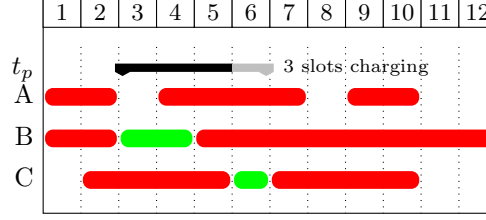


Figure 20: Example schedule illustrating the Flexible strategy.

be rejected. In our implementation, though, the scheduler automatically decreases the parking time (and thus u_i) and re-checks the condition. Thus, it tries to find the longest possible charging/parking time that can be offered to the vehicle. Only if no charging is possible at all (i.e., only intervals of less than 5 minutes are available), the job is finally rejected.

Each job is in exactly one of three states: **inactive**, **active**, or **running**. A job that has been admitted starts as **inactive**. **Inactive** jobs are stored, but they are not eligible to be scheduled (i.e., to make a CS reservation). When the corresponding vehicle is dropped off by the driver, the job becomes **active** and eligible to be scheduled from that point on. For each activated job i , the admission counter is increased by u_i/m .

Every time the `schedule` method is invoked (that is once every dispatch interval, for instance every 2 minutes), it selects as many **active** jobs ordered by their priority as there are idle charging stations. These jobs then become **running**, which means that a charging station reservation with a fixed charging time is created for them.

Jobs that have been **running** for their assigned time frame are set back to **active**, provided that the job is not about to end in the current interval, which could happen either by the vehicle having been charged up to the required level or by being picked up earlier as expected. An ending job i is not set to **active** anymore. Instead, the admission counter is decreased by u_i/m and the job is deleted.

Dynamic scheduling strategies like EDF inherently suffer from large numbers of context switches or, in our case, charging station switches. This is the case because each time the scheduler is invoked, it might select different (higher priority) jobs, suspending previously running jobs. In order to at least minimize the cost of (potentially frequent) switches, it is necessary to minimize the distance between a vehicle's current parking site and its scheduled destination site.

4.4 Simulation Setup & Scenarios

First, we describe the design of our simulation in Section 4.4.1. Second, in Section 4.4.2 we argue which configuration parameters for the simulation are reasonable. Where possible, real-world parking statistics are used to derive meaningful input values.

4.4.1 Simulation Setup

To evaluate our system, we have designed a simulator for the V-Charge Server that automatically produces input values for all necessary customer interaction steps, namely check-in, handover, and pick-up. Figure 21 exemplifies the life cycle of a simulated vehicle in the V-Charge system and shows which parameters the simulator needs to specify. These *simulation parameters* are explained in the following:

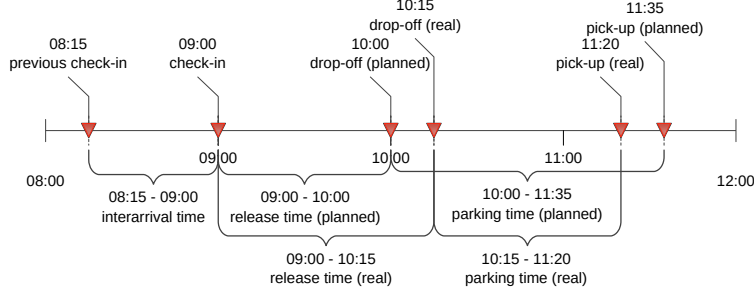


Figure 21: Example timeline showing the interrelation of simulation parameters.

- *Interarrival time*: The time between two check-ins. The interarrival time is derived from the arrival process, i.e., the number of vehicles arriving in a certain period of time.
- *Energy demand*: The energy demand for the checked-in vehicle (not depicted).
- *Release time (planned)*: Time between when the check-in request is made, and the planned arrival time of the vehicle at the parking lot.
- *Release time (real)*: The time between when the check-in request is made, and the actual arrival time of the vehicle at the parking lot. This can be equal to the planned release time, but typically differs because customers arrive early or late, e.g., because of traffic. The release time may be zero if the customer did not reserve a parking spot in advance.
- *Parking time (planned)*: The duration for which the customer is making a parking reservation, i.e., the time between the planned arrival and planned departure.
- *Parking time (real)*: The duration for which the customer parks, i.e., the time between the actual arrival and actual departure. If the vehicle is dropped off and picked-up at the same time as specified during the check-in, this is equal to the planned parking time. Typically, it differs, because vehicles are dropped off and picked up early or late.

For each simulated vehicle, the simulator creates the aforementioned parameters according to strategies specified in Section 4.4.2. The resulting events *check-in*, *drop-off*, and *pick-up* are kept in memory ordered by when they are due. When an event is activated, the simulator sends the corresponding request to the V-Charge Server. Subsequent events that depend on preceding events, such as a drop-off depending on a check-in, are only activated if the preceding event was successful. Hence, a drop-off request will only be submitted to the server if the server has accepted the preceding check-in. If, for instance, the parking requirements could not be scheduled (as explained in Section 4.3), the check-in request would have been rejected and the simulator would have generated neither a drop-off nor a pick-up event for this vehicle.

4.4.2 Simulation Parameters

As described in Section 4.4.1, the simulation depends on an array of parameters. In order to set up the simulation as realistically as possible, we sought to obtain real parking statistics. Despite great efforts, it proved extremely difficult to do so. We contacted more than 30 different parking lot operators, such as private companies and airport and city authorities in Los Angeles, San Francisco, San Diego (all California), Berlin, Hamburg, Stuttgart, Frankfurt, and Munich (all Germany). Almost all contacted operators either had no statistics available or refused to cooperate. Eventually, we succeeded in obtaining parking statistics from Hamburg Airport and the City of Braunschweig, both in Germany. We use these statistics to develop

Table 2: Simulation parameters per scenario

	Scenario 1	Scenario 2
Setting	Airport Hamburg	Braunschweig
Interarrival time	$\lambda = 15$	$\lambda = 80$
Energy demand	(2,23,20) [kWh]	(0,20,4) [kWh]
Release time (planned)	(0,2,1) [days]	see Table 3
Release time (dev.)	(-20,20,0) [minutes]	(-20,20,0) [minutes]
Parking time (planned)	see Figure 22	(0.25,12,2) [hours]
Parking time (real)	$\pm 50\%$ (planned)	$\pm 50\%$ (planned)
# Parking spots	200	1250
# Charging stations	{1,3,5,8,10,15}	{10,20,30,40,50}
# Days	30	30

two simulation scenarios, that are detailed in the following. Because of the different parking usage at airports compared to downtown areas, these scenarios afford us to examine the effectiveness of the V-Charge system in two very likely mission scenarios.

Our goal is to describe the system’s behavior through finite samples. We thus define an empirical or theoretical distribution for each input parameter, in order to generate according random values. Where the sample data are detailed enough, we favor using an empirical distribution as opposed to deriving a theoretical distribution, because:

- Adapting a theoretical distribution yields loss of information. The sample contains all the observed data.
- It is oftentimes not clear which distribution family fits best.
- In many cases, the estimation of distribution parameters is not robust, i.e., minor fluctuations in the data yields major discrepancy in the parameters.

Unfortunately, there are some cases where no data are available at all, making an accurate modeling of input parameters impossible. However, oftentimes it is still possible to estimate some characteristics of the data, e.g., minimum and maximum values. If so, the data can be approximated through a triangular distribution, whose probability density function (PDF) is given by $f(x|a,b,c)$ with lower limit a , upper limit b , and mode c .

Table 2 contrasts the parameters of the airport scenario with those of the downtown scenario. Where a triangular distribution has been used, $f(x|a,b,c)$ is abbreviated to (a,b,c) . In the following, we shortly describe both scenarios. Due to space constraints, we can only cover some selected parameters.

Scenario 1: Airport In this scenario, we evaluate the V-Charge Server in a setting that resembles parking at Hamburg Airport. The V-Charge system is very well suited in this scenario because due to its autonomic parking capabilities it releases customers from their concerns to find parking and to walk or to take a shuttle to the terminal.

Interarrival time The arrival process at an airport parking structure can be very well described by a Poisson process, because the length of the observation period is fixed by our simulation duration, the arrivals can be

assumed to occur at a constant average rate and to be independent from the time of the last arrival. This assumption is well documented in operations research literature and commonly assumed in transportation applications [Tra10]. We derive a model of the arrival process based on the distribution of length of stay of vehicles we received from Hamburg Airport (see an extract of the full data in Figure 22). These data represent all customers at parking facility PH4 in September 2011. The total of customers is 117 257, distributed over 30 days. In order to keep the simulation duration per scenario feasible, we decrease the expected number of arrivals by one order of magnitude and round the result to make calculations easier, yielding $\lambda = 15$ reservation requests per hour. The decrease by the factor 10 also corresponds to the simulated parking capacity, which is one order of magnitude smaller than PH4’s very high average occupancy of about 2000. The actual interarrival time, i.e., the time between events in a Poisson process, can now be described by an exponential distribution, whose PDF is given by:

$$f_{inter}(x|\lambda) = \lambda e^{-\lambda x} = 15e^{-15x}, x \geq 0.$$

Energy demand Since the market share of EVs is very small yet, there are no statistics about the percentage of EV owners among parking customers, and consequently no data about their energy demand are available. We therefore use a triangular distribution to estimate the energy demand for this scenario. It is reasonable to assume an average battery capacity of 20 kWh for EV in the compact class [Grü+11a]. Further, it is unlikely that EVs parking at Hamburg Airport have an energy demand of less than 2 kWh, since the airport is about 10 kilometers away from downtown Hamburg and therefore from most residential areas. Assuming that the average mileage is 20 kWh per 100 kilometers (which is a realistic figure in urban traffic with current compact battery electric vehicles according to [Grü+11b]), yields the aforementioned 2 kWh minimum energy demand.

Release time (planned) We restrict the release time to be between zero and two days. We estimate the expected value of the planned release time to be one day. In other words, customers are expected to check-in their vehicle one day before their actual parking requirement starts. Yet, the system can also handle customers who do not check in their vehicle in advance, but do so immediately before their reservation starts (i.e., a release time of 0).

Parking time (planned) When examining the length of stay distribution (see Figure 22), one will notice that the majority of customers parks for up to two hours. At first view, this may seem unexpected, because air travel usually implies a much longer absence of customers. However, the data can be explained as follows. The data were recorded at parking structure PH4, which is the central parking facility at Hamburg Airport and is mostly used for short-duration and daily parking, e.g., by business travelers. Long-duration parking for up to 3 weeks is also observable, but not very significant. Very peculiar are the peaks for whole days, e.g., at 1 d, 2 d, etc. This is likely to be a result of the price structure at Hamburg Airport: customers pay per quarter of an hour. For parking times longer than 6 hours, a unit price for the whole day applies. Consequently, there are values for the first six full hours per day (i.e., an aggregation of 4 values per hour), and after that all other customers fall in the same category of full day parking.

Scenario 2: Downtown In this scenario, we evaluate the V-Charge Server in a setting that resembles parking in downtown Braunschweig, where the V-Charge system saves customers valuable time from searching for a vacant parking spot / charging station.

Interarrival time To the best of our knowledge, there is no reservation system for parking garages in downtown Braunschweig (which is also true for the other German cities we have examined). Hence, there are no data about how many reservations per hour would be made, if customers had this option. Nevertheless, we were able to obtain parking statistics from a major parking facility in downtown Braunschweig. These data were recorded during September 2011 and show how many vehicles per day and business hour physically

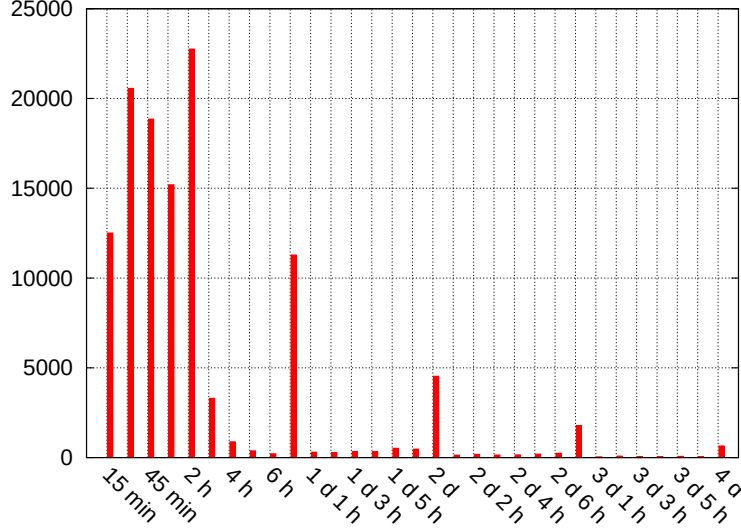


Figure 22: Empirical distribution of length of stay at Hamburg Airport (extract).

arrive at the parking structure. The total arrivals in September 2011 were 27 643, yielding

$$\lambda_{Braunschweig} = \frac{27\,643}{30 \cdot 24} \approx 38$$

arrivals per hour, while the average occupancy was 567. $\lambda_{Braunschweig}$, however, is the average physical arrival rate. The interarrival time as we model it represents the time between reservation requests per hour. Thus, we set the arrival rate such that the actual physical arrival rate equals the data in Table 3.

Release time (planned) The break down of the raw data (averaged over a week) into arrivals per business hour and weekday can be found in Table 3. These data were recorded in September 2011.

We use the more detailed data (that have not been averaged) as an empirical distribution for the release time, such that vehicles only arrive at the parking lot during the following business hours:

- Monday through Saturday: 07:00 – 21:00
- Sunday: 11:00 – 18:00

The number of vehicles arriving at a certain day within a certain hour mirrors the arrivals per weekday and hour in September 2011. The capacity of the parking structure used is 1250.

Parking time (planned) The most likely use case is a customer who uses the parking structure to go shopping in a nearby mall for, say, 2 hours. Very short parking in order to post a letter, for instance, may only require 15 minutes of parking. Some customers may also use the parking structure while they are at work, yielding an upper bound of 12 hours (includes hours of work, potential long hours, and commuting time between the parking structure and the place of work).

Parking time (real) In order to evaluate our system as realistically as possible, we also take a deviation of the planned parking time into account, like we did in Scenario 1. This is relevant in cases where customers cannot exactly estimate how long their trip or errand (e.g., picking someone up) is going to take and thus over- or underestimate the parking time. We model the real parking time relative to the absolute (planned) parking time, with a deviation of $\pm 50\%$.

Table 3: Average number of arrivals per weekday and hour in a major downtown Braunschweig parking structure.

Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
7	23.75	26	29.25	23.5	24.2	5.25	–
8	65	62.75	60.75	69.25	74	39.5	–
9	123.25	114.25	124	128.75	119	166.75	–
10	86	78	94	77	83	123.75	–
11	90.25	80	76	77	89	80.25	45
12	76.75	68.5	79	81.75	87.6	79.25	68.75
13	90.5	77.25	91.75	79.75	102.4	94.5	58.25
14	85.75	80	90.25	102.5	103.8	89.5	59.75
15	95.5	93.25	108.5	80.75	115.4	93.25	58.25
16	97.75	94.5	98.25	100	91.6	77.25	46
17	69.25	84.25	79.25	78.75	67.8	64.75	20.75
18	61.25	55.75	53.5	58.25	60.6	48.5	–
19	28	27.75	27.5	35.25	34.6	40.25	–
20	3.75	5.75	3	4.5	6.4	13.25	–

4.5 Evaluation

The focus of our evaluation is on the suitability of the schedulers described in Section 4.3 for the real-world scenarios described in Section 4.4.2. Moreover, we show that the V-Charge Server is able to efficiently handle realistic parking volume and performs well in fulfilling customer requirements, e.g., energy demand for the next driving task. First, we develop metrics to evaluate the performance of the implemented scheduling strategies. Second, we apply these metrics and present our results per scenario. Because of the large amount of data generated in the simulations, we are only able to show the results either in aggregated form or selected details of them.

4.5.1 Metrics

In order to determine the most effective vehicle scheduling strategy it is important to define appropriate metrics. Since we are not developing a commercial product, metrics commonly used in business, such as revenues, payment methods, operating costs, and marketing goals, are not considered here. Instead, we focus on:

- Facility use, such as:
 - Overall peak-period utilization (e.g., daily spaces occupied),
 - Charging station utilization, and
- Customer satisfaction, such as:
 - Percent satisfied customer parking requirements.

