

Design of a Shared Parking System with special attention to security aspects

Simon Englert (2136190)

Würzburg, am August 29, 2018

Contents

1 Abstract

As major cities are growing more and more, the parking situation in those cities is becoming more precarious than ever. Parking demand already exceeds the limited amount of parking spaces available, and thus people searching for a place to park generate a significant part of those cities traffic. This leads to frustration with drivers, more traffic jams, unnecessary use of petrol and further air pollution. The creation of new parking spaces is often either difficult or very expensive, which means existing parking spots have to be used more efficiently. Solutions realizing that idea to try and solve those problems are for the most part called intelligent parking systems. This thesis is focused on Shared Parking, a special kind of intelligent parking system, which opens the possibility of sharing a parking place between different people. This work features the design and implementation of such a system with special attention given to security aspects.

2 Indrodution

Parking in large cities is a well-known problem. For decades, the search for a parking space has been something that annoys most people. Due to the ever faster growing cities in recent years, the resulting difficulties are only intensifying. The ensuing traffic for example already represents a significant part of the total traffic in city centres. According to studies around 30 percent of the daily city traffic comes from car park seekers.[?] In addition to the rising number of people in cities, the percentage of people who own a car continues to increase, which adds to the traffic problem. However, the traffic is not just frustrating for the drivers. The ensuing traffic jams and slow-moving traffic cause an increased amount of petrol to be burned and the air to become additionally polluted. Especially in these times, when air pollution and climate warming are becoming even more important due to Dieselgate and the capricious weather conditions, one should be looking for a solution.

The creation of new parking spaces is rarely an option. On one hand, this is often associated with high costs, on the other hand, there is usually no space for the construction of new parking spaces available in the cities. Instead, existing parking spaces must be used more effectively.

This can be done in many different ways. So-called intelligent parking systems increase the smartness of parking in different areas and therefore often lead to a more effective use of parking spaces. Intelligent parking systems are available in various forms and for most of them there are numerous scientific papers and completed projects. However, there are hardly any relevant developments for shared parking systems, which are by definition a type of intelligent parking system. A shared parking system is a trading platform on which private users

can rent out and rent parking spaces. Sharing systems, such as car sharing, have become increasingly influential in recent years and are expected to continue to grow strongly. If an object is shared among different users, it is used more effectively. This applies to parking places and a shared parking system as well. Therefore, the main part of this work will be concerned with designing a shared parking system. Special attention is paid to security aspects by detailing how the system handles fraud.

First, an overview of the system is presented, in particular the system model, which depicts the different roles in a shared parking system. This is followed by functional and security requirements to be met by the design. In contrast to existing systems, the design of our system will try avoid complex hardware and manual intervention. Possible cases of fraud are described and it is noted that our system must recognize and prevent them, punish the adversaries and compensate the injured party. The actual design is described in the next step. First all functional features and then the security features that determine how to deal with fraud are explained. Various modules are introduced to deal with fraud, for example enabling the rating of users or the reporting and verification of fraud.

Subsequently, the design is evaluated on the basis of the requirements. We shall find, that the security requirements are met in most cases and that the error probability approaches 0 with increasing system size.

Once it has been shown that the design meets the requirements, an implementation, which is also part of this study, will be discussed in more detail. The design was implemented as a client-server model with Java web service running on a TomCat server and Android app for the end user.

Finally, future work will be proposed and a summary of the results will be drawn up.

3 Related Work and Background

If one takes a look at existing work in the field of intelligent parking systems, he will encounter countless scientific papers in the most diverse sub-areas of intelligent parking systems. In addition, there are numerous projects in which such systems have already been implemented. There are also some works that have set themselves the task of categorizing related work in this field. Nevertheless, there does not seem to be any definite consensus among scientists on definitions and categorizations. In the following we will present an introduction to the topic based on these papers.

3.1 Overview

Intelligent Parking Systems In general terms, Intelligent Parking Systems, also called smart parking systems, are systems that try to solve the countless parking problems already mentioned while integrating 'advanced technologies and researches from various academic disciplines' [?]. Thus the words intelligent or smart have different meanings depending on the time at which a system was developed [?]. With further progress, especially in information technology, new opportunities are opening up and old implementations no longer seem to be 'advanced' or 'smart', but were at the time of development.