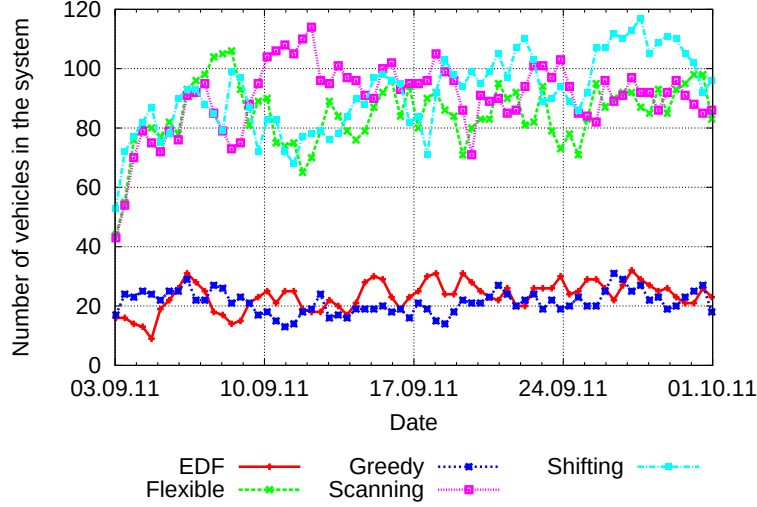


Figure 23: Number of vehicles in the V-Charge system in Scenario 1 during the simulated time period per scheduling strategy, with 15 CSs.

4.5.2 Scenario 1: Airport

For Scenario 1, we have simulated all scheduling strategies for 30 days each. Furthermore, in order to determine the impact of the number of charging stations in the system, each strategy has been simulated in 6 different configurations, namely with the number of charging stations set to $i \in \{1, 3, 5, 8, 10, 15\}$. In sum, 30 different simulations were run, with a simulation time of 30 days each.

Utilization Figure 23 plots the number of vehicles in the V-Charge system, i.e., how many vehicles had a confirmed reservation at each point in time, against the simulated time period of 30 days. Each scheduler had 15 charging stations at its disposal. Two groups can be identified: first, schedulers with very low overall utilization, namely EDF and Greedy. Second, high utilization schedulers such as Flexible, Scanning, and Shifting.

The poor performance of EDF can be explained by taking its admission control mechanism into account. As described in Section 4.3.2, a job i is only admitted if $u_i/m + A \leq m/(2m - 1)$. With the majority of vehicles only parking for up to two hours, the distribution of energy demand in this scenario yields a high average u_i per vehicle. For instance, a vehicle i parking for only 15 minutes, but with an energy demand of 10 kWh (which corresponds to 30 minutes charging time in our simulation setup), cannot be fully charged within its parking time. Hence, its charging duration is automatically reduced to 15 minutes, in order to make the most of the situation. Consequently, $u_i = \frac{t_c}{t_p} = \frac{15}{15} = 1$. As a result, only very few vehicles with similar parking requirements are necessary to let the admission counter reach the capacity bound, thus not admitting any more vehicles.

The reason why Greedy does not perform well is a shortcoming already described in Section 4.3.1: the Greedy algorithm requires allocated intervals to seamlessly tie in with previous ones. With an increasing number of short intervals allocated, it becomes more difficult to find adequate time slots. Since Flexible overcomes this drawback, it performs significantly better.

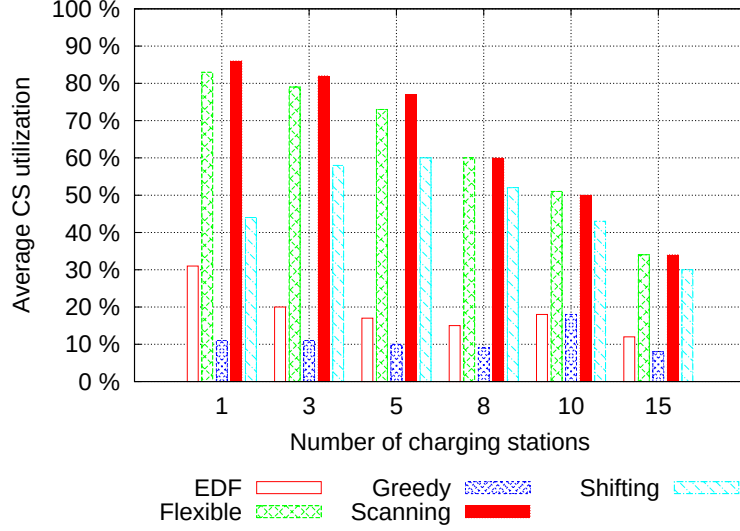


Figure 24: Average charging station utilization for Scenario 1.

CS Utilization Figure 24 shows the average charging station utilization for each scheduling strategy as a function of the number of charging stations in the system. In general, the average CS utilization decreases with an increasing number of charging stations in the system. This indicates that the system was not used to its full capacity, which the below-mentioned evaluation of accepted check-in requests confirms.

One will notice that Flexible and Scanning consistently outperform the other strategies, closely followed (for at least 5 CSs) by Shifting. EDF only achieves a low CS utilization, yet Greedy performs the worst, stagnating at about 10%.

Satisfied parking demand A crucial viability factor of any system is the percentage of satisfied users. In our case, this is particularly the percentage of satisfied parking requirements. The scheduler tries to allocate sufficient parking/charging time to satisfy each user’s parking requirement. If it cannot be met, though, the scheduler has to reject the request. Figure 25 depicts the percentage of accepted check-ins, i.e., parking requirements that can be scheduled, as a function of the number of charging stations.

EDF and Greedy perform poorly, for the same reasons that they only achieve a low overall utilization. From 8 charging stations on, Flexible, Scanning, and Shifting perform equally well, meeting more than 99% of the parking demand. For less charging stations, Scanning is slightly better which is a consequence of the size of unallocated time slots it produces. Scanning produces larger gaps between CS allocations, because Flexible and Shifting distribute charging time over multiple stations and shift reservations backward, respectively, which makes more efficient use of gaps between allocations, thus producing smaller gaps. The more larger gaps exist, however, the more likely it is that the next received check-in request can be scheduled because more contiguous charging time is available. Hence, Scanning can accept and schedule more user requests.

4.5.3 Scenario 2: Downtown

For Scenario 2, we have simulated all scheduling strategies for 30 days each and in 5 different configurations, namely with the number of charging stations set to $i \in \{10, 20, 30, 40, 50\}$. These larger numbers, as compared to Scenario 1, were chosen because of the significantly higher arrival rate (see Section 4.4.2) and in order to

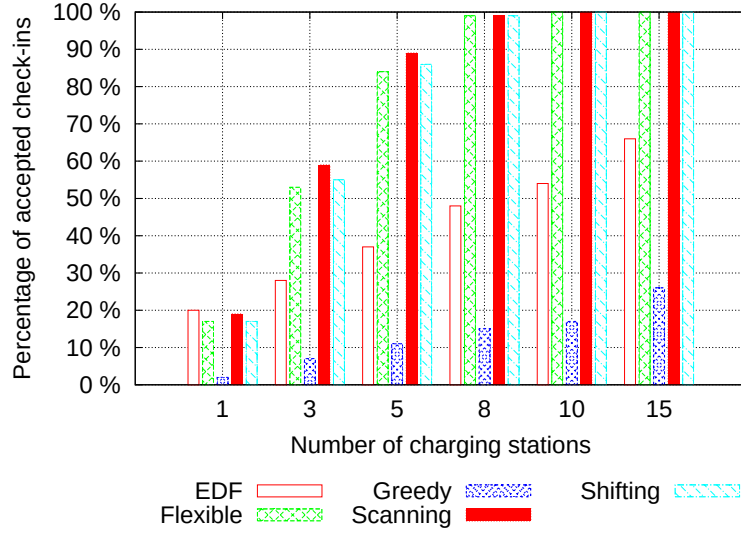


Figure 25: Percent check-ins accepted for Scenario 1.

examine how the system copes with the increased management overhead of large CS quantities. In sum, 25 different simulations were run, with a simulation time of 30 days each.

Utilization Figure 26 plots the number of vehicles in the V-Charge system, i.e., vehicles with a confirmed reservation at each point in time, against the simulated time period of 30 days. Each scheduler had 50 charging stations at its disposal.

Unlike in Scenario 1, we can observe cycles of alternating low and high utilization for all schedulers. This is a result of one of the scenario’s main characteristics: arriving vehicles are bound to business hours. Over night, no vehicles arrive and the parking structure is emptied. During the day, vehicles arrive and the structure fills up. This cycle repeats every day. A peculiar fact is that on certain days, namely September 18 and September 25, the utilization is significantly lower than on most days. This is because these dates are Sundays, which have shortened business hours in Germany. However, September 11, which was also a Sunday in 2011, does not follow this pattern. Instead, it is indistinguishable from weekdays which seems illogical but can be explained with additional data from the chamber of commerce and industry’s Web site³, according to which Sunday shopping was allowed on September 11. Consequently, most stores were open this day from 13:00 till 18:00, attracting parking customers to use the parking structure. A city festival called “Magnifest”⁴ on September 4 yielded a similar effect.

We can divide the schedulers in three categories, as compared to two in Scenario 1. First, only the Greedy strategy has a very low utilization, with a very small number of peaks as high as 75–80 vehicles. Second, EDF achieves an overall utilization significantly better than Greedy, unlike in Scenario 1. Peak values of about 200 vehicles are achieved regularly. Third, Flexible, Scanning, and Shifting achieve very high utilizations of about 350 vehicles, and are visually almost indistinguishable from each other. The reasons for the schedulers’ different performances have been presented in Section 4.5.2 already. A trend that becomes much clearer here, though, is the superiority of Scanning, Shifting, and Flexible over Greedy and EDF. It even shows that Greedy inherently performs poorly, independently from the number of charging stations.

³<http://www.braunschweig.ihk.de/pressearchiv/pressearchiv-2011.html>

⁴<http://www.magnifest.com>

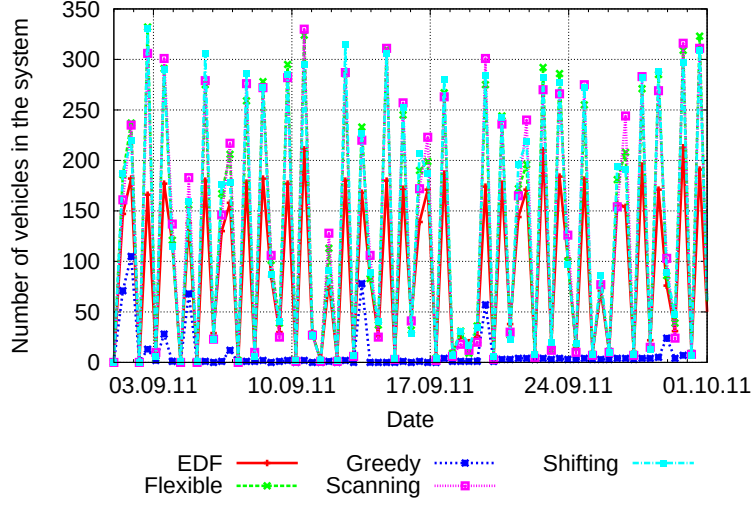


Figure 26: Number of vehicles in the V-Charge system in Scenario 2 during the simulated time period per scheduling strategy, with 50 CSs.

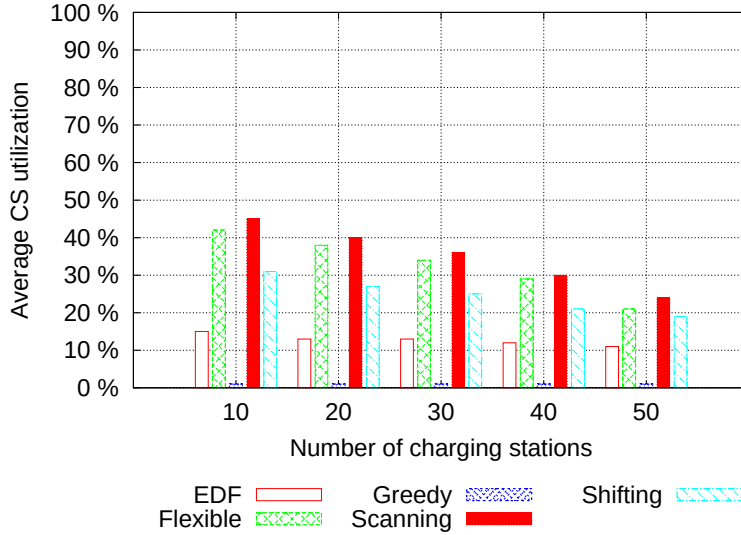


Figure 27: Average charging station utilization for Scenario 2.

CS Utilization Figure 27 shows the average charging station utilization for each scheduling strategy as a function of the number of charging stations in the system. For reasons given in Section 4.5.2, the average CS utilization decreases. As in Scenario 1, Flexible and Scanning consistently outperform the other strategies, followed by Shifting. EDF only achieves a low CS utilization, yet Greedy performs the worst, stagnating at about 1%.

Satisfied parking demand Figure 28 depicts the percentage of accepted check-ins, i.e., parking requirements that can be scheduled, as a function of the number of charging stations. Greedy performs poorly, for the same reasons that it only achieves a low overall utilization. EDF starts out with a low acceptance rate, and only achieves about 70% with 50 CSs. This shows that EDF requires significantly more charging

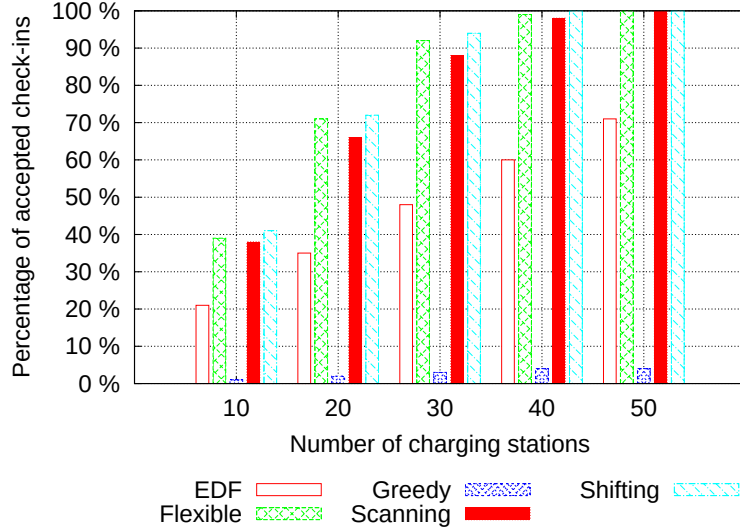


Figure 28: Percent check-ins accepted for Scenario 2.

resources to meet customer demands. From 40 charging stations on, Flexible, Scanning, and Shifting perform almost equally well, meeting more than 98% of the parking demand.

4.6 Conclusion

The combination of an autonomous parking system with automatic recharging capabilities offers great convenience to electric vehicle drivers. It also addresses major challenges to the wide-scale deployment of electric vehicles: the time it takes to charge the battery, and the narrow range due to limited battery capacity. The V-Charge system proposes new concepts for an optimal combination of public and individual transportation as well as for coordinated recharging strategies. Hence, the customer acceptance of electric vehicles is increased by satisfying the customer’s needs.

We described our implementation of parking management and charging station scheduling components. Five different schedulers were implemented and explained in detail. The performance of each scheduler was evaluated in two different scenarios. For this purpose, real-world parking statistics from Hamburg Airport and the City of Braunschweig were obtained and used to derive simulation parameters for typical mission scenarios. To render possible extensive evaluations in a feasible time-frame, we developed a simulation infrastructure and according concepts to speed the simulation process up. With this at hand, we showed that the V-Charge Server is able to efficiently handle realistic parking volume and compared the schedulers’ suitability in two very likely scenarios. With the metrics we have developed, we were able to identify the best suited scheduler for each scenario, namely the Scanning strategy which yields high utilizations while minimizing CS switching overhead.

Future Work Possible future work includes other scheduling schemes that we were not able to examine in the scope of this paper. For instance, other dynamic schedulers, e.g., laxity-based as well as Pfair [Bar+93] algorithms, seem worth examining in more detail. So far, we have only examined homogeneous charging stations, i.e., all CSs in the system had the same specifications. Thus, another interesting idea is to examine the impact of heterogeneous charging stations on the overall behavior.

5 Security Challenges and Solutions

Centralized architectures of today’s Intelligent Transportation Systems (ITSs) impose several security challenges. To address those, we define requirements specifically to V-Charge’s design and develop solutions based on open standards. The following sections are devoted to certain security components, each with its own requirements and attack scenarios. Security challenges and their solutions regarding V-Charge’s central web service are discussed, i.e., secure authentication and session management. Mobile devices, such as smartphones, introduce new usability patterns and also allow more control over client-side security due to the use of applications instead of restricted web services. Thus, different requirements and corresponding solutions are proposed for those client applications. Identity management in the vehicular domain especially regarding V-Charge’s central scenarios is introduced, including secure interaction between cloud and vehicular networks. Of particular interest to us is the question how drivers can securely register their vehicle with the V-Charge service and deploy keys for securing Vehicle-to-Cloud (V2C) communications. Before going into the details of these security challenges, an overview of V-Charge’s architecture is given.