Shared Parking Shared Parking Systems form a separate category of intelligent parking systems, because they use the emerging 'shared technology' in connection with mobile devices, which are available to most people due to technological advances. Shared parking provides the framework for making purely private parking spaces available to the public by allowing them to be traded as a commodity. [?] By sharing, parking spaces can be used more effectively and information can be distributed more easily, which reduces the aforementioned parking problems. It is taken advantage of the fact that most private parking spaces are only used by certain groups of motorists at particular times and thus remain unused for a large part of the time and that those parking times for the different groups often do not overlap.[?] A supermarket, for example, needs its parking spaces mainly during the day when the supermarket is opened, while a hotel needs the parking spaces at night for the overnight guests.

3.2 Intelligent Parking Systems

To give an overview for intelligent parking systems we are introducing a categorization of those systems proposed by Susan Shaheen in Smart parking management field test: A bay area rapid transit (bart) district parking demonstration.[?] Shaheen divides intelligent parking systems into five categories.

Guidance Information Systems Guidance information systems are the simplest type of intelligent parking systems. The aim of such systems is to point the car park seeker to the nearest available parking spaces. In many cities this is done by simple displays on the streets, which point to car parks and show the number of free parking spaces available in real time.

Transit-Based Information Systems Transit-base information systems are just like the previously mentioned GIS pure information systems. However, special importance is given in this case to draw attention to parking spaces and to direct

to parking spaces that are linked directly to local public transport. In most cases such parking spaces are called 'Park+Ride' and allow free parking, for example. The main objective of these systems, in addition to reducing the traffic of those seeking parking, is to reduce any traffic in the city centres. The aim is to make public transport more attractive and useable in order to prevent air pollution and congestion in the cities themselves.

Smart Payment Systems With smart payment systems operators try to minimize their maintenance costs. Setting up, emptying and repairing coin-operated machines at parking lots is often an expensive business. Operators of Smart Payment Systems therefore try to forgo using these machines. Instead, the customer pays, even contactless, with smartcards or via his own smartphone. This usually has advantages for both sides. For the customer the payment process is accelerated and he can avoid using cash and at the same time the operating costs are reduced.

E-parking By e-parking, Shaheen referred to a special system that was under development in 2005 by a research consortium. The aim of the concept was to combine the various smart parking ideas. The system includes advanced guidance information systems, smart payments methods, reservation options and connectivity to other digital services. More generally, we consider e-parking to be systems that use all means of digitalisation to make the parking experience better and more effective for the user in all possible aspects.

Automated Parking Automatic parking is a system in which the user simply drives the car to a general drop-off point and later picks it up from there. The system takes over the remaining transport route of the vehicle from the delivery point to the actual parking lot. Such systems also exist as manual versions, then called valet parking.

3.3 Shared Parking

Shaheen, like many others, pays little to or no attention to the topic of shared parking. However, according to the above definition, shared parking is a kind of intelligent parking system. Overall, there are hardly any scientific papers on this subject. There are already a few implementations of the idea, but these are either limited to certain cities or have hardly any users, which renders them useless. In addition, these systems are operated by private companies and therefore no information about the security within the system is publicly available.

Of the few scientific papers that exist, most are concerned with the allocation and pricing of parking spaces to potential tenants.

Shao et al. [?] and Yu et al. [?] propose to use integer linear programming to maximize profits. Both studies show that their proposed strategies, in contrast to 'first-come-first-serve' or 'first-book-first-serve' strategies, lead to higher parking space utilisation and greater profits for operators of shared parking systems. Yu et. al. refer in their work only to the sharing of parking spaces of shopping malls at night. This also implies that, unlike our system, there is no fluctuation in the number of parking spaces over time which their system would not be able to handle.

Xu et al. [?] extend a classic matching mechanism (top trading cycles, ttc) for the parking spaces so that cash flow is allowed. They assumed that each participant owns at least one parking space and participates with this parking space in the ttc mechanism. For situations where participants are unable to rent out their own parking space or do not wish to rent another parking space, the mechanisms '(price-compatible) top trading cycles and deals (TTCD)' and 'price-compatible top trading cycles and chains (PC-TTCC)' are proposed as extensions to ttc. These allow, in addition to the already possible exchange of parking space in ttc, the flow and exchange of money within the system. These mechanisms allowed participants to save costs, but under certain circumstances negative platform's payoff would occur.