A V-Charge service called ParkingManager (PM) [TW12b] is deployed to each parking lot or garage that is using the V-Charge system. The PM is responsible for managing parking resources by allowing users to make parking reservations and to provide their vehicles with mission control information, starting with their arrival at the check-in location. Since each parking garage has its own PM, there is a central back-end service that connects all PMs (via the Internet) and allows users to register with the V-Charge system, which takes care of authentication. Using mobile devices connected to V-Charge’s back-end server via the Internet, customers can command the system to take control over their vehicles. A local PM will then take-over the control over the vehicle and commands it to drive autonomously to a free parking spot. The central back-end service also serves web services for browsers and other client applications, consisting of a website and a RESTful JSON interface. Customers are able to check the vehicle’s charging status over the Internet using their smartphone or the V-Charge website. Java Platform, Enterprise Edition (Java EE) was chosen as the software stack for V-Charge’s software architecture, which is deployed on Glassfish, an open-source application server [Tim12].

In order to cope with the challenging network conditions of the vehicular domain (e.g., intermittent connectivity) and the V-Charge scenario [TW12b] (e.g., no vehicular Internet access), we facilitate a Disruption- or Delay-/Disruption-Tolerant Network (DTN) architecture [Cer+07] to establish robust and secure V2C communications between the vehicles and the PM. For this purpose, we utilize IBR-DTN⁵ [Sch+11], a lightweight and modular Bundle Protocol [SB07] implementation.

Because of the current rise of smartphones and their rapid integration into our daily life, the V-Charge Android application is the primary tool for users to interact with the system. Its features include reservation making, status checking, and issuing drop-off and pick-up commands that trigger the local PM to take control of the vehicle or to send it to the pick-up zone, respectively. It does not serve as a remote control or access token and is not part of the DTN.

Parts of this chapter were published as “Secure Smartphone-based Registration and Key Deployment for Vehicle-to-cloud Communications” [TSW13] in *Proceedings of the First International Academic Workshop on Security, Privacy & Dependability for Cyber Vehicles (CyCAR ’13)*.

After discussing related work in Section 5.1, server-side security of the central web service is presented in Section 5.2. Our registration and key deployment process requires authorization between smartphone and the V-Charge cloud. Therefore Smartphone-to-Cloud security is introduced in Section 5.3. DTN security between cloud, local PMs and vehicles is described in Section 5.4. Conclusively, a discussion evaluates the overhead of DTN security (cf. Section 5.6). The results presented here supersede the first version of the security concept presented in D5.2.

⁵<http://www.ibr.cs.tu-bs.de/projects/ibr-dtn>

5.1 Related Work

This section deals with authentication and session management, Transport Layer Security (TLS), and password security in modern web applications. Besides those classic web development security practices, the main focus is secure key deployment for DTNs via mobile devices, including authorization protocols. Thus, relevant literature research has been reviewed on security in DTNs in addition to research on general-purpose web security, while focusing on authorization of client applications and deployment of keys.

5.1.1 Web Security

The Open Web Application Security Project (OWASP)⁶ defines best practices to secure web applications and serves as the main guideline for secure web development. This includes authentication configurations and password protection.

Conventional web application authentication can be done via HTTP basic or digest authentication [Fra+99]. The basic authentication allows authentication using username/password combinations via browser-side implementations. The credentials are forwarded as plaintext, imposing the requirement of TLS like in all authentication protocols. While it secures against phishing attacks due to the fact that the login dialog is implemented on the client side, it provides no way for a web application to control sessions. Thus, it is unsuitable for deployment if a web application should be used from public browsers, as this requires a possible logout, i.e., session expiration. The digest scheme is based on a challenge-response protocol using nonces per authentication. While the digest scheme provides more security than the basic scheme and provides mitigation against some TLS attacks, it is not deployed widely enough due to differences in browser implementation [Mor10]. Conclusively, most web services implement their own login technique based on a HTTP web form sent via POST request. Corresponding best practices can be found on the OWASP website.

Several open-web authentication/authorization frameworks have been designed with different scenarios in mind. OpenID 2.0⁷ is designed as a single sign-on framework for establishing identity with a web service [RR06]. It allows identity management using URLs as identifiers and is based on the idea of decentralized OpenID providers. Users can register with one provider and login to other consumers by using their identifier URL only. OpenID 2.0 is not adopted widely enough due to its complex integration into existing identity management systems and its vulnerability to phishing attacks [Bra07].

OAuth 2.0⁸ is designed as an authorization protocol to allow clients access to specific resources. OAuth is not a strictly defined standard; it can be seen as a collection of best practices to build such authorization techniques into web applications. It is widely deployed as an authorization protocol among social media websites, e.g., by Google, Twitter, and Facebook⁹. OAuth clients can delegate authentication to OAuth providers, retrieving an token to access resources of a provider on users' behalf.

OpenID Connect¹⁰ extends the OAuth 2.0 protocol with identity management. While OAuth itself only provides authorization, OpenID Connect permits user identity verification by an API to query for session information using a RESTful interface.

A different approach to completely web-based authentication and authorization frameworks is Mozilla Persona¹¹. Using verified email addresses as identifiers instead of usernames, Persona relies on asymmetric

⁶<https://www.owasp.org>

⁷<https://openid.net/developers/specs>

⁸<https://tools.ietf.org/wg/oauth>

⁹List of OAuth service providers: <http://en.wikipedia.org/wiki/OAuth>

¹⁰<https://openid.net/connect>

¹¹<https://www.mozilla.org/en-US/persona>

cryptography implemented in the browser itself or as a fallback on providers, such as `browserid.org`. If a browser implementation is available, the actual key generation is solely done on the user’s device, which is a significant security gain (cf. browser-based implementation in basic and digest authentication).

5.1.2 Delay-Tolerant Networking

In the V-Charge scenario, Vehicle-to-Cloud communication via DTN is highly centralized; thus, we focus on research in DTN security with centralized scenarios. RFC 6257 [Sym+11] defines basic security message formats, ciphersuites, security blocks, and how to process them; it does not explicitly propose key management frameworks. Some key management techniques were proposed in literature to deploy adequate key management in networks with high delays and no direct connection to central authorities. Pairing-Based Cryptography (PBC) fosters new cryptographic protocols called Identity-Based Cryptography (IBC), where node IDs are also their public keys, previously generated by a central authority using a secret. As this secret is only known to the central authority, valid public keys can only be generated by this authority. In contrast to traditional Public Key Infrastructures (PKIs), the protocol of requesting the public key for an ID from an authority is obsolete. Due to this property, IBC was proposed for DTN [KZH07; Aso+07; Van10]. However, our scenario does not require complex protocols and facilitates connection to central authorities, as vehicles are charged in regular intervals at V-Charge parking spaces.

5.1.3 Key Deployment in Vehicular Networks

To the best of our knowledge, most vehicular key management research that facilitates some kind of PKI assumes that the keys have somehow been securely deployed to the vehicle, for instance by the Original Equipment Manufacturer (OEM) [Pap+08; BFV11] or a vehicle service station [Stu+09]. Others do not consider this point at all [Was+10]. In contrast to the PKI proposed by Car 2 Car Communication Consortium (C2C-CC)¹², for instance, the approach herein focuses on minimizing the trust required in central authorities for key generation/deployment on a per-service base. It consists of a process to generate/deploy keys to a vehicle’s Hardware Security Module (HSM) using a smartphone, under full user control.

Our idea, however, shares some commonalities with car immobilizer systems. Tillich and Wójcik [TW12a] analyze one of the few open car immobilizer protocols released by Atmel. Despite their focus on revealing security issues, their work serves as an example of how commercial immobilizers work. Communication between key fobs and a reader module implemented in the vehicle works by utilizing a challenge-response protocol over RFID. While most car immobilizer systems are based on proprietary protocols [TW12a], Busold et al. [Bus+13] present an open security framework, which utilizes smartphones. Vehicle owners are equipped with NFC-enabled smartphones to authenticate against compatible HSMs employed in cars. Replacing dedicated transponders, smartphones can be used for a more flexible authentication system that allows enhanced features, such as delegation of access to guest drivers for a specific time. Besides the normal operating system, a Trusted Execution Environment (TrEE) is required on the mobile device. Access tokens to unlock the car are generated by the manufacturer with an Authenticate-then-Encrypt scheme using an authentication secret and encryption key pre-deployed on the car.

Similar to Busold et al. [Bus+13], we utilize an NFC-enabled HSM to store the keys securely. In contrast to the above-mentioned immobilizer system, though, we utilize the smartphone as a trusted interim device to perform registration and key generation/deployment instead of serving as a key replacement. As mentioned above, we neither rely on pre-deployed keys nor a specialized TrEE. Since exact hardware requirements are beyond the scope of this thesis, we refer to existing comparisons of available HSMs for vehicles [WG12].

¹²<http://www.car-2-car.org>

Instead of utilizing the smartphone as a key replacement, it serves as a trusted interim device to perform registration, key generation, and key deployment. After a successful key deployment, the responsibility to store the key securely is shifted to the HSM and our DTN stack, deployed for V2C communication. Relying on already established standards, the remaining Smartphone-to-Cloud communication is secured using OAuth.

5.2 Web Service Security

In addition to providing necessary functionality for managing charging stations and scheduling electric vehicles, V-Charge’s back-end server is responsible for user and session management. We based our security architecture on Glassfish’s built-in support for the Java EE Security Specification [Orab], in those cases where it matched our requirements. Requirements going beyond built-in Java EE support are evaluated individually and implemented using additional libraries.

Wherever possible, our architecture should be based on already established standards to ensure that all utilized methods are well-studied and reviewed. Thus, our underlying security principles and cryptographic techniques must comply with state-of-the-art practices in secure web development, described by OWASP¹³. From a usability perspective, the web interface should provide login functionality that integrates seamlessly into the existing layout. It should provide a logout functionality and methods for managing accounts, including the option to change the password. Because the web interface could be used from terminals or publicly accessible computers, the session should be time limited and the user should be logged out automatically on time expiration. Short expiration times prevent leaving access open to subsequent users of the terminal.

OWASP provides a top 10 selection of most critical web application security risks, which is a highly recognized and up-to-date collection [OWAb]. The risks are sorted by their current importance (A1–A10), chosen according to data collection, which is based on 8 datasets from 7 firms specializing in application security as of 2013. In this thesis, we concentrate on a security assessment of V-Charge’s web service considering the OWASP Top 10. Java EE already provides protection against most OWASP’s attack targets when properly utilized, e.g., protection against Cross-Site Request Forgery (CSRF) and SQL injection attacks. According to the scope of this thesis, we improved the implementation and solidified it especially against the following attack scenarios:

Broken Authentication and Session Management (OWASP A2). User sessions identified by session IDs, often stored as cookies or appended to URLs providing a stateful connection, are a promising attack target, as it is feasible to impersonate other users. Using Cross-site Scripting (XSS) attacks, insecure/compromised connections, or Trojans infecting the user’s computer, these session IDs can be extracted or calculated by an attacker. Due to the severe consequences of impersonation attacks in V-Charge, e.g., controlling vehicles to a certain degree, special care is required to ensure confidentiality of session IDs.

Attacking Transport Layer Security (part of OWASP A5). The connection established between web browsers and our web service could be eavesdropped on or actively forwarded by Man-in-the-Middle (MitM) attackers. Attack scenarios include open Wifi connections, e.g., passive attackers in Internet cafés or airports. While passive attacks can be prevented by key exchange protocols and encryption, active attacks intercepting, modifying, and forwarding data must be prevented by authenticating the server connection.

Database Leak (part of OWASP A2, A6). A database leakage could happen through implementation issues in the web service itself: database dumps or by remote exploits and privilege escalation techniques

¹³<https://www.owasp.org>

in the webserver/database software. While it is assumed that compromising in-memory server secrets is difficult, sensitive data stored in the database should be protected by cryptographic primitives that even prohibit or withstand database leakage.

5.2.1 Authentication and Session Management

Glassfish provides built-in support for realms consisting of user collections and their corresponding groups. To create relationships between users and other entities, e.g., vehicles, a user table exists in our web service, which is defined by the Enterprise JavaBeans (EJB) entity `Customer` [Tim12].

Previously, user authentication in V-Charge was done with HTTP Basic Authentication [Fra+99] for both the web service and the RESTful JSON API. Consequently, the user interaction in the web service needed for authentication is handled by the browser, usually implemented by a client-side pop-up dialog. This method delegates input handling of usernames/passwords to the browser application itself, which protects against injection attacks. Besides this advantage, it does not provide session management that can be controlled properly by the application itself. The browser stays logged in until its cache is cleared, which is done rarely by typical users. The vulnerability of injecting JavaScript code into the web service can only be exploited while the user is logged in. Due to the browser's feature of remembering credentials, the user is vulnerable even when V-Charge website is currently not opened. Moreover, a logout mechanism cannot be provided by the web service.

To overcome these issues and to integrate the Login into V-Charge's design, we implemented a HTTP form based Login. It was realized by using built-in features of Java EE and Glassfish. To provide perfect integration with our existing layout, we used Java Server Faces and the programmatic login using the `HttpServletRequest.login()` method provided by Servlet 3.0 compliant Java EE servers. Upon successful login, a `CustomerSession` object is created and added to the in-memory session map of the current context for fast retrieval of customer properties. It works as a view on the actual `Customer` EJB entity, inheriting most of its properties while omitting the hashed password. This is done because a) the password is not needed in-memory after login and b) leaking the hashed password by exploiting the web application needs to be prevented. A session is configured to timeout after being idle for 15 minutes. A corresponding session ID is stored in a session cookie provided by Glassfish [Orab], which is considered best practice for managing sessions [OWAa]. Appending session IDs to URLs could lead to revealing them when sharing URLs over separate communication channels, e.g., instant messengers.

The actual protection of functionality in V-Charge's web service is done by adding Java EE security annotations to selected methods and constraining certain web pages to our default `customer` role [Orab].

5.2.2 Transport Layer Security

To secure the connection between web browser and server, TLS is deployed. Its predecessor SSL has been available since 1994 and has been improved up to its current 3.0 version [FKK11]. SSL 3.0 and its successor TLS are implemented in all major web browsers and are now the most widely deployed standards for secure communication on top of TCP. In the last two years several weaknesses of TLS 1.0 have been discovered and successful attacks have been shown. The first attack called "Browser Exploit Against SSL/TLS" (BEAST) exploits vulnerabilities in the utilized Cipher Block Chaining (CBC) Mode in TLS 1.0 [RD11]. The last shown attacks are called "Compression Ratio Info-leak Made Easy" (CRIME) and "Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext" (BREACH, backronym) [RD12; GHP13]. These recent attacks exploit the fact that compression algorithms, which are used in conjunction with TLS, produce smaller ciphertexts when information is repeated within the plaintext. As attackers are often able to forge

or inject plaintext by changing parts of the input, e.g., via web forms, the length of the produced ciphertext can be used as an oracle to brute force through possible plaintext injections to recover small secrets, e.g., session IDs. Other proposed attacks exploit weaknesses in RC4 encryption algorithm or in padding schemes, e.g., as utilized by the “Lucky Thirteen” attack [AP13].

While TLS 1.1 and 1.2 mitigate these attacks, they are not implemented widely enough¹⁴ to require them for connections to our servers. Additional countermeasures taken by browser vendors and utilized libraries, e.g., NSS or SChannel, provide sufficient security for TLS 1.0, while being backward compatible. For example “1/n-1 record splitting” has been implemented in SChannel to protect against BEAST, which is described in the Microsoft Security Bulletin MS12-006 [Mic] and released as a cumulative security update for Internet Explorer (MS11-099). Thus, we propose to deploy TLS 1.0 with an implementation included in a recent version of a Java EE compliant web server, i.e., the newest Glassfish server with its dependencies.

To allow browsers to verify if a TLS connection is established to the originally requested server, i.e., no MitM attacker is present, X.509 certificates can be used. Certificate Authorities (CAs) authenticate that someone is indeed the owner of a particular domain and issue a certificate to be used for this server. This is done by generating a private/public key pair and sending a Certificate Signing Request (CSR) that includes the public key to a recognized CA. The CA validates the ownership using a second channel, e.g., email or other proofs of domain ownership. Any following TLS handshake would be signed using the generated private/public key pair and contains the certificate for this public key. Thus, MitM attackers would a) need to forge a valid signature to impersonate the server, which is considered impossible under the assumption that no utilized cryptographic primitives are broken or b) need to obtain a valid certificate for the targeted domain. Specific dishonest CAs have issued certificates to misbehaving entities and several security breaches have occurred during the last several years, which has raised concerns regarding this centralized model [CAc13]. CA Transparency [LLK13] and various decentralized approaches [WAP08; Mar13] are discussed to improve the situation.

There is currently no practical alternative to standard X.509 certificates when serving web services for general purpose browsers, as no final decision is made by browser vendors regarding decentralized approaches. Thus, until better standards are adopted, we propose the utilization of a certificate issued by a traditional CA to allow verification of the server’s public key by general all major browsers.

We configured the V-Charge service for usage of a X.509 certificate and configure the web service according to the requirements. For our current development environment, a self-created CA was used. To facilitate a production environment, certificates signed by common CAs should be employed.

5.2.3 Password Protection

To protect against leakage of passwords, even when parts of the database are compromised, only the hash of a password is stored. This allows for authentication by comparing the stored hash with a calculated hash of the entered password. Attackers trying to recover passwords from hashes need to utilize a brute-force algorithm to go through all possible passwords and build a lookup table. Therefore, the complexity of such attacks is based on the complexity of the hash function itself, multiplied with the number of combinations from a chosen character alphabet of a specific length. Pre-calculated lookup tables for common hash functions and character alphabets are available on the Internet. To prevent the usage of these tables, passwords are concatenated with salts, which are stored per website, to make calculation unique for this particular application. With growing CPU power available as cloud services, finding hash collisions has become feasible in a small amount of time and has low costs when dealing with non-salted passwords hashed with only one iteration [Per09].

¹⁴State of TLS 1.2 in Firefox: https://bugzilla.mozilla.org/show_bug.cgi?id=480514

The National Institute of Standards and Technology (NIST) published the Password-Based Key Derivation 2 (PBKDF2) in NIST SP 800-132 [Tur+10], which is also defined in RFC 2898 [Kal00], to provide a standard on how to store passwords securely. The algorithm was adopted by several protocols and implementations, e.g., WPA, WPA2, Linux Unified Key Setup (LUKS) for key derivation [Com+07; Fru11].

Definition (PBKDF2). *It is based on a pseudo-random function $\text{prf}()$, e.g., SHA-512, to derive keys mapping them to the length of this underlying function. Important parameters to execute PBKDF2 are the password p , salt s , and an iteration count n . Per octet-block i of the derived key, a block T_i is generated by*

$$T_i = U_1 \oplus U_2 \oplus \dots \oplus U_n,$$

where $U_1, \dots, U_n$ are calculated as

$$\begin{aligned} U_1 &= \text{prf}(p \parallel s \parallel i) \\ U_2 &= \text{prf}(p \parallel U_1) \\ &\vdots \\ U_n &= \text{prf}(p \parallel U_{n-1}). \end{aligned}$$

The resulting derived key k is the concatenated result of all blocks $k = T_1 \parallel T_2 \parallel \dots \parallel T_l$, where l is based on the desired length of the derived key.

While we selected PBKDF2 due to its certification as a Federal Information Processing Standard (FIPS) [Tur+10], bcrypt [PM99] and scrypt [Per09; PJ13] are available alternatives. The scrypt algorithm is designed to make brute-force attacks, which work by guessing passwords up to a specific length, harder by using sequential memory-hard functions and represents current state-of-art techniques in password hashing. Recent password database leaks shows that key derivation functions are not adopted widely and need more attention from the cryptographic community. For instance, the password database of LinkedIn was leaked and revealed on the Internet [Lin]. The passwords were stored as unsalted SHA-1 hashes, which made them easy to recover by guessing password combinations up to a specific length [Fic]. To push the motivation for research on better password derivation functions, The Password Hashing Competition¹⁵ was started in 2013. It is operated like similar cryptographic competitions, e.g., NIST's competition to standardize the block-cipher AES. It aims to provide better understanding of underlying cryptographic principles and encourages the cooperation between researchers to design a new password hashing algorithms. Selected algorithms will be finally presented in the second quarter of 2015.

Unfortunately, Glassfish JDBC Realm does not support any of those password-based key derivation functions. It only supports MessageDigest algorithms included in Java SE 7 [Oraa] to be used directly without salting and hashing iterations, e.g., hash algorithms like SHA-512. To store passwords as PBKDF2 derived keys, a custom JDBC Realm implementation is needed. A detailed tutorial on how to implement this realm can be found in *GlassFish JDBC Security with Salted Passwords on MySQL* [Eis12].

The actual deployment of this method is postponed for a production version of V-Charge, as the implementation of a JDBC Realm complicates development. When implemented, every Java EE server running V-Charge needs to use this custom JDBC Realm.

5.3 Smartphone-to-Cloud Security

Client Applications, especially the V-Charge Android client, impose additional and partially different requirements regarding security compared to the web service. As well as in our web service, authentication

¹⁵<https://password-hashing.net>

techniques in client applications should be based on established and well-studied standards. Usability is a key requirement in mobile applications, which heavily depends on security implementations. The hassle of requiring the user to repeatedly login before using it should be prevented. When implementing unlimited user sessions for client applications, revocation of access is an important new requirement. Considering a stolen smartphone, the user should be able to revoke access from this device in a convenient manner, before a thief can unwarrantably use V-Charge. The actual V-Charge password should never be stored on the mobile device, while providing revocable user sessions per device. Conclusively, the requirements are as follows:

- Standardized security frameworks
- Focus on usability
- Easy device revocation.

Besides those requirements, we assume that V-Charge’s infrastructure and all corresponding client applications are controlled by the V-Charge project itself. As no third-party API is planned, authorization is solely implemented for internal use within the V-Charge project.

We chose to incorporate the OAuth 2.0 standard [Har12], which is a basic specification described in Section 5.3.1. Although its name is derived from authentication [Ham11], it provides authorization for client applications. In our specific scenario, it authorizes our smartphone application to access the central V-Charge service. Applications can retrieve tokens to access services on behalf of a user. Access is provided without including username/password combination on every request.

OAuth 2.0 provides only a basic specification of an Authorization framework and leaves many details and extensions open to the provider. Due to V-Charge’s centralized infrastructure and our specific attack scenarios, which are further described in Section 5.6.2, we implemented a well-defined subset of the OAuth 2.0 standard (cf. Section 5.3.2). Along with our use case, this adapted OAuth flow is applicable in a wide range of centralized scenarios, as described in the following. In the V-Charge project, one central server provides authorization as well as resources, while all client applications are controlled by the project itself. Section 5.3.3 describes the details of our proposed client implementation of this standard, considering Android’s underlying system security.

Because client applications of V-Charge always connect to the same servers, we can work around problems arising due to centralized CAs. In contrast to general purpose web browsers, we know in advance which public key to expect from a connection. Hence, pre-deployed public keys are utilized, as described in Section 5.3.4.

5.3.1 OAuth Specification

The OAuth 2.0 Framework, including its components and message syntax, is defined in RFC 6749 [Har12], while RFC 6750 [JH12] describes the usage of a Bearer Token as an extension to the HTTP authorization header. Related security considerations can be found in RFC 6819 [LMH13]. Following RFC 6749, the OAuth 2.0 standard defines the separated entities client, resource owner, authorization server, and resource server. A typical authorization flow works as follows: a client requests access from the resource owner, which is available as a web service. Access is usually granted by redirecting to the resource owners’ website and logging into this trusted service. On successful login, the owner redirects back to the requesting client with the authorization code. This authorization code can be used to query the authorization server for an access token, which is the actual grant for access. Finally, the access token can be used to access the resource server using its API. This typical OAuth flow is designed for clients implemented as web services, such as the

OAuth provider itself, e.g., Twitpic¹⁶ requesting access to resources from Twitter¹⁷. Including authorization codes, four authorization grants are defined:

Authorization code. A code is issued by the resource owner, which can then be exchanged for an access token by querying the authorization server.

Implicit Authorization. This grant type can be used to issue access tokens directly without an intermediate authorization code, proposed for clients implemented with scripting languages, such as JavaScript as it can be used without client verification.

Resource owner password credentials Retrieve an access token by directly providing user/password combinations encoded to be part of the URL.

Client credentials. Client credentials allow requesting specific resources without user interaction, clearly restricted to non-user resources.

All grant types except client credentials can optionally be used with refresh tokens. Refresh tokens are long-living access grants that are stored in client applications instead of passwords and can be used to renew expired access tokens by querying the authorization server. To restrict access to specific client applications, every request for tokens must include a client ID, previously registered at the OAuth provider. For example, before any third-party web service can request access tokens for Twitter resources, the service itself needs to be registered using their developer interface¹⁸.

A typical OAuth flow is based on redirecting users between an OAuth provider and other HTTP services acting as a consumer [Har12]. Authorization codes are issued by the resource owner before redirecting the user back to the client. These authorization codes can then be used as a grant type to get authorization with access tokens via the authorization server. This grant type is designed to shift the user's trust regarding the login away from third-party services to the OAuth provider.

In contrast to typical social HTTP services, e.g., Twitter, V-Charge is a self-contained system with no public API for third-party clients. This means that all client applications are developed by the project itself, maintaining control and trust regarding the implementation of the actual login process (authentication). Thus, instead of using an authorization code based grant type, we allow direct authentication with user credentials. The authorization server and the resource server are both are part of our central V-Charge service and run on the same machine. Further security considerations regarding the avoided OAuth redirection flow implemented by most social providers can be found in Section 5.3.1.

OAuth Exploits and Shortcomings Most attacks against OAuth result from implementation bugs of OAuth's most common grant type, 'authorization_code'. OAuth providers serve a HTML-based login form (authorization endpoint) that can be used from OAuth consumers to request an authorization code. This is done by redirecting from the consumer to the provider's login page via a browser's redirect method. After login and accepting access to certain scopes, the browser is redirected back to the consumer's web service containing an *access_token* if "implicit flow" is utilized. The consuming web service can then use this token to access certain resources of the provider's resource server, which can be a different entity as the authorization server. The most fundamental problem here is that all parameters and return values are passed by an URL; there is typically no direct communication between an OAuth provider and consumer. This makes the protocol implementations prone to attacks originating from common XSSs or CSRFs attacks (e.g., [BBM12; Hom13])). Because the URL of the consumer is passed as a parameter named *redirect_uri* to

¹⁶<http://www.twitpic.com>

¹⁷<https://www.twitter.com>

¹⁸<https://dev.twitter.com>

Your Account

Authorized Applications/Services **Change Password**

You allowed these applications or services to use V-Charge with your account. You can revoke the authorization by right-clicking and choosing 'revoke'.

Application/Service	Device	Issued at	Last access
V-Charge-Android	Dalvik/1.6.0 (Linux; U; Android 4.3; Nexus 4 Build/JSS15J)	26.08.2013 11:14	26.08.2013 11:14
V-Charge-Android	Dalvik/1.6.0 (Linux; U; Android 4.2.2; Nexus 7 Build/JDQ39E)	26.08.2013 11:17	26.08.2013 11:17

✕ Revoke Access

Figure 29: Revocation of device’s access via V-Charge’s web interface

the authorization endpoint, it is especially vulnerable to modification. This allows attackers to redirect the requested token to their own malicious web service.

A second widespread problem is the misunderstanding on certain aspects of OAuth 2.0. After OAuth 1.0 was proposed, several proprietary extensions were developed, leading to the successor OAuth 2.0, standardized by IETF. Currently, the workgroup¹⁹ lists 5 approved RFCs, 6 active drafts, and 15 related active documents. The 71-page RFC 6819 [LMH13] describes OAuth’s threat model and security considerations. To pick out one common misconception, the *client_id* and *client_secret* is sometimes misused as a client application authentication. Native applications contain these secrets hardcoded in their binary, which makes it possible to extract them. RFC 6819, Section 4.1.1 explicitly explains this threat and provides several countermeasures, such as “Don’t issue secrets to public clients or clients with inappropriate security policy” [LMH13] or “Revoke client secrets” [LMH13]. Twitter has decided to revoke compromised client secrets and were doing so until their own official Twitter client was decompiled and its client secret recovered [Pau]. Revoking this secret would prevent users from accessing Twitter with their own official client. This leakage was especially problematic as their own client secret allows access to more resources and scopes than usually registered clients. Thus, issuing different client secrets to allow for different access levels is not practical for native clients due to the fact that consumers can extract those secrets and circumvent access control by using them for their own purposes.

5.3.2 OAuth Server

The OAuth server should only implement the specific subset of OAuth functionality required for V-Charge, while omitting its problematic parts (cf. Section 5.3.1). As performed by providers like Twitter or Google, only the ‘password’ and ‘refresh_token’ grant types are required to permit login from completely trusted clients.

V-Charge’s OAuth implementation is solely based on a token endpoint and allows the grant type ‘password’ to retrieve a *refresh_token* and an *access_token* on first login. The grant type ‘refresh_token’ serves to query a new *access_token*, which is valid for one hour, using the long-living *refresh_token*.

Revocation of a device’s access to the V-Charge service is possible by logging into V-Charge’s web interface and deleting the *refresh_token* of the lost/stolen devices (cf. Figure 29). Revocation is only possible using the secret password of a user, which is not stored on the device itself. Changing a user password is only

¹⁹<https://tools.ietf.org/wg/oauth>

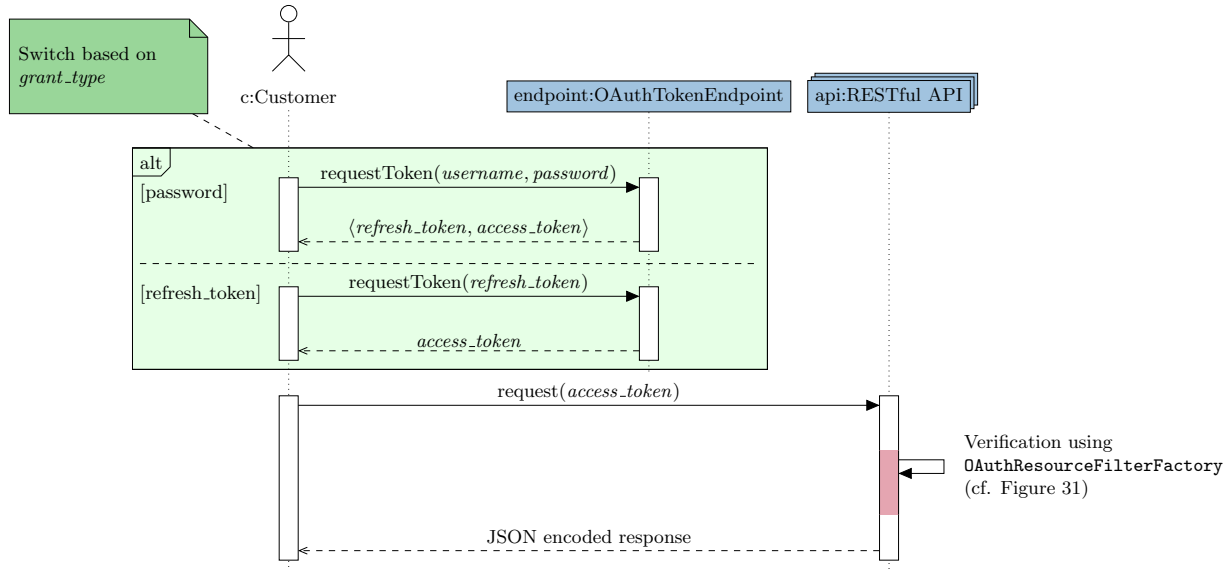


Figure 30: OAuth 2.0 authorization flow to query V-Charge's API

possible by entering the old password, not by using a *refresh_token*, making it impossible for a device thief to overtake a V-Charge account as a whole.