Xiao et al. [?] assign parking spaces to tenants in their work using a double auction mechanism. They present the "demander competition padding method (DC-PM)" auction mechanism based on a slot allocation rule and a transaction payment rule. Both demanders and bidders place their bids by specifying the parking price and time and the mechanism determines the allocation of the parking spaces. Since this mechanism can lead to distorted social welfare, a "modified demander competition padding method (MDC-PM)" auction mechanism is also proposed, which is however inferior to the first mechanism in terms of its usefulness for the participants. They show that both mechanism "can realize asymptotic efficiency as both demanders and suppliers approach infinity", but do not take the spatial requirements of the participants into account during the allocation process.

Zhang et al. [?] try to improve the competitiveness of shared parking lot providers compared to traditional parking lots. They use the Hotelling model to compare the product differentiation of the two parking systems and use equilibrium analyses to evaluate the parking prices.

In all these papers, the utilisation of parking spaces or the social welfare in the system improves compared to the 'first-book-first-serve' mechanism, but the systems do not provide the user with the same level of transparency. In our shared parking system, users should be able to see exactly which parking spaces are available, choose from them and be sure to be able to use the selected parking

space. This can only be implemented with 'first-book-first-serve' and the systems mentioned above are not able to provide this functionality.

In addition to parking space allocation through a matching mechanism, Kim et al. [?] propose an infrastructure to help drivers find the optimal parking space. The infrastructure consists of Road Side Units (RSUs), which communicate with the cars of parking seekers and Fog servers, which monitor the availability of each parking lot. RSUs communicate directly with Fog servers in their vicinity and with each other through a Roadside Cloud. Users can now make a request while driving, which is then accepted by the RSUs. Through communication between the RSUs and with the Fog Servers, a matching problem with all car park seekers is established, solved and the allocation of parking spaces is sent back to the vehicles through the RSUs.

The big problem for this system are the expensive implementation and maintenance costs which arise because a lot of hardware and sensors have to be installed. The operator has to provide the RSUs, the Fog Server, sensors for each parking spot and the Roadside Cloud and every participant requires appropriate channels of communication on his car to connect to the RSUs. Our system, on the other hand, will do without expensive hardware to reduce maintenance costs and enable anyone to participate.

A number of shared parking platforms have already been implemented, but in very few cases they have become established. Internationally, for example, justpark.com or mobypark.com offer this service, in Germany there are providers like parkplace.de and parklist.de. All these platforms offer private individuals the opportunity to offer their own parking spaces, but the supply is very limited.

At parkinglist.de, even in large German cities such as Munich, there is almost exclusively only information about public car parks or offers for monthly rented parking spaces from car park operators, but hardly any private offers.

At parkplace.de there is also only a small number of almost exclusively long-term parking spaces available. Moreover, this platform does not offer the possibility to automatically perform payments. Instead, payment-specific matters have to be agreed on with the landlord separately.

At justpark.com, which originates from England, there are almost exclusively public parking garages available in London, for example. However, many car parks also offer the possibility of making reservations.

At mobypark.com you can find some listings from private persons. However, Mobypark is not widespread and is only really represented in a few cities, such as Amsterdam and Paris.

However, none of the providers discloses information on how they can protect their systems and users from fraud and how they generally deal with fraud. This is presumably because, as Mobypark itself reports on its website, parking spaces

are hardly rented for short periods of time like a few hours, but mainly for longer periods (several days to several months). In addition, most of the rented parking spaces are not publicly accessible and key cards or the like must be exchanged prior to and at the end of the rental process. These conditions already lead to a reduction of fraud possibilities. In our application, however, it should also be possible to rent freely accessible parking spaces, even for short periods of time. This creates new opportunities for fraud, which, unlike in the scientific papers mentioned above, we will address in this paper.

3.4 Trust Systems

In order to distinguish malicious and genuine users in our system, we will use a trust system. Since there is a large amount of scientific work and discussion in the area of trust systems, we will not develop any new methods ourselves and will instead fall back on already matured methods, adapt them to our system and thereby make them better fit our system.

Trust systems are used in many different forms for many different purposes. Therefore, we have sought to find methods developed for systems that are similar to ours. Finally, we will use a method from the area of "Trust Management and Reputation Systems in Mobile Participatory Sensing Applications". These are systems in which participants use smartphones or similar devices to send information that they received at various locations through the sensors of their devices to a backend server. We will later show how trust methods of such systems can be easily mapped to our trust system. Mousa et. al. wrote a survey on this topic in 2015, which gives a fundamental overview of this area of trust systems[?]:

In all these systems, trust is used to assess the reputation of individual users. The calculated reputation is therefore a value for the trustworthiness of the participants and the quality of their contributions. Mousa classifies trust/reputations systems according to methodology, type of distribution and anonymity of participants.

Methodology Either a Trusted Platform Module (TPM) or a reputations score can be used as a methodology for trust assessment. TPM are hardware chips that are installed in the devices of the participants and are therefore not suitable for our system. In the "Future Work" section, trust assessment using TPMs is discussed briefly. For our system, on however, the use of a reputation score, which is calculated directly from the behavior of the participant, is suitable. For the computation of this score the participants' latest activities and his old score are used.

Type of distribution The distribution in a trust system can be either central, collaborative or a combination of both. Since our trust score should be hidden from the participants and we possess a trust server, a centralized structure is suitable for us. In this case the trust server receives all information, i.e. the contributions and feedback of the participants, from which it is able to calculate the score of all participants. In collaborative systems, however, each participant computes a separate score for all other participants.

Anonymity of participants Finally, anonymous and non anonymous systems can be distinguished. In anonymous systems the anonymity of the participants is preserved. This is neither necessary nor useful in our system, as each participant must provide certain personal data to access our system.

In the section x.y.z.z reputation module we will use a trust system corresponding to the above established classification and modify it for our purposes.

The use of such a trust system however allows different attack types which are described in more detail in section x.y attacker model. In the security analysis we will finally show the robustness of our system against these attacks.

4 System Model and Requirements

Now that an overview of the topic has been provided, the next part will deal with a basic system model and the requirements that will be imposed on the design. The system model gives an overview of the different parties that exist in our shared parking model and their relationships with each other. The subsequent requirement analysis is divided into functional and security requirements. Special attention is paid to ensuring that the design is secure. For us security requirements do not only refer to standard security aspects such as encryption, user authentication and authentication, but in particular to the goal of designing our shared parking system in a way that it has the ability to handle fraud independently. For instance, the system should prevent and punish individual users from scamming other users or the system. Various forms and examples of such fraud will be considered and evaluated. The system design presented in Chapter ?? will adhere to these requirements. We will now start of with a high-level overview of the shared parking system.

4.1 High-level Overview

This section gives a high-level overview of the proposed shared parking system and deals with the basic capabilities of the system. The shared parking system adheres to the definition supplied at the beginning. It offers a platform for

an online marketplace where both private and business users can rent and lease parking spaces. End users have the possibility to install an app and participate in the marketplace using their smartphone. This also clarifies the basic capabilities of the system. Every user can offer his own parking spaces for specified periods of time on the platform for rent. Users can also view other users' listings, choose from them and rent those parking spaces. In this system, particular importance is given to handling fraud. The subsequent requirements define the exact framework under which the design was developed.

4.2 System Model

The following system model, which is depicted in figure xy, shows the various parties in our **shared parking system SPS**. On the one hand, there is exactly one **operator** or **service provider SP**. The *SP* is responsible for the implementation of the design, operating the *SPS* and making the systems services available to the users. The **users U** are those who utilize the *SPS*. They can be divided into two roles, whereby a user can be part of both roles at different points in time. On the one hand, there are **landlords** who own a private **parking space** and want to rent it out over a certain period of time. On the other hand, there are **tenants** who are looking for a free parking space to rent for a specific time period.

This work covers fraud in *SPS* in detail. In the **case of fraud**, users take on new roles. On the one hand there is the **adversary**, who is committing fraud, on the other hand there usually is an **injured party**, which has a disadvantage caused by the fraud case.

As mentioned in ?? we will utilize different modules that deal with those fraud cases. In each of those modules users once again assume different roles.

The rating module deals with **ratings**. For each rating there is one user who issues, called the **rating user** and one user who receives, called the **rated user**.

The reporting module deals with **reports**. One report belongs to one case of fraud. For each report there is one user who issues, called the **reporting user** and one user who receives, called the **reported user**.

The verification module verifies reports and the corresponding fraud case. Users who verify the reports are called **inspectors**.

In the reputation module, **contributions** from all **participants** of one fraud case are processed. The participants include the reporting user, reported user and the inspectors. A participants' contribution is the information he provides about the fraud case.

All roles are described in more depth in the detailed description of the modules.

To-Do: Figure of system model

4.3 Functional Requirements

The following will cover the main functional requirements in detail.

Low deployment cost and of the shelf hardware Neither the operator nor the user should need any additional and expensive hardware. For getting the system to work the user just needs a off-the-shelf smartphone and the operator only needs to provide enough computational power to get the server running. There will be no sensors installed at the parking spots, no additional hardware used for confirming parking time or the likes of that.