Following Figure 30, querying V-Charge's RESTful API using our OAuth authorization flow works as follows:

1. On first use, the user of the client application needs to authenticate against the token endpoint by sending an access token request with password grant type, including username and password. *client_id* and *client_secret* are specific values defined per developed client application and included in every request to our token endpoint. While they are not secrets in a cryptographic sense, they are defined by the RFC and can be used to differentiate between benign client implementations.

```

1 POST /VCharge/oauth/token HTTP/1.1
2 Content-Type: application/x-www-form-urlencoded
3 User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.2.2; Build/JB_MR1.1)
4
5 grant_type=password&username=test@v-charge.eu&password=vcharge
6   &client_id=V-Charge-Android
7   &client_secret=2YotnFZFEjr1zCsicMWpAA

```

2. The client application receives an *access_token* and a long-living *refresh_token*.

```

1 HTTP/1.1 200 OK
2 Content-Type: application/json
3
4 {
5   "access_token": "382a53dabc7e3b57b41cdabc6022bacb2",
6   "expires_in": 3600,
7   "refresh_token": "6e9554c8351c3b25c071972bd3076f0e",
8 }

```

3. To access V-Charge’s API, the query includes the *access_token* using the Bearer header field. A typical request, e.g., for registering the client with Google Cloud Messaging (GCM), looks like the following.

```
1 POST /VCharge/rest/customer/register-gcm HTTP/1.1
2 Authorization: Bearer 382a53dabc7e3b57b41cdbc6022bacb2
3 Content-Type: application/x-www-form-urlencoded
4 User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.2.2; Build/JB.MR1.1)
5
6 gcmId=APA91b
```

4. After expiration of the current *access_token*, a new one can be requested by using the ‘refresh_token’ grant type and the previously stored *refresh_token*.

```
1 POST /VCharge/oauth/token HTTP/1.1
2 Content-Type: application/x-www-form-urlencoded
3 User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.2.2; Build/JB.MR1.1)
4
5 grant_type=refresh_token
6 &refresh_token=6e9554c8351c3b25c071972bd3076f0e
7 &client_id=V-Charge-Android
8 &client_secret=2YotnFZFEjr1zCsicMWpAA
```

5. The client application receives a new *access_token*, again valid for one hour.

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json
3
4 {
5   "access_token": "7e716d0e702df0505fc72e2b89467910",
6   "expires_in": 3600,
7 }
```

We utilized Apache Oltu²⁰ for server and client implementations. The project provides Java libraries for generating, parsing, and validating OAuth 2.0 compliant requests and responses. In addition to V-Charge’s existing RESTful API classes, which consume URL-encoded POST requests and produce JSON-encoded responses, we implemented `OAuthTokenEndpoint`. This endpoint employs the authorization grant types ‘password’ and ‘refresh_token’ only. Tables 4 and 5 describe all utilized parameters and return values for the OAuth token endpoint.

Due to the limited number of V-Charge client applications, there is no need to allow dynamic registration of new clients. Thus, a list of allowed client ids and corresponding client secrets is hardcoded in our web service and token requests are checked against this list to verify the requesting client. Refresh and access tokens are stored in the database, defined by an EJB entity `OAuthToken`. Every row contains a *refresh_token* and its corresponding last *access_token*, validity periods of these tokens, and a foreign key to V-Charge’s `Customer` table. An *access_token* is configured to be valid for a period of one hour, when a new *access_token* can be requested by sending a request to `OAuthTokenEndpoint` including the long-living *refresh_token*.

Implementation details of our OAuth flow are depicted in Figure 31. V-Charge’s RESTful API is protected by resource filters, generated by the factory filter in the case presented herein. Every API method annotated with `@OAuthRolesAllowed("regular_customer")`, an annotation we introduced to declare OAuth protected

²⁰<http://oltu.apache.org>

Table 4: Input parameters of OAuth token endpoint at URL `/oauth/token`

Parameter	Type	Comment
<i>client_id</i>	String	Client application identifier
<i>client_secret</i>	String	Secret value corresponding to <i>client_id</i>
<i>grant_type</i>	String	Possible values: 'password': credential based authentication 'refresh_token': request new tokens
Grant Type: 'password'		
<i>username</i>	String	
<i>password</i>	String	
Grant Type: 'refresh_token'		
<i>refresh_token</i>	String	long-living token

Table 5: Return values of OAuth token endpoint at URL `/oauth/token`

Grant Type	Return Value	Type	Comment
'password'	<i>access_token</i>	String	
	<i>refresh_token</i>	String	long-living token
'refresh_token'	<i>access_token</i>	String	

functions, triggers our filter factory. This factory confirms the validity of a given *access_token* included in requests and creates a custom `SecurityContext` for this session holding properties like *username*.

5.3.3 OAuth Client

V-Charge's use case scenario is highly centered around mobile devices to manage drop-off and pick-up of vehicles, request reservations, and the query status of charging vehicles. An Android application was developed previously to carry out those functionalities while focusing on usability aspects. As discussed in the introduction of this section, our security implementation should be mostly transparent to the users, i.e., should not restrict its usage. The following goals were formed and then made protocols:

User Authentication. Login-based authentication on first-time use of V-Charge or on adding new V-Charge accounts to the application.

Client Authorization. Long-living authorization of the client application to overcome the hassle of manual login on every usage.

Multi-user capability. Allow unlimited accounts per application and the possibility to switch between them.

Secure Implementation. The implementation should follow best practices regarding account management, e.g., not storing sensitive tokens on the SD card.

To benefit from Android's security model, which provides sandboxed applications and a sophisticated permission management [Fel+11], our solution uses Android's provided account management system where

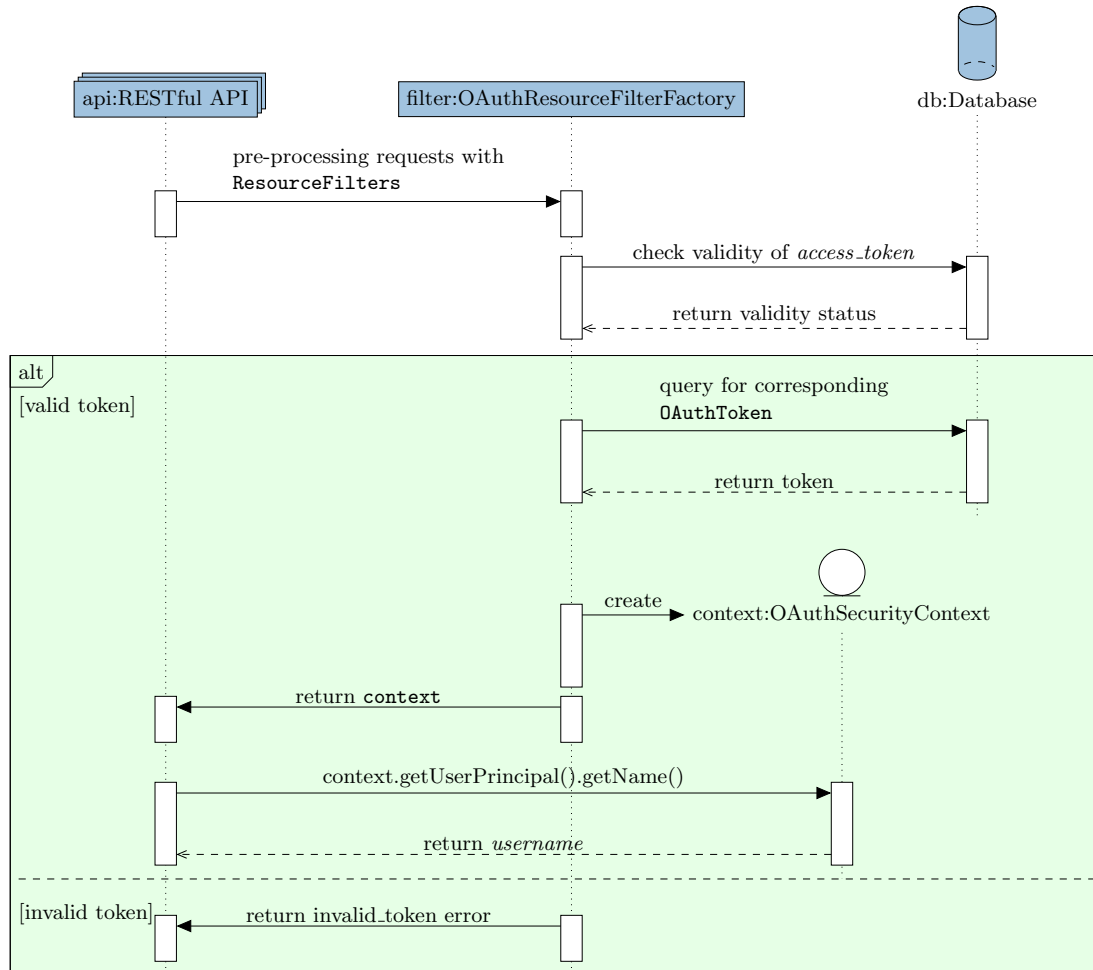


Figure 31: Verification of *access_token* and creation of custom **SecurityContext**

possible. Parsing OAuth responses and generating OAuth requests are provided by the Apache Oltu Client library. Similar to Google's own OAuth provider [Goo], our setup leaves storing authentication tokens to Android's **AccountManager**²¹.

On first use, the login screen is shown to enter the *username* and *password*. These credentials are used to request an *access_token* and *refresh_token* from our token endpoint. An Android **Account** object is created containing the *username* and the *refresh_token*. The *access_token* is also saved as an authentication token for this account. The *password* is never saved on the device. On every request to the RESTful API, the last saved *access_token* is retrieved from Android's storage and utilized for authorization. If the V-Charge server responds with an error indicating that the token is no longer valid, a new *access_token* is queried using the long-living *refresh_token*.

²¹<http://developer.android.com/reference/android/accounts/AccountManager.html>

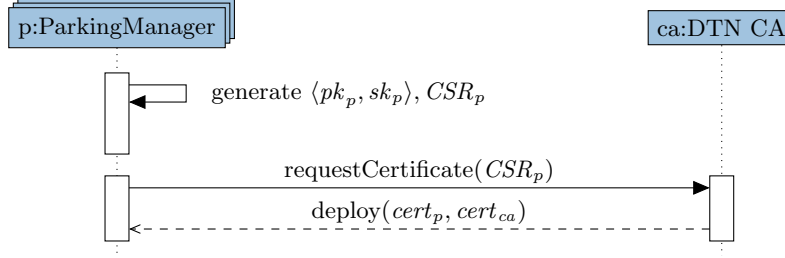


Figure 32: Deployment of certificates issued for local Parking Managers

5.3.4 Transport Layer Security for Client Applications

In V-Charge’s Android application, we implemented certificate checking without using a list of trusted CAs like in TLS for Browser-to-Server communication. On Android, a list of trusted CAs is provided by the operating system itself, while applications are able to deviate from that by instituting their own TLS validation. As discussed in Section 5.2.2, currently many problems exist with CAs. Therefore, instead of requesting a signed certificate from one of Android’s supplied CAs, we created our own V-Charge CA to secure the RESTful API solely. This self-created CA certificate is included in our Android client and used exclusively, which is called certificate pinning. It allows us to shift the trust from outsourced CA management to our smartphone application by including the solely used public key in the application package. Conclusively, all secure connections are verified with this key only.

5.4 Vehicle-to-Cloud Security

The V-Charge service communicating with the vehicles consists of these central components:

ParkingManager (PM). Placed at each parking station communicating securely with vehicles via DTN. PMs themselves are connected to central V-Charge backend servers.

CustomerManager. The CustomerManager [TW12b] provides registration functionality and all necessary service methods (pick-up, drop-off, status checking) via RESTful Web services that are being interfaced by the V-Charge smartphone application. All methods of the interface are protected by the OAuth 2.0 standard.

OAuth Authorization Server. A central OAuth Authorization Server handles all OAuth sessions and provides verification methods. The connection between smartphone and the RESTful Web services itself is secured by TLS.

DTN CA. The DTN CA manages certificates deployed to all participants in the DTN, i.e., PMs and vehicles.

As depicted exemplary in Figure 32, a public/private key pair $\langle pk_p, sk_p \rangle$ is generated by each PM p . The DTN CA processes p ’s CSR after it physically receives it from p and sends back a certificate $cert_p$ signed with the CA’s secret key sk_{ca} . After $cert_p$ has been deployed on p , vehicles equipped with the DTN CA’s root certificate $cert_{ca}$ can verify if bundles are affiliated to a PM (i.e., signed with sk_p) and are thus allowed to issue control commands (i.e., drop-off, pick-up). Contrary to vehicle certificates, PM certificates installed during their physical deployment contain a ‘ParkingManager’ flag issued by our CA to allow differentiation between inter-vehicle bundles and bundles coming from servers acting as PMs. The connection between PM servers and central V-Charge services, such as DTN CA, is secured by a VPN.

5.4.1 Security in Delay-Tolerant Networks

RFC 6257 [Sym+11] describes ciphersuites, Security Blocks, security policies, and how to deal with fragmented bundles. Bundle Authentication Blocks (BABs) are defined to provide hop-by-hop integrity and authenticity, which can be verified by every relaying node along the (multi-hop) path to the destination. The ciphersuite `BAB_HMAC` allows HMAC-authenticated bundles; end-to-end security is achieved by using the Payload Integrity Block (PIB) for integrity and authenticity and the Payload Confidentiality Block (PCB) for confidentiality (also providing integrity by AES in GCM). The ciphersuite `PIB_RSA_SHA256` is designed for signing bundles using RSA with SHA-256 as the hashing algorithm, while `PCB_RSA_AES128_PAYLOAD_PIB` can be used to encrypt PIBs and the payload. Decryption and verification is typically performed by the destination node.

In our scenario, every participating node needs to be able to verify if an incoming bundle is affiliated with V-Charge. Our goal is to restrict DTN communication to nodes participating in V-Charge. Thus, bundles are always encrypted first using the PCB, and then signed using PIB to achieve end-to-end security. We configure IBR-DTN to accept/send PIB-signed and PCB-encrypted bundles only. In order to thwart black/gray hole attacks, we facilitate hop-by-hop verification. To this end, BABs only support authentication, while PIBs allow signatures for verification using public keys. However, instead of using BABs or PIBs to verify relayed bundles, we utilize IBR-DTN’s support for TLS on the TCP convergence layer. A major advantage of TLS over BABs is that, if a node gets compromised, only its certificate needs to be revoked, instead of having to replace the symmetric BAB key across all nodes. Moreover, when bundles are relayed between nodes, TLS allows authenticated acknowledgments when peers accept bundles sent by neighboring nodes in contrast to the sole use of PIBs.

5.4.2 Certificate Revocation

Certificate Revocation in DTNs is one of the interesting problems of DTN security. Naïve implementations require an additional check whether the certificate is currently valid when relaying/receiving a bundle. This can be done by querying a central database or by distributing Revocation Lists, whereas both solutions are problematic in networks with high delays and disruptions.

However, in the V-Charge scenario, vehicles periodically are in physical proximity to PMs or other vehicles, from which they are able to receive Revocation Lists via DTN. We propose to provide Revocation Lists as bundles sent by the local PMs and signed by the central DTN CA to allow verification using $cert_{ca}$. These Revocation List bundles are sent and updated with period δ and have a lifetime of $\delta + x$, where x defines a transition time that allows new revocation lists to invalidate old ones. Because all vehicles participating in V-Charge are relaying and storing currently valid Bundle Revocation Lists while parking, these lists can be kept as “Floating Content” [Ott+11]. Thus, arriving vehicles immediately receive currently valid lists either from the PM or from other parked vehicles. Not only is the PM holding current lists, it becomes much more difficult for attackers to execute blackhole attacks to prevent distribution of current revocation lists. By verifying the signature, the lists’ validity can be checked.

5.4.3 Implementation

While the proposed certificate-based security would provide the most flexible use for typical ITS, our deployed DTN implementation, IBR-DTN, only supports encryption/verification using locally stored keys. In IBR-DTN, public keys for all communicating entities are stored on every participating node and used to encrypt outgoing PCBs and verify incoming PIBs. IBR-DTN assumes a file-based key deployment, i.e., every node is required to have public keys from all other nodes in its storage. Currently, this deployment needs to be done

manually. While this thesis covers all required fundamentals for such certificate-based security deployment, their actual implementation is out of its scope and considered future work. For current V-Charge testbeds, we utilized a simplified model using available techniques.

V-Charge’s main web service allows exporting a set of all public keys currently stored in its database in a format suitable for IBR-DTN (Base64-encoded PEM format). Consecutively, this set is deployed on every PM in V-Charge’s testbed. After successfully linking PMs to the central backend server via a secure connection like VPN, this can be further automated by periodically updating PMs about newly registered vehicles. Vehicles are equipped with a set of public keys for all participating PMs to permit sending PCBs and allow verification of incoming PIBs. IBR-DTN is configured to require signed and encrypted incoming bundles, therefore accepting only bundles sent by PMs as endpoint bundles, which are processed further. This is required to restrict the control of vehicles to PMs only. The setup is sufficient for development scenarios. It is not suited for production scenarios because the addition of new PMs results in redeployment of all public key sets on all vehicles.

5.5 Vehicle Registration and Key Deployment

Vehicle registration and its corresponding key deployment should integrate in V-Charge’s existing scenario. Besides a framework focusing on usability for end users, the primary objectives of our key deployment process design are as follows:

Minimum of trust in central authorities. Private keys should always be generated and managed by the user itself; thus, there is no need to trust a service to handle key generation correctly, e.g., without storing or even worse leaking them. This includes the requirement that users have full control over the DTN stack and key deployment.

Feasible security audits due to open protocol design. All cryptographic operations should be auditable by sophisticated users, i.e., they are part of an open protocol.

Our process to register a new vehicle for usage with the V-Charge system is designed to be executed on off-the-shelf mobile devices that already provide a platform sufficiently trusted by its owner. Because the vehicle’s DTN stack is configured to only send/receive signed and encrypted bundles, it denies all incoming connections in its delivery condition, as neither keys nor certificates are pre-deployed. After successfully finishing the registration using our smartphone application by entering user credentials and additional information, a manual identification step, e.g., using Postident²², can finalize the registration process.

5.5.1 Proposed Registration and Key Deployment Process

Following Figure 33, the user is now able to log into the V-Charge system using the chosen $\langle username, password \rangle$ combination. This is done by requesting a long-living *refresh_token* from our OAuth Authorization Server and sending the credentials over an TLS connection (cf. Section 5.2.2). Additionally, the device d also obtains an *access_token* valid for one hour. As described in RFC 6749 [Har12], the *refresh_token* can be used to retrieve a new *access_token*.

Vehicles are uniquely identified by their Vehicle Identification Number *VIN*. The *VIN* of newly assembled vehicles, which come equipped with the required communication technology, can be directly added to the V-Charge database alongside the owner on purchase. To register refitted or resold vehicles for an account,

²²<http://www.deutschepost.de/dpag?xmlFile=1015469>

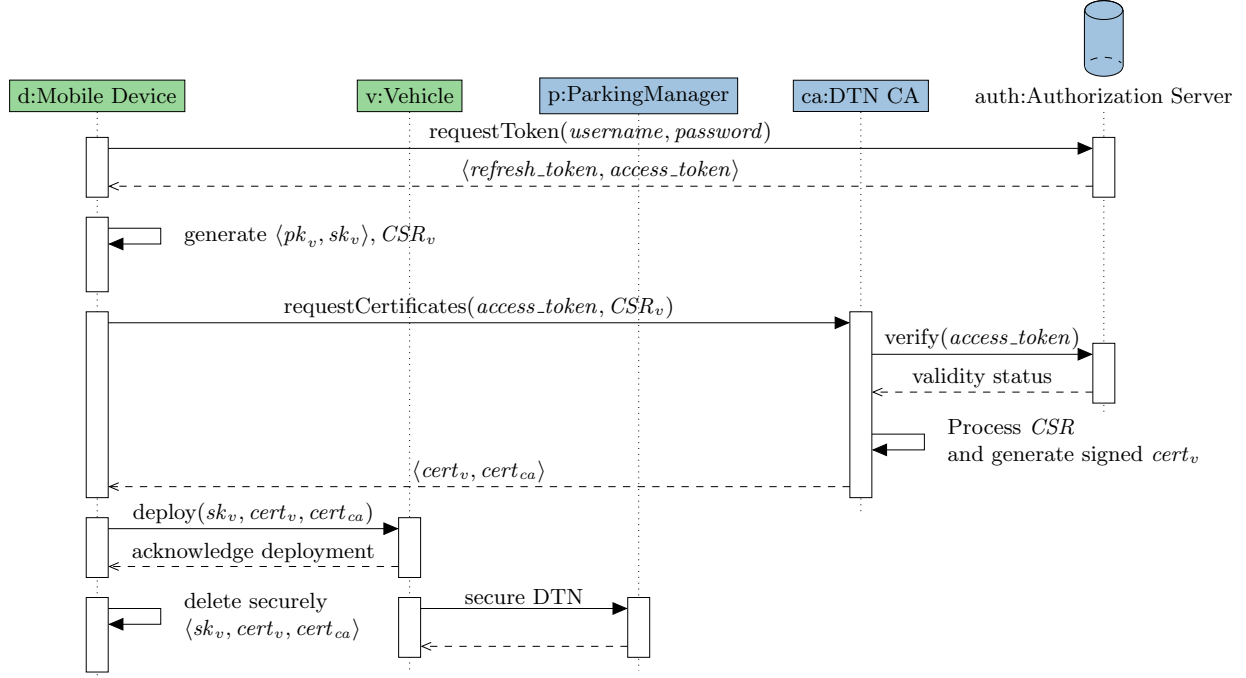


Figure 33: Key Deployment in V-Charge with aid of smartphones to allow key generation on mobile devices, which are owned and inherently trusted by customers themselves.

customers are required to drive to a V-Charge accredited service station. Employees manually check the VIN_v and store it as a new database entry belonging to the customer. Alternatively, other proof of possessing a specific vehicle with a corresponding VIN , such as digital possession certificates, could be accepted to overcome the manual verification step. The (optional) manual verification of VIN_v is the only step executed by employees of the service. The remaining key generation and deployment are solely done by the owner of the vehicle.

She starts the key generation process by using the Vehicle Registration Wizard, embedded in the smartphone application. After entering the vehicle for registration, identified by the VIN_v , a key pair $\langle pk_v, sk_v \rangle$ is generated offline on the mobile device d . We chose 4096 bit RSA key pairs for long-term security²³, supported by RFC 6257 [Sym+11]. A corresponding CSR_v , including pk_v and VIN_v , is built, signed by sk_v and sent to V-Charge’s DTN CA. While the connection is secured by TLS, the authorization and user affiliation are checked by validating the enclosed $access_token$ against the Authorization Server.

The DTN CA queries the vehicle database $\mathcal{V}$ of an authorized user, containing VINs and issued certificates:

$$\mathcal{V} = \{ \langle VIN_1, cert_1 \rangle, \dots, \langle VIN_n, cert_n \rangle \}.$$

CSR_v is processed if $\langle VIN_v, \epsilon \rangle \in \mathcal{V}$, i.e., the VIN exists in V-Charge’s database and no currently valid certificate is available. Our DTN CA generates $cert_v$ by signing the public key pk_v contained in CSR_v . Besides $cert_v$, the root certificate $cert_{ca}$ of our CA is returned.

Finally, $\langle sk_v, cert_v, cert_{ca} \rangle$, which is stored on the mobile device, must be deployed on the vehicle by utilizing NFC: an HSM is installed inside the vehicle along with the DTN stack that deploys new root certificates or $\langle sk_v, cert_v \rangle$ -pairs. The HSM is designed to only accept certificates that are issued for the VIN of the

²³Key length recommendations: <http://www.keylength.com>

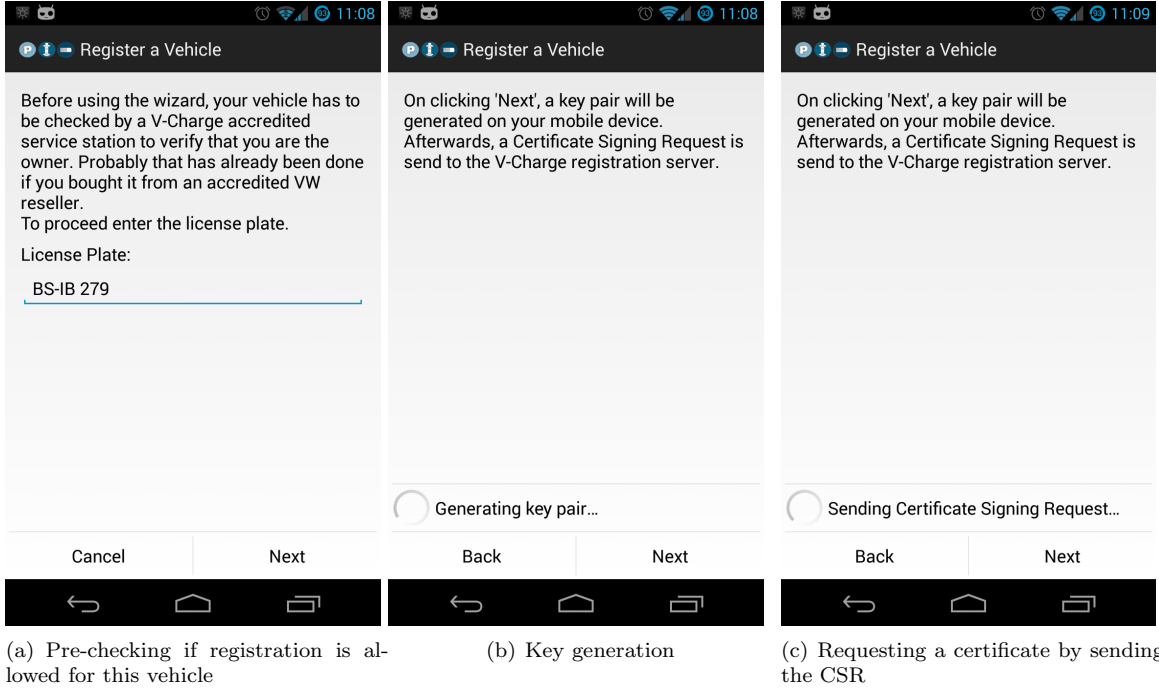


Figure 34: Vehicle Registration and Key Deployment Wizard on Android

Table 6: Input parameters of RESTful certificate API

Name	Type	Comment
URL: /rest/certificate/is-vehicle-existing		
<i>licensePlate</i>	String	For pre-checking vehicle in client's registration wizard
URL: /rest/certificate/certificate-signing-request		
<i>csr</i>	String	URL-safe Base64 encoding of <i>CSR</i>

corresponding vehicle. Only providing a deploy and reset functionality, it does not allow retrieval of private keys and is sufficiently secure against attacks attempting to extract keys. A PIN, which is stored offline by the vehicle owner, is needed to access the API of the HSM. After successful deployment, $\langle sk_v, cert_v, cert_{ca} \rangle$ is deleted from the mobile device in a secure way. For a short discussion of hardware requirements and assumptions, see Section 5.6.1.

The DTN stack in the vehicle is now able to communicate securely with PM servers and to verify if incoming bundles belong to the V-Charge DTN by checking if certificates corresponding to attached signatures are signed by $cert_{ca}$. Every outgoing bundle is signed using sk_v , so that other nodes will accept the bundle because of its signature, which is only valid for V-Charge participants.

Table 7: Return values of RESTful certificate API

Return Value	Type	Comment
URL: /rest/certificate/is-vehicle-existing		
<i>vehicleExisting</i>	boolean	
URL: /rest/certificate/certificate-signing-request		