Good Performance and usability The implementation of the basic features and use cases should be intuitive to use. First time users should be able to book or rent a parking space in a few clicks. Performance of all use cases should be such that there are no loading times longer than 3 seconds between each action of a user.

Automatic Processing Where possible, the *SPS* should be prevented from requiring manual intervention by the *SP*. The system should be able to run all basic use cases should and may only request manual intervention in exceptional cases. On the one hand this should lead to a fast processing of all inquiries and on the other hand reduce the costs for the *SP* stemming from employing people for manual tasks.

Streichen: Neutral Balance The operator or the system has different costs and revenues. Thus, the operator can either make a profit or lose money. We impose the condition on the system that in the long run, the revenues at least cover or even exceed the costs. This means that there may be time spans in which the costs are greater than the revenues, but if you take a look at the system in the long term, the revenues must surpass the costs.

More to come hi

There are also areas that do not place any functional requirements. There are aspects to which no special significance is attached in this work. This includes, for example, all legal matters. On the one hand, this work is in the field of computer science and the author does not have the fundamental knowledge to evaluate and judge legal matters. On the other hand, the legal aspect of a shared parking system involves many and various legal factors, which would exceed the scope of this work. Nevertheless, attention was paid during the design to carry

out as little legal interference as possible in order to avoid legal difficulties during the implementation.

4.4 Adversary Model and Assumptions

4.4.1 Adversary Model and Classes of Fraud

When we talk about fraud, we mean a user's ability to scam other users or the system using its normal functionality in order to gain a profit. In many cases, this has a negative impact on other users. There are numerous possibilities for fraud in the system, some of which are mentioned as follows. We will categorize all fraud cases into three different classes. As mentioned in ?? we refer to a user who commits fraud as a adversary. Users who are affected by fraud in a negative way will be called injured or injured party.

Class 1 The first class of fraud cases are the result of the general nature of a shared parking system. This class consists of fraud cases where a tenant has booked a parking space, but it is not available when he arrives. There could be different possibilities as to why the parking space is not available, examples of which are the following.

- adversary parks in a car park, which another user has booked for this time.
- Tenant overstays his booked parking time.
- Landlord rents out a parking space, but remains on it himself.
- Landlord offers the same parking space multiple times.
- Landlord offers a parking space that does not exist.

To deal with these class 1 cases, we will introduce various modules, such as a report module and a reputation module, which enables detection of the attacks described above. As we will see later, Class 1 cases can be handled by our system without manual intervention.

Class 2 The reputation module helps to deal with class 1 cases of fraud, however it itself is subject to a new series of fraud cases. We group these new fraud cases into class 2. All well known attacks in this area are described in [?] and are applied to our system as follows:

- **Corruption attack:** An adversary unintentionally or deliberately sends corrupted data and information. An example of a corruption attack is when

a reporting user enters an incorrect license plate number for the reported user.

- **On-off attack:** An attacker alternates between malicious and genuine behavior. This is particularly difficult to detect when the transition appears to occur randomly.
- **Re-entry attack:** An adversary tries to white wash his trust value by rejoining the system with a new identity. In our system this corresponds to a new registration of an existing user.
- **Collusion:** Several adversaries act together and execute attacks collectively. Joint attacks have a greater impact on the system and are harder to detect than acting independently.
- **Sybil attack:** This is an extension of the re-entry attack. An adversary creates multiple identities or accounts within the system and utilizes them in a collusion attack.
- **Reputation lag exploitation:** The adversary uses the period between his first attack and the evaluation and corresponding downgrading of his trust value to launch a large number of attacks.
- **GPS spoofing attacks:** The adversary tampers with the GPS location of his device.
- **Unfair ratings:** The adversary does not rate another user according to his genuine opinion.
- **Bad mouthing or negative discrimination attack:** The adversary raises false accusations or assigns a bad rating to a genuine user.
- **Ballot stuffing or positive discrimination attack:** The adversary does not report misbehaviour or assigns a good rating to a malicious user.

We shall see that Class 2 cases can also be dealt with mostly without manual intervention by using a robust reputation module.

Class 3 While the first two classes cover the main and most frequent cases of fraud, there are others. Class 3 cases differ from the other classes in that it will not be possible to solve them without manual intervention. These are exceptional cases that occur only in low frequency. Examples are given below.

- Landlord rents out parking spaces that they do not own themselves.

- Tenant cancels booking of a parking space without paying.

For this class of fraud cases, we will also show methods of dealing with them.