<i>cert</i>	String	Base64-encoded certificate
<i>caCert</i>	String	Base64-encoded CA certificate

5.5.2 Implementation

Our implementation on the server and client side is based on the cryptographic library Bouncy Castle [Bou]. We utilized a API to enable processing of CSRs via REST requests. Related input parameters are described in Table 6 and their corresponding return values in Table 7. The input parameter *csr* expects an CSR according to RFC 5967 [Tur10]. Because all REST requests are URL-encoded parameter maps, the URL-safe variation of Base64 (`'base64url'`) is implemented, where `'-'` is used instead of `'+'` and `'.'` is used instead of `'/'` (cf. Section 5, RFC 4648 [Jos06]). As depicted in Figure 34, a convenient registration and key deployment wizard has been implemented for Android. At first, the wizards checks if registration is allowed for an entered license plate by querying the RESTful API. On the server side, the database is queried for accessing the username and returns true if the vehicle is allowed for a specific user. A 4096 bit RSA key pair is generated on the mobile device, followed by a CSR where its commonName (CN) is defined as the given *licensePlate*. This CSR is processed on the server by first checking again if the vehicle indeed is allowed to register (like in the first step). A certificate is generated by issuing a certifiante signed by the CA certificate of V-Charge’s DTN CA corresponding to the given CSR.

Currently, to ease testing in development scenarios, the license plate of a vehicle is used as an identifier. Later deployment in production environments must use the Vehicle Identification Number (VIN) as an identifier to guarantee collision freeness.

5.6 Discussion

By means of a smartphone-based registration and key deployment process for V2C communications, we are able to achieve a high degree of user independence from third parties, since nobody but the owner (not even the OEM) ever possesses the vehicle’s private key. Further, our open and easily auditable protocol warrants user trust in the underlying cryptographic principles.

Our system can be put into practice without dependencies on specialized mobile devices and with few requirements regarding HSMs for vehicles (cf. Section 5.6.1). Our process handles vehicles with already installed DTN stacks, as well as vehicles that require the installation of DTN hardware/software combination. This makes it usable for real world use cases. If DTN hardware is available, our registration and key deployment are solely based on in-app wizards to register accounts and vehicles to lower the acceptance threshold for new users. Car manufacturers can decide to sell vehicles with pre-configured DTN stacks (without pre-deployed keys or certificates), while it is also feasible to upgrade existing vehicles with HSMs and communications hardware.

Our implemented prototype uses operating system specific security practices where possible and provides a

convenient interface for vehicle registration. We utilized Android’s `AccountManager` for creating accounts and storing tokens.

Although we propose a solution aiming at the V-Charge project, our concepts are more generally applicable. For instance, any vehicular cloud service relying on a PKI could use the same process since we provide an application-independent means for secure key deployment without entrusting the service provider or a third party with the private key. Moreover, the proposed solution is applicable to vehicles that come pre-deployed with the required communication technologies as well as refitted ones. To extend the system for multi-service use, we suggest providing a user interface to switch between different service providers and their certificates to restrict the DTN stack to one provider at a time and prevent conflicting communications. In the illustrated example, we facilitate DTN as the underlying technology. Yet, the same principles can be applied to IP-based communications, for example, by replacing the Bundle Security Protocol with IPSec. Hence, we provide a solution to a trust issue in vehicular PKIs, thereby filling a research gap.

5.6.1 In-Vehicle Key Storage

To securely store private keys on the vehicle, an HSM is required. While specific hardware implementation details are beyond the scope of this thesis, some design specifications are assumed. In the automotive domain, HSMs should be especially long-lasting. Our DTN stack delegates the decrypt/sign operations to the HSM, while encrypt/verify can still be executed by the stack itself. This way, sk_v is never exposed. To protect sk_v even further, a viable option for future implementations might be to let the keys be directly generated by the HSM or to use its stored keys in a secure fashion [WG12]. In comparison to smartphone-based key generation though, this would require an additional step to extract the generated pk_v prior to requesting $cert_v$, as well as a more complex user interaction. The API should only consist of a mode to reset its memory and a deployment mode to store new $\langle sk_v, cert_v \rangle$ -pairs besides root certificates. On vehicle sale, the new owner is able to revert it to delivery condition using the reset mode. As vehicles are equipped with HSMs by service stations or car manufacturers, they need to be configured to only accept $\langle sk_v, cert_v \rangle$ -pairs issued for the vehicle’s VIN. We assume that a PIN is required to access the API, while NFC is proposed to serve as the communication protocol. In order to mitigate eavesdropping attacks, we assume a Diffie-Hellman (DH) key exchange, followed by an AES encrypted transmission. Hence, by using the NFC-SEC standard [ISO10] or an according implementation on the application layer, sk_v cannot be intercepted. As shown by previous research, MitM attacks on NFC are unfeasible [HB06].

5.6.2 Attack Scenarios

We discuss possible attack scenarios from the point of view of an attacker T , while t denotes the vehicle of attacker T .

Steal sk_v before deployment. An attacker deploys a Trojan on the user’s device by tricking him/her into installing software or by exploiting a vulnerability. To extract sk_v from the storage of the V-Charge application, the attacker needs to exploit the operating system for privilege escalation.

We argue that it is difficult to execute such attacks, as sk_v is stored only for a short duration on the mobile device, e.g., for a couple of hours, until the user deploys the key to the vehicle. Thus, an attacker needs to know in advance that a user will register for the V-Charge system. On Android, these type of attacks are directed towards the well-studied Unix filesystem permissions, as each application storage is separated using unique Unix users [And13; VVC11]. To strengthen smartphone security, a TrEE can be utilized, as it is required for solutions designed as key replacements [Bus+13].

Intercept $\langle username, password \rangle$. To intercept passwords, a keylogger needs to be installed before using the V-Charge app for the first time—as passwords are never stored—because our implementation is based on OAuth refresh tokens. This attack is almost as difficult to execute as the previous one because it needs to take advantage of vulnerabilities to gain system permissions and interfere with other applications by logging their keystrokes.

Another possibility is to intercept the connection between smartphone and server. By using certificate pinning for the TLS connection, the possibility to execute MitM attacks with forged certificates using an overtaken CA can be excluded. Thus, the TLS standard itself needs to be attacked, which we consider beyond the scope of typical attackers.

Intercept *access_token*. As described before, this would require attacking the TLS standard itself.

Extract *refresh_token*. A Trojan is required to extract tokens from `AccountManager`, implemented in the Android Operating System²⁴. Because this `AccountManager` is specifically designed to store passwords and access tokens, this attack also requires exploits to accomplish privilege escalation.

Another attack vector is the theft of the mobile device to gain physical access. This also includes unnoticed physical access while the user is distracted. Given that this allows an attacker to read the whole flash storage, the *refresh_token* can be extracted. As this is actually the most realistic and common attack scenario to access user data, we implemented a revocation mechanism by logging into the V-Charge service and revoking the device’s access.

Reset HSM credentials. As our HSM allows resetting and deployment of key pairs/certificates, an attacker could prevent the vehicle from operating with the V-Charge system by resetting HSM storage. This attack requires physical access to the HSM and previous theft of the PIN is needed to access it. Thus, we argue that this attack is difficult to execute, while easier forms of attacks on availability exists, e.g., stabbing the tires.

Eavesdropping on NFC. Eavesdropping while the private key is deployed requires an attacker who is in very close proximity to the devices and HSM. We assume a DH key exchange, followed by an AES encrypted transmission. Hence, by using the NFC-SEC standard [ISO10] or a corresponding implementation on the application layer, sk_v cannot be intercepted.

Relay attacks on NFC. Since the transmission of sk_v from the smartphone to the vehicle only happens once, as opposed to vehicular access control systems, the risk of a relay attack is low. Additionally, a manual confirmation step on the smartphone makes this attack even harder.

Register vehicle with made up VIN. As we require a manual verification step of the VIN by driving to a V-Charge service station or by providing alternative forms of owner verification, the attacker needs to fake the VIN or collude with V-Charge employees. Registering vehicles using a VIN previously not authenticated, is not possible as the CSR will be rejected by our CA.

Deploy attacker’s $\langle sk_t, cert_t \rangle$ to a victim’s vehicle. When the attacker has legitimately registered a vehicle, the retrieved pair $\langle sk_t, cert_t \rangle$ could be deployed to a victim’s vehicle using physical access to the HSM. This attack is prevented as our HSM only accepts $cert_t$ if it is issued for the corresponding VIN_v of the vehicle.

Extract $\langle sk_v, cert_v \rangle$ from a victim’s vehicle. This requires hacking into the HSM, which we assume to be infeasible by average attackers due to discussed assumptions on hardware. However, if sophisticated attacks are observed that successfully extract these secrets, a revocation of $cert_v$ has to be initiated and the owner of v is forced to re-generate sk_v and request a new $cert_v$.

Table 8: Overhead of cryptographic schemes in IBR-DTN; measurements in bytes

Key type	Cryptographic scheme			
	PIB	PCB	PCB-then-PIB	PIB-then-PCB
RSA-1024	142	174	316	365-367
RSA-2048	270	302	572	621-623
RSA-4096	526	558	1084	1132-1135

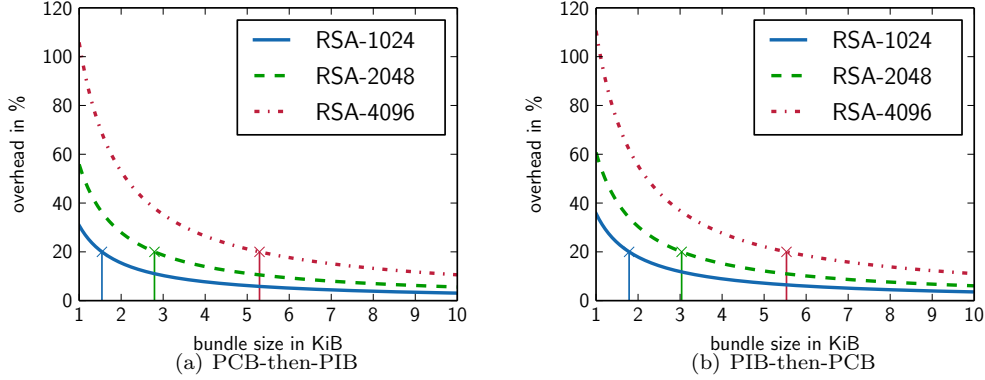


Figure 35: Overhead in percent depending on bundle size serialized by IBR-DTN; plotted line represents 20 % boundary

5.6.3 Evaluation

We evaluated the IBR-DTN daemon regarding its overhead resulting from cryptography as required by the thesis assignment. The evaluation is based on unit tests calculating the overhead in bytes for different cryptographic schemes. We differentiate between the schemes PIB-only (sign-only), PCB-only (encrypt-only), PCB-then-PIB (encrypt-then-sign), and PIB-then-PCB (sign-then-encrypt). IBR-DTN is configured to use ciphersuite `PIB_RSA_SHA256` for PIBs, and `PCB_RSA_AES128_PAYLOAD_PIB_PCB` for PIBs, as defined in RFC 6257 [Sym+11]. V-Charge’s configuration favors PCB-then-PIB to allow verification of bundles’ security source by nodes forwarding this bundle (cf. Section 5.4.1). While we evaluated the overhead for RSA with 1024, 2048, and 4096 bit, currently V-Charge’s Android application generates 4096 bit keys, for long-term security (cf. Section 5.5). Our numerical results are presented in Table 8. The results are of fixed size, because every test is configured to use the same ciphersuites, as previously introduced. Insignificant deviations in the PIB-then-PCB scheme result from non-deterministic output from the surrounding PCB due the underlying padding scheme. Figure 35 depicts the overhead corresponding to the bundle size for relevant schemes with both authenticity and confidentiality, (35(a)) PCB-then-PIB and (35(b)) PIB-then-PCB.

As defined by the associated EU project, cryptographic schemes should not introduce more than 20% overhead. This boundary is depicted in Figure 35 by a boundary line. Conclusively, to comply with this requirement when utilizing V-Charge’s favored scheme, PCB-then-PIB with 4096 bit RSA key pairs, the initial bundle size should be at least 1084 byte.

²⁴<http://developer.android.com/reference/android/accounts/AccountManager.html>

6 Final Integration and Evaluation

In this section, we provide the final integration and evaluation of the binding system. First, test specifications corresponding to T5.3 and T5.4 in M6 and as described in Deliverable 6.3 (D6.3) are defined. Second, the resulting experiments and their evaluation are described.

6.1 Experiment 1: Parking Management

Task 5.3 of M6 in D6.3 specifies the evaluation of “Management and Communication Components” as follows:

Capability tested: Demonstrate parking spot management, charging station management, and pick-up/drop-off station management.

Outline of test procedure:

1. Simulate a certain number of vehicles.
2. Vehicles are checked in, dropped off, and picked up.
3. Server assigns parking spots and charging stations to the vehicles and generates vehicle commands accordingly.

Description of evaluation metric: Server efficiently manages:

- 50 parking slots
- 2 charging stations
- 2 pick-up/drop-off stations
- 50 vehicles

Results: The simulation of the parking management components has been described in detail in Section 4.4. Striving for evaluation results being as realistic as possible, we went out of our way to obtain real parking statistics to base the simulations on. As a result, our simulation parameters outperform the originally specified parameters in Section 6.1 greatly. For instance, instead of 50 parking spots, we have simulated two scenarios with 200 and 1250 parking spots, respectively. Instead of 2 charging stations, our scenarios include up to 50.

We showed that the V-Charge Server is able to efficiently handle realistic parking volume and compared the schedulers’ suitability in two very likely scenarios. With the metrics we have developed, we were further able to identify the best suited scheduler for each scenario, namely the Scanning strategy which yields high utilizations while minimizing CS switching overhead.

6.2 Experiment 2: Security Overhead

According to Task 5.4 of M6 in D6.3, security during data transmission (confidentiality/integrity) shall reduce throughput by not more than 20%.

More specifically:

Capability tested: Security concept setup and functional.

Outline of test procedure:

1. Turn confidentiality/integrity on.
2. Transmit data.
3. Compare throughput with non-secured channel.

Description of evaluation metric: Security during data transmission (confidentiality/integrity) reduces throughput by not more than 20%.

Results: The final security concept and implementation has been described in detail in Section 5, which includes an extensive discussion in Section 5.6. In addition to the performance metric specified in Section 6.2, possible attack scenarios and their mitigations are discussed. Finally, Section 5.6.3 provides results on the resulting security overhead.

References

- [AAJ03] Björn Andersson, Tarek Abdelzaher, and Jan Jonsson. “Global Priority-Driven Aperiodic Scheduling on Multiprocessors.” In: *Proceedings of the International Parallel and Distributed Processing Symposium*. Nice, France: IEEE Comput. Soc, Apr. 2003. ISBN: 0-7695-1926-1. DOI: 10.1109/IPDPS.2003.1213082. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=1213082%20http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1213082.
- [Abd+12] Kamilah Abdullah, Nor Hanimah Kamis, Nurul Farahiyah Nor Azahar, Siti Faiross Shariff, and Zairina Che Musa. “Optimization of the parking spaces: A case study of Dataran Mawar, UiTM Shah Alam.” In: *CHUSER*. Kota Kinabalu, Malaysia: IEEE, Dec. 2012, pp. 684–687. ISBN: 978-1-4673-4617-7. DOI: 10.1109/CHUSER.2012.6504399. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6504399>.
- [And13] Android Open Source Project. *Android Security Overview*. [Online; accessed 30-June-2013]. 2013. URL: <https://source.android.com/tech/security>.
- [AP13] Nadhem J AlFardan and Kenneth G Paterson. “Lucky thirteen: Breaking the TLS and DTLS record protocols.” In: *IEEE Symposium on Security and Privacy*. 2013.
- [Aso+07] N. Asokan, Kari Kostiaainen, Philip Ginzboorg, Jörg Ott, and Cheng Luo. “Applicability of identity-based cryptography for disruption-tolerant networking.” In: *Proc. of the 1st International MobiSys Workshop on Mobile Opportunistic Networking (MobiOpp ’07)*. San Juan, Puerto Rico: ACM, 2007, pp. 52–56. ISBN: 978-1-59593-688-2. DOI: 10.1145/1247694.1247705.
- [Bar+93] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. “Proportionate Progress: A Notion of Fairness in Resource Allocation.” In: *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC ’93)*. Vol. 15. 6. San Diego, CA: ACM Press, June 1993, pp. 345–354. ISBN: 0-89791-591-7. DOI: 10.1145/167088.167194.
- [BBM12] C. Bansal, K. Bhargavan, and S. Maffei. “Discovering Concrete Attacks on Website Authorization by Formal Analysis.” In: *25th Computer Security Foundations Symposium (CSF 2012)*. 2012, pp. 247–262. DOI: 10.1109/CSF.2012.27.
- [BFV11] S. Busanelli, G. Ferrari, and L. Veltri. “Short-lived key management for secure communications in VANETs.” In: *11th International Conference on ITS Telecommunications (ITST)*. 2011, pp. 613–618. DOI: 10.1109/ITST.2011.6060129.
- [Bou] Bouncy Castle Contributors. *Bouncy Castle Specifications*. URL: <http://www.bouncycastle.org/specifications.html>.
- [Bra07] Stefan Brands. *The problem(s) with OpenID*. [Online; accessed 2013-10-24]. Aug. 2007. URL: <http://www.untrusted.ca/cache/openid.html>.
- [Bus+13] Christoph Busold, Ahmed Taha, Christian Wachsmann, Alexandra Dmitrienko, Hervé Seudie, Majid Sobhani, and Ahmad-Reza Sadeghi. “Smart Keys for Cyber-Cars: Secure Smartphone-based NFC-enabled Car Immobilizer.” In: *Proc. of the Third ACM Conference on Data and Application Security and Privacy (CODASPY ’13)*. ACM, 2013, pp. 233–242. ISBN: 978-1-4503-1890-7.
- [CAc13] CAcert Wiki. *CA Risk History*. [Online; accessed 30-June-2013]. 2013. URL: <http://wiki.cacert.org/Risk/History>.
- [Cer+07] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. *Delay-Tolerant Networking Architecture*. RFC 4838 (Informational). Internet Engineering Task Force, Apr. 2007. URL: <http://www.ietf.org/rfc/rfc4838.txt>.

- [Com+07] LAN/MAN Committee et al. “IEEE Std 802.11-2007: IEEE Standard for Information Technology-Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks-Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.” In: *IEEE Computer Society* (2007).
- [EH10] Melike Erol-Kantarci and T. Mouftah Hussein. “Prediction-Based Charging of PHEVs from the Smart Grid with Dynamic Pricing.” In: *Proceedings of the 35th IEEE Conference on Local Computer Networks (LCN)*. Denver, CO: IEEE, Oct. 2010, pp. 1032–1039. ISBN: 978-1-4244-8387-7. DOI: 10.1109/LCN.2010.5735676. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=5735676.