Most of these attacks can only be performed by registered users. In contrast, however, there is a case of fraud, which can also be carried out by persons outside the system. A parking offender who occupies a booked, private parking space does not necessarily have to be registered in the system. Dealing with this problem is not the main focus of this work, but a suggestion on how to deal with it when realizing the system will be provided in section ??.

4.4.2 Assumptions

In the following we specify assumptions that apply to our system.

Attacker collaboration We assume that adversaries can not only attack the system on their own, but collaborate with other adversaries to jointly start and carry out attacks. We will show that even under this assumption the *SPS* can protect itself from the attacks described above.

Sufficient parking spaces We assume that there are always enough parking spaces available for compensation. The assumption is reasonable, since if the system has low occupancy levels, a suitable free parking space can always be found. If this is not the case, there will always be landlords who offer their parking spaces for a particularly expensive and overpriced costs. These places will still be vacant even in situations where almost all capacities are occupied. These are parking spaces that are still available for use as compensation for injured parties, even when there is a high level of occupancy.

Majority assumption We assume that the majority of users in our system are acting according to the rules, meaning that there are always a maximum of 50 percent adversaries. Even under the *attacker collaboration* assumption it is reasonable to believe that there will be plenty of genuine users that use the system for parking and not for fraud as this is a typical assumption for distributed systems (e.g. [?], [?], [?]). One can expect that the majority of people are genuine, especially when they know that offenses are being punished hard. We further offer an opportunity to report inadvertent misdemeanors, in order to prevent major penalties, by yourself, which should prevent inadvertent misdemeanors by genuine users.

4.5 Security Requirements

The following will cover the main security requirements in detail. On the one hand, requirements are made on information security and on the other hand, requirements are made on how to deal with fraud in the shared parking system.

The system must be able to handle all these cases of fraud mentioned in the adversary model above successfully. In the following we will define what we mean by 'successful handling'. We therefore set requirements that must be met for the various cases of fraud.

Fraud Prevention The design of the system should be such that fraud is fundamentally prevented. If this is not possible in all cases, it should be ensured that it is not worth the effort for the adversary or that adversaries are quickly discovered. This is where the second point comes into play.

Fraud Detection The system should be able to detect fraud. This includes, on the one hand, the simple recognition of fraud and, on the other hand, the ability to correctly identify the adversary. The latter is particularly important for the next point.

Fraud Punishment The design should be such that a adversary can be punished by the system. After a adversary is detected, the system should always be able to penalize him. Punishment can also support the first requirement by acting as a deterrent for fraud.

Fraud Compensation Finally, the system should be able to compensate an injured party. If injured are compensated, there is no need for genuine users to worry about fraud and thus enhances the attractiveness of the system.

There are also areas that do not place any special requirements on security. Even without a shared parking system, attacks on the private parking infrastructure are possible. One person can for example park their car in another person's private car park. The injured party has two options, both of which can have strong disadvantages for him. He could call the towing service, hope that the adversary has not left by its arriving time or he will have to bear the expensive towing fees himself. The other option is to do nothing and look for a new parking space for yourself. Such problems, which are already present, do not need to be solved. However, it can be seen as a added bonus to prevent such problems with good design choices for the shared parking system.

5 Design

With the requirements being defined, the next part is devoted to the actual design. As with the requirements, we will first have a look at the functional features before dealing with fraud.

5.1 Functional features

The functional features will be presented as use cases.

User registration The registration process is rather simple. A user registers by entering his e-mail address, his first and last name and a password. After confirming the data, a mail is sent to the specified address, in which the user must verify whether it actually is his address.

From this point on he can log into the system at any time using his mail address and password.

Balance system Each user has a credit balance that can be charged and paid out using standard payment options. Various events, such as renting or hiring out a parking space, lead to an increase or decrease to the credit balance. The precise calculation of the credit is defined below for the individual events. If a user's balance is negative, they will no longer be able to use the system. His credit must be recharged before his next action.

Parking Space management Before you can rent a parking space, you have to register it in the system. To do this, enter the address and select the exact location on a map so that the coordinates (latitude and longitude) of the car park can be stored in the system. In addition to registration, these parking spaces can also be removed.

Renting out a parking space If a user wants to rent out a parking space, he inserts an offer. He selects one of his parking spaces, specifies the time at which the parking space is available and the price that the rental will cost per hour.

Renting a parking space If a user wants to rent a parking space, first a map is presented. There his location can be displayed or he can search for a city/address to be centred. All currently available parking spaces are displayed on this map. In addition to the location, the user can specify the time at which he wishes to book the parking space. If he does, only parking spaces that are available for the entire time period are displayed.