- [Eis12] Markus Eisele. *GlassFish JDBC Security with Salted Passwords on MySQL*. July 2012. URL: <http://blog.eisele.net/2012/07/glassfish-jdbc-security-with-salted.html>.
- [ESM12] Melike Erol-Kantarci, Jahangir H. Sarker, and Hussein T. Mouftah. “Quality of Service in Plug-in Electric Vehicle Charging Infrastructure.” In: *Proceedings of the IEEE International Electric Vehicle Conference (IEVC ’12)*. Greenville, SC: IEEE, Mar. 2012, pp. 441–445. ISBN: 978-1-4673-1561-6. DOI: 10.1109/IEVC.2012.6183227. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=6183227.
- [Fel+11] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. “Android permissions demystified.” In: *Proc. of the 18th ACM Conference on Computer and Communications Security (CCS ’11)*. Chicago, Illinois, USA: ACM, 2011, pp. 627–638. ISBN: 978-1-4503-0948-6. DOI: 10.1145/2046707.2046779.
- [Fer+14] Michel Ferreira, Luis Damas, Hugo Conceicao, Pedro M. D’Orey, Ricardo Fernandes, Peter Steenkiste, and Pedro Gomes. “Self-Automated Parking Lots for Autonomous Vehicles Based on Vehicular Ad Hoc Networking.” In: *Intelligent Vehicles Symposium (IV)*. Dearborn, MI: IEEE, June 2014, pp. 472–479. ISBN: 978-1-4799-3638-0. DOI: 10.1109/IVS.2014.6856561. URL: <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6856561>.
- [Fic] FictiveKin and friends. *LeakedIn*. [Online; accessed 2013-06-30]. URL: <http://leakedin.org>.
- [FKK11] A. Freier, P. Karlton, and P. Kocher. *The Secure Sockets Layer (SSL) Protocol Version 3.0*. RFC 6101 (Historic). Internet Engineering Task Force, Aug. 2011. URL: <http://www.ietf.org/rfc/rfc6101.txt>.
- [Fra+99] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. *HTTP Authentication: Basic and Digest Access Authentication*. RFC 2617 (Draft Standard). Internet Engineering Task Force, June 1999. URL: <http://www.ietf.org/rfc/rfc2617.txt>.
- [Fre+10] Rui Freire, Joaquim Delgado, Joao M. Santos, and Anibal T. de Almeida. “Integration of Renewable Energy Generation with EV Charging Strategies to Optimize Grid Load Balancing.” In: *Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. Madeira Island, Portugal: IEEE, Sept. 2010, pp. 392–396. ISBN: 978-1-4244-7657-2. DOI: 10.1109/ITSC.2010.5625071. URL: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5625071>.
- [Fru11] C. Fruhwirth. *LUKS On-Disk Format Specification Version 1.2*. 2011.
- [Fur+13] Paul Furgale, Ulrich Schwesinger, Martin Rufli, Wojciech Derendarz, Hugo Grimmer, Peter Mühlfeßner, Stefan Wonneberger, Julian Timpner, Stephan Rottmann, Bo Li, Bastian Schmidt, Thien Nghia Nguyen, Elena Cardarelli, Stefano Cattani, Stefan Brünig, Sven Horstmann, Martin Stellmacher, Holger Mielenz, Kevin Köser, Markus Beermann, Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, René Iser, Rudolph Triebel, Ingmar Posner, Paul Newman, Lars Wolf, Marc Pollefeys, Stefan Brosig, Jan Effertz, Cédric Pradalier, and Roland Siegwart. “Toward Automated Driving in Cities using Close-to-Market Sensors: An Overview of the V-Charge Project.” In: *Intelligent Vehicles Symposium (IV)*. Gold Coast, Australia: IEEE, June 2013, pp. 809–816. ISBN: 978-1-4673-2754-1. DOI: 10.1109/IVS.2013.6629566.

- [GHP13] Yoel Gluck, Neal Harris, and Angelo Ángel Prado. *BREACH: REVIVING THE CRIME ATTACK*. 2013. URL: <http://breachattack.com/resources/BREACH%20-%20SSL,%20gone%20in%2030%20seconds.pdf>.
- [Goo] Google Services. *Authorization*. [Online; accessed 2013-10-22]. URL: <http://developer.android.com/google/play-services/auth.html>.
- [Grü+11a] Max Grünig, Mark Witte, Dominic Marcellino, Jordan Selig, and Huib van Essen. *Impact of Electric Vehicles—Deliverable 1: An Overview of Electric Vehicles on the Market and in Development*. Tech. rep. 11.4058.03. Delft, Netherlands: CE Delft and Ecologic, Apr. 2011. URL: http://www.cedelft.eu/publicatie/impact%5C_of%5C_electric%5C_vehicles/1153.
- [Grü+11b] Max Grünig, Mark Witte, Dominic Marcellino, Jordan Selig, and Huib van Essen. *Impact of Electric Vehicles—Deliverable 3: Assessment of the Future Electricity Sector*. Tech. rep. 11.4058.05. Delft, Netherlands: CE Delft and Ecologic, Apr. 2011. URL: http://www.cedelft.eu/publicatie/impact%5C_of%5C_electric%5C_vehicles/1153.
- [GTL11] Lingwen Gan, Ufuk Topcu, and Steven Low. “Optimal Decentralized Protocol for Electric Vehicle Charging.” In: *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*. Orlando, FL: IEEE, Dec. 2011, pp. 5798–5804. ISBN: 978-1-61284-801-3. DOI: 10.1109/CDC.2011.6161220. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=6161220.
- [Ham11] Eran Hammer-Lahav. *The OAuth 1.0 Guide - History*. [Online; accessed 2013-06-30]. July 2011. URL: <http://hueniverse.com/oauth/guide/history>.
- [Har12] D. Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749 (Proposed Standard). Internet Engineering Task Force, Oct. 2012. URL: <http://www.ietf.org/rfc/rfc6749.txt>.
- [HB06] Ernst Haselsteiner and Klemens Breitfuß. “Security in Near Field Communication (NFC).” In: *Workshop on RFID Security (RFIDSec)*. 2006.
- [HD97] John M. Harris and Yasser Dessouky. “A Simulation Approach for Analyzing Parking Space Availability at a Major University.” In: *Conf. Winter Simulation (WSC)*. Atlanta, GA: ACM Press, Dec. 1997, pp. 1195–1198. ISBN: 078034278X. DOI: 10.1145/268437.268759. URL: <http://dl.acm.org/citation.cfm?id=268437.268759>.
- [Hom13] Egor Homakov. *How we hacked Facebook with OAuth2 and Chrome bugs*. [Online; accessed 2013-10-24]. Feb. 2013. URL: <http://homakov.blogspot.co.uk/2013/02/hacking-facebook-with-oauth2-and-chrome.html>.
- [ISO10] ISO. *Information technology – Telecommunications and information exchange between systems – NFC Security – Part 2: NFC-SEC cryptography standard using ECDH and AES*. ISO/IEC 13157-2. Also available as ECMA-386. 2010.
- [Jee+10] P Jeevan, F Harchut, B Mueller-Bessler, and B Huhnke. “Realizing Autonomous Valet Parking with automotive grade sensors.” In: *IROS*. Taipei, Taiwan: IEEE, Oct. 2010, pp. 3824–3829. ISBN: 978-1-4244-6674-0. DOI: 10.1109/IROS.2010.5649387. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5649387>.
- [JH12] M. Jones and D. Hardt. *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. RFC 6750 (Proposed Standard). Internet Engineering Task Force, Oct. 2012. URL: <http://www.ietf.org/rfc/rfc6750.txt>.
- [Jos06] S. Josefsson. *The Base16, Base32, and Base64 Data Encodings*. RFC 4648 (Proposed Standard). Internet Engineering Task Force, Oct. 2006. URL: <http://www.ietf.org/rfc/rfc4648.txt>.
- [JTG13] Chenrui Jin, Jian Tang, and P Ghosh. “Optimizing Electric Vehicle Charging With Energy Storage in the Electricity Market.” In: *IEEE Transactions on Smart Grid* 4.1 (2013), pp. 311–320. ISSN: 1949-3053. DOI: 10.1109/TSG.2012.2218834.

- [Kal00] B. Kaliski. *PKCS #5: Password-Based Cryptography Specification Version 2.0*. RFC 2898 (Informational). Internet Engineering Task Force, Sept. 2000. URL: <http://www.ietf.org/rfc/rfc2898.txt>.
- [Kim+10] Hye-Jin Kim, Junghoon Lee, Gyung-Leen Park, Min-Jae Kang, and Mikyung Kang. “An Efficient Scheduling Scheme on Charging Stations for Smart Transportation.” In: *Security-Enriched Urban Computing and Smart Grid*. Ed. by Tai-hoon Kim, Adrian Stoica, and Ruay-Shiung Chang. Vol. 78. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2010, pp. 274–278. ISBN: 978-3-642-16443-9. DOI: 10.1007/978-3-642-16444-6_35. URL: http://dx.doi.org/10.1007/978-3-642-16444-6_35.
- [KZH07] A. Kate, G.M. Zaverucha, and U. Hengartner. “Anonymity and security in delay tolerant networks.” In: *Third International Conference on Security and Privacy in Communications Networks and the Workshops (SecureComm 2007)*. 2007, pp. 504–513. DOI: 10.1109/SECCOM.2007.4550373.
- [Lin] LinkedIn Corporation. *An Update on LinkedIn Member Passwords Compromised*. [Online; accessed 2013-06-30]. URL: <http://blog.linkedin.com/2012/06/06/linkedin-member-passwords-compromised>.
- [LLK13] B. Laurie, A. Langley, and E. Kasper. *Certificate Transparency*. RFC 6962 (Experimental). Internet Engineering Task Force, June 2013. URL: <http://www.ietf.org/rfc/rfc6962.txt>.
- [LMH13] T. Lodderstedt, M. McGloin, and P. Hunt. *OAuth 2.0 Threat Model and Security Considerations*. RFC 6819 (Informational). Internet Engineering Task Force, Jan. 2013. URL: <http://www.ietf.org/rfc/rfc6819.txt>.
- [Lop+13] Christian Loper, Claas Brunken, George Thomaidis, Stephan Lapoehn, Paulin Pekezou Fouopi, Henning Mosebach, and Frank Koster. “Automated Valet Parking as Part of an Integrated Travel Assistance.” In: *Conf. ITS (ITSC)*. The Hague, Netherl.: IEEE, Oct. 2013, pp. 2341–2348. ISBN: 978-1-4799-2914-6. DOI: 10.1109/ITSC.2013.6728577. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6728577>.
- [LS11] Dieter Lohse and Werner Schnabel. *Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung: Band 2 - Verkehrsplanung*. Beuth Studium. Beuth Verlag GmbH, 2011. ISBN: 9783410206880. URL: <http://books.google.de/books?id=I2f-bSsW04MC>.
- [Mar13] M. Marlinspike. *Trust Assertions for Certificate Keys*. Ed. by T. Perrin. Internet-Draft. Internet-Draft. Internet Engineering Task Force, Jan. 2013.
- [MC13] Kyoung-Wook Min and Jeong-Dan Choi. “Design and Implementation of Autonomous Vehicle Valet Parking System.” In: *Conf. ITS (ITSC)*. The Hague, Netherl.: IEEE, Oct. 2013, pp. 2082–2087. ISBN: 978-1-4799-2914-6. DOI: 10.1109/ITSC.2013.6728536. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6728536>.
- [MIA10] Said Munzir, Mahyus Ikhsan, and Zainal Amin. “Linear Programming for Parking Slot Optimization: A Case Study at Jl. T. Panglima Polem Banda Aceh.” In: *Conf. Math., Stat. and its Appl. (ICMSA)*. Kuala Lumpur, Malaysia, 2010, pp. 462–472.
- [Mic] Microsoft Corporation. *Microsoft Security Bulletin MS12-006*. URL: <http://technet.microsoft.com/en-us/security/bulletin/ms12-006>.
- [Mor10] Timothy D. Morgan. *HTTP Digest Integrity: Another look, in light of recent attacks*. [Online; accessed 2013-10-24]. Jan. 2010. URL: <https://secure.vsecurity.com/download/papers/HTTPDigestIntegrity.pdf>.
- [NAP13] Anuj K. Nayak, H.C. Akash, and Geetha Prakash. “Robotic Valet Parking System.” In: *Proc. Texas Instruments India Educators’ Conf.* Bangalore, India: IEEE, Apr. 2013, pp. 311–315. ISBN: 978-0-7695-5146-3. DOI: 10.1109/TIIEC.2013.62. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6757160>.

- [Oraa] Oracle Corporation. *Java™ Cryptography Architecture Standard Algorithm Name Documentation*. URL: <http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#MessageDigest>.
- [Orab] Oracle Corporation. *The Java EE 6 Tutorial*. URL: <http://docs.oracle.com/javaee/6/tutorial/doc/javaeetutorial6.pdf>.
- [Ott+11] Jörg Ott, Esa Hyytiä, Pasi Lassila, Tobias Vaegs, and Jussi Kangasharju. “Floating Content: Information Sharing in Urban Areas.” In: *IEEE International Conference on Pervasive Computing and Communications (PerCom)*. Seattle, USA, Mar. 2011.
- [OWAa] OWASP Foundation. *Session Management Cheat Sheet*. URL: https://www.owasp.org/index.php/Session_Management_Cheat_Sheet.
- [OWAb] OWASP Foundation. *Top 10 - 2013*. URL: <http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf>.
- [Pap+08] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Zhendong Ma, F. Kargl, A. Kung, and J-P Hubaux. “Secure Vehicular Communication Systems: Design and Architecture.” In: *Communications Magazine, IEEE* 46.11 (2008), pp. 100–109. ISSN: 0163-6804. DOI: 10.1109/MCOM.2008.4689252.
- [Pau] Ryan Paul. *Compromising Twitter’s OAuth security system*. [Online; accessed 2013-10-24]. URL: <http://arstechnica.com/security/2010/09/twitter-a-case-study-on-how-to-do-oauth-wrong>.
- [Per09] Colin Percival. “Stronger key derivation via sequential memory-hard functions.” In: *The Technical BSD Conference (BSDCan 2009)*. 2009.
- [PJ13] C. Percival and S. Josefsson. *The scrypt Password-Based Key Derivation Function*. Internet-Draft. Internet Engineering Task Force, Sept. 2013. URL: <http://www.ietf.org/rfc/rfc6847.txt>.
- [PM99] Niels Provos and David Mazieres. “A future-adaptable password scheme.” In: *Proceedings of the Annual USENIX Technical Conference*. 1999.
- [QZ11] Hua Qin and Wensheng Zhang. “Charging Scheduling with Minimal Waiting in A Network of Electric Vehicles and Charging Stations.” In: *Proceedings of the Eighth ACM International Workshop on Vehicular Inter-NETworking (VANET ’11)*. Las Vegas, NV: ACM Press, Sept. 2011, pp. 51–60. ISBN: 9781450308694. DOI: 10.1145/2030698.2030706. URL: <http://dl.acm.org/citation.cfm?id=2030698.2030706>.
- [RD11] Julianio Rizzo and Thai Duong. *BEAST*. [Online; accessed 15-October-2013]. 2011. URL: <http://vnhacker.blogspot.de/2011/09/beast.html>.
- [RD12] Julianio Rizzo and Thai Duong. *The CRIME attack*. [Online; accessed 2013-10-15]. 2012. URL: http://netifera.com/research/crime/CRIME_ekoparty2012.pdf.
- [RR06] David Recordon and Drummond Reed. “OpenID 2.0: a platform for user-centric identity management.” In: *Proceedings of the Second ACM Workshop on Digital Identity Management (DIM ’06)*. Alexandria, Virginia, USA: ACM, 2006, pp. 11–16. ISBN: 1-59593-547-9. DOI: 10.1145/1179529.1179532.
- [RWB93] Sabah U. Randhawa, Stephen J. White, and Sheikh Burhanuddin. “A Simulation-based Analysis of Parking System Performance.” In: *Conf. Winter Simulation (WSC)*. Los Angeles: ACM Press, Dec. 1993, pp. 1231–1238. ISBN: 078031381X. DOI: 10.1145/256563.257011. URL: <http://dl.acm.org/citation.cfm?id=256563.257011>.
- [SB07] K. Scott and S. Burleigh. *Bundle Protocol Specification*. RFC 5050 (Experimental). Internet Engineering Task Force, Nov. 2007. URL: <http://www.ietf.org/rfc/rfc5050.txt>.

- [Sch+11] Sebastian Schildt, Johannes Morgenroth, Wolf-Bastian Pöttner, and Lars Wolf. “IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation.” In: *Electronic Communications of the EASST* 37 (Jan. 2011).
- [SKK12] Mukesh Singh, Praveen Kumar, and Indrani Kar. “Coordination of Multi Charging Station for Electric Vehicles and its Utilization for Vehicle to Grid Scenario.” In: *Proceedings of the IEEE Transportation Electrification Conference and Expo (ITEC)*. Dearborn, MI: IEEE, June 2012, pp. 89–95. ISBN: 978-1-4673-1408-4. DOI: 10.1109/ITEC.2012.6243430. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=6243430.
- [Stu+09] A. Studer, E. Shi, Fan Bai, and A. Perrig. “TACKing Together Efficient Authentication, Revocation, and Privacy in VANETs.” In: *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON ’09)*. 2009. DOI: 10.1109/SAHCN.2009.5168976.
- [Sym+11] S. Symington, S. Farrell, H. Weiss, and P. Lovell. *Bundle Security Protocol Specification*. RFC 6257 (Experimental). Internet Engineering Task Force, May 2011. URL: <http://www.ietf.org/rfc/rfc6257.txt>.
- [Tim+15] Julian Timpner, Stephan Friedrichs, Johannes van Balen, and Lars Wolf. “k-Stacks: High-Density Valet Parking for Automated Vehicles.” In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Seoul, Korea: IEEE, June 2015.
- [Tim12] Julian Timpner. “Design, Implementation, and Evaluation of Central Components for the V-Charge System.” MA thesis. Germany: IBR, TU Braunschweig, 2012.
- [Tra10] Transportation Research Board. *ACRP Report 40: Airport Curbside and Terminal Area Roadway Operations*. ACRP report. The National Academies Press, 2010. ISBN: 9780309281010. URL: http://www.nap.edu/openbook.php?record%5C_id=14451%20http://books.google.de/books?id=rrW0hq-8GjIC.
- [TSW13] Julian Timpner, Dominik Schürmann, and Lars Wolf. “Secure Smartphone-based Registration and Key Deployment for Vehicle-to-cloud Communications.” In: *Proc. of the First International Academic Workshop on Security, Privacy & Dependability for Cyber Vehicles (CyCAR ’13)*. Berlin, Germany: ACM, 2013, pp. 31–36. ISBN: 978-1-4503-2487-8. DOI: 10.1145/2517968.2517970.
- [Tur+10] Meltem Sönmez Turan, Elaine Barker, William Burr, and Lily Chen. “Recommendation for Password-Based Key Derivation.” In: *NIST special publication* 800 (2010), p. 132.
- [Tur10] S. Turner. *The application/pkcs10 Media Type*. RFC 5967 (Informational). Internet Engineering Task Force, Aug. 2010. URL: <http://www.ietf.org/rfc/rfc5967.txt>.
- [TW12a] Stefan Tillich and Marcin Wójcik. “Security Analysis of an Open Car Immobilizer Protocol Stack.” In: *Trusted Systems*. Ed. by ChrisJ. Mitchell and Allan Tomlinson. Vol. 7711. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 83–94. ISBN: 978-3-642-35370-3.
- [TW12b] Julian Timpner and Lars Wolf. “A Back-end System for an Autonomous Parking and Charging System for Electric Vehicles.” In: *IEEE International Electric Vehicle Conference*. Greenville, SC, Mar. 2012, pp. 693–700. ISBN: 978-1-4673-1561-6.
- [TW12c] Julian Timpner and Lars Wolf. “Efficient Charging Station Scheduling for an Autonomous Parking and Charging System.” In: *Proceedings of the Ninth ACM International Workshop on Vehicular Inter-NETworking, Systems, and Applications (VANET ’12)*. Low Wood Bay, Lake District, UK: ACM Press, June 2012, pp. 145–148. ISBN: 9781450313179. DOI: 10.1145/2307888.2307918. URL: <http://dl.acm.org/citation.cfm?id=2307888.2307918>.
- [TW14] Julian Timpner and Lars Wolf. “Design and Evaluation of Charging Station Scheduling Strategies for Electric Vehicles.” In: *IEEE Transactions on Intelligent Transportation Systems* 15.2 (2014), pp. 579–588. DOI: 10.1109/TITS.2013.2283805.

- [Van10] William L. Van Besien. “Dynamic, non-interactive key management for the bundle protocol.” In: *Proc. of the 5th ACM Workshop on Challenged Networks (CHANTS '10)*. Chicago, Illinois, USA: ACM, 2010, pp. 75–78. ISBN: 978-1-4503-0139-8. DOI: 10.1145/1859934.1859951.
- [VVC11] Timothy Vidas, Daniel Votipka, and Nicolas Christin. “All Your Droid Are Belong to Us: A Survey of Current Android Attacks.” In: *Proc. of the 5th USENIX Conference on Offensive Technologies*. USENIX Association, 2011.
- [WAP08] Dan Wendlandt, David Andersen, and Adrian Perrig. “Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing.” In: *Proc. of the USENIX Annual Technical Conference*. Boston, MA, June 2008.
- [Was+10] A. Wasef, Rongxing Lu, Xiaodong Lin, and Xuemin Shen. “Complementing Public Key Infrastructure to Secure Vehicular Ad Hoc Networks [Security and Privacy in Emerging Wireless Networks].” In: *Wireless Communications, IEEE* 17.5 (2010), pp. 22–28. ISSN: 1536-1284. DOI: 10.1109/MWC.2010.5601954.
- [WG12] Marko Wolf and Timo Gendrullis. “Design, Implementation, and Evaluation of a Vehicular Hardware Security Module.” In: *Information Security and Cryptology (ICISC 2011)*. Ed. by Howon Kim. Vol. 7259. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 302–318. ISBN: 978-3-642-31911-2.
- [Xie+11] Feixiang Xie, Mei Huang, WeiGe Zhang, and Juan Li. “Research on Electric Vehicle Charging Station Load Forecasting.” In: *Proceedings of the International Conference on Advanced Power System Automation and Protection*. Beijing, China: IEEE, Oct. 2011, pp. 2055–2060. ISBN: 978-1-4244-9621-1. DOI: 10.1109/APAP.2011.6180772. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=6180772.
- [You88] William Young. “A review of parking lot design models.” In: *Transport Reviews* 8.2 (Apr. 1988), pp. 161–181. ISSN: 0144-1647. DOI: 10.1080/01441648808716682. URL: <http://www.tandfonline.com/doi/abs/10.1080/01441648808716682>.
- [YY97] Wen Long Yue and William Young. “Simulation in Selecting the Best Parking Lot Layout.” In: *Congress Modelling and Simulation (MODSIM)*. Hobart, Australia, Dec. 1997, pp. 1484–1489.