More to come

5.2 Security features

In order to prevent fraud different modules are introduced. There is a rating module, reporting module and a verification module, all of which are explained below. These modules work together and provide information for a reputation module that assesses the trustworthiness of each user. At the end of this section an example is given on how the design utilizes those systems to handle fraud.

5.2.1 Rating Module

The design of the *SPS* includes a rating module, allowing users to rate each other. After each rental agreement has been concluded, both tenants and landlords have the opportunity to give the other party a rating. The rating consists on the one hand of a points/star rating between 1 and 5, and on the other hand of a rating text. The user should share his experiences made with the other party and evaluate how satisfied the agreement went for him. The ratings each user received are displayed on his user page and in all of his offers. The rating system is designed to help eliminate adversaries and encourage users to be as friendly and courteous as possible.

5.2.2 Reporting Module

The next important component is the reporting module. Whenever a user detects a rule violation by another user, he has the possibility to report it. A distinction is made between the reporting user, the reported user and the inspectors, who will be explained in the section ?? ?? later on.

Firstly, there is the possibility of reporting a violation that occurs during a rental contract. On the overview page of the contract, you can choose from various reporting reasons, all of which refer to this contract. If the user finds a suitable violation from the list, he selects it and the system can respond to it without manual intervention. For example, tenants always have the option "The rented parking space is occupied by another vehicle", in which the license plate of the car, which occupies the parking space, is also provided. In a similar way, there is the option "Tenant has overrun parking time". for landlords. For all reasons that are not explicitly mentioned, meaning that the system cannot resolve them automatically, there is a general reporting function that requires manual intervention from the provider.

If the system receives a report, it can be sure that it is always a case of fraud. Either the user reports a real fraud case, which in this case has to be fraud class 1, or it is a false report from an adversary, which he expects to benefit from.

This would again be a case of fraud, which can be classified in fraud class 2. The system must therefore first check the truth of the report. The verification module exists for this purpose and is explained below. After the verification module has identified who the adversary is, the necessary steps for punishment can be taken. The section ?? ?? explains in detail how exactly the system acts to compensate or punish users.

5.2.3 Verification Module

Another important security tool is the verification module. It allows the system to check who has really committed a rule violation. If a rule violation situation occurs and another user cannot occupy his legally entitled parking space, he will report it to the system. Since the report may also be false, the system must first verify who the real offender is: the reported user or the reporting user. For this purpose, other users are called upon to head to this place and to make a contribution to this case. Those users will then be called inspectors. The truth of the report can be verified by comparing the contributions of the reported user, the reporting user and the inspectors. The inspectors will receive a bonus on their balance for their service.

The module works in several steps. After a report is received, potential inspectors are selected and requested. They confirm whether they will verify the event or not. Afterwards, they verify the event and the system evaluates the contributions.

Since the location of the inspectors must be taken into account when selecting them, users have the option of deciding whether they want to participate in the verification system or not. There are two ways for all participating users to provide the system with location information.

- The user allows the app to retrieve GPS data from the mobile device. The GPS data is only collected if a report needs to be verified. Directly after viewing the data, it is calculated whether the user is in the area, and the GPS data is deleted again.
- The user selects an area on a map in which he generally agrees to verify reports. GPS data is not necessary in this case.

A list of all users that are eligible for verification is created. This means only users who have agreed to participate in the verification system and who are either in the vicinity of the location of the event themselves or whose verifying area contains the location of the event are chosen. This list will now be sorted in a way that the users who are most suitable are at the top and users who are less suitable are at the bottom. Suitability is determined using a 'connection'

score, which is calculated for all available inspectors. This score indicates the independence between the inspector and the reporting and the reported user. A low score is a sign of great independence. 'connection' is a score, which represents the number of common events between the inspector and the reporting user plus the number of common events between the inspector and the reported user in a certain period of time (e.g. the last six months). The following events increment the score between user 1 and 2 by a single point:

- User 1 books a parking space from user 2.
- User 1 reports user 2.
- User 1 verifies a report from user 2.

Next, the users from the list are asked whether they are able and willing to verify this event. This request is not sent to all users at the same time. Based on the reputations score (see below) it is determined how many inspectors n are needed to solve the case and requests are sent to the first $n \cdot 3$ inspectors of the list.

The selected users will receive a notification on their smartphone indicating the location of the event for review. They then confirm or reject whether they are able and willing to perform the verification within the next 30 minutes. If a user refuses or does not respond within 15 minutes, he will be removed from the list and the next user not yet contacted will receive a request. After n users have accepted the request, all other requests are revoked.

As soon as a user has accepted the request, he will receive exact information about the parking space. He then moves to the inspection point and states whether this parking space exists and which car is parked on it according to the license plate.

In the last step the contributions must be evaluated. This is done using the reputation system described below. For this purpose, the reputation of both parties to the conflict is assessed. As seen down below, not the trust values of the two parties are compared, but the trust of the contributions by those parties, in which both the trust value and other parameters have an influence. These other parameters include, for example, the inspectors' contributions and the ratings the two parties have received.

The verification module returns the user whose contribution possesses the lower trust value as the adversary.

In addition to the above, the verification module asks the reported user for feedback on this case. The reported user can either confirm or contradict the report of the reporting user. This is intended as a means of self-reporting by the reported user, which can reduce his punishment and does not negatively affect his reputation score. This is a good way for a genuine user to admit inadvertent misconduct. If it does, no more inspectors need to be requested and the verification module returns the reported user as the adversary. However, even if the reported user admits the fraud with his contribution, the case is passed to the reputation module to update the reputation scores of the participating users.

5.2.4 Reputation Module

The last part of the security features is the reputation module. It stores a reputation score R_u for each user u , which is updated using the other security modules and saved in the central database, hidden from the users. The value indicates the trustworthiness of a user. It is needed to decide who is right in the event of a conflict. Based on this reputation score, a separate trust value $Trust(C)$ is determined for each contribution C taking other parameters into account. In all cases where the contributions of the reporting and the reported user do not match, this value is used to decide which contribution is considered as the truth.

Our reputation module is instantiated on the Dynamic Trusted Set Based Reputation System (DTSRS) developed by Mousa et. al. [?]. It utilizes state-of-the-art trust systems presented in [?] and offers better evaluation results than those. DTSRS was created for use in Mobile Participatory Sensing Applications, but is also suitable for our purposes. It utilizes an ever-changing trusted set of participants to calculate the ground truth and compares all contributions to this ground truth value. In the calculation of its trust score it further takes ratings and a proximity score into account. In the following we will explain the system, our changes to it and the mapping of our parameters to those of the DTSRS.

The reputation module uses a trust score to evaluate participants and their contributions. The trust of a contribution $Trust(C)$ is the probability that contribution C is correct. The reputation score R_u of a user u is the synthesized probability that his contributions are and will be correct. In *SPS*, every user can be a participant of any fraud case, but the number of participants changes from case to case. After all, the participants in each case are the reported user, the reporting user and all inspectors. Each participant of a case submits exactly one contribution for this case. The contributions are therefore the report by the reporting user, the confirmation or denial of the report by the reported user and the verification contributions from the inspectors.

After every report received by the system, R_p of all participants p involved in the report is updated. In addition, the trust of the contributions of the reported user and the reporting user is evaluated. The trust is updated in these five steps:

Contribution Evaluation In the first step, the contributions C_p by participants p are evaluated and a contribution evaluation value θ_C is calculated. In our case, the contributions are the report by the reporting user, the confirmation or denial of the report by the reported user and the contributions from the inspectors resulting in n contributions. In the original DTSRS, the values of the contributions are within a certain range. For our module we assume the special case that each of these contributions has exactly one of two values: (0) the contribution is consistent with the contribution of the reporting user or (1) the contribution is not consistent with the contribution of the reporting user. The two values occupy exactly the margins of the original range. This allows us to continue calculating in exactly the same way as DTSRS.

DTSRS first arranges C_1, \dots, C_n in descending order based on corresponding reputation scores R_1, \dots, R_n . The first $m < n$ contributions of that order form the trusted set. The mean $\mu(C)$ of those m contributions of the trusted set is then regarded as the ground truth. The module then calculates the contribution deviation d_p for every contribution as depicted in the equation below and normalizes it to range $[0,1]$. If d_p^no , which represents the normalized value, equals 0 it is implied that C_p equals the ground truth. A normalized deviation of 1 implies a complete contradiction to the ground truth. With the help of an exponential distribution and d_p^no , a score between 0.37 and 1 is assigned, which corresponds to the contribution evaluation value θ_p of the contribution C_p .

$$\mu(C) = \frac{\sum_{i=1}^m C_i}{m} \quad (1)$$

$$d_p = \text{abs}(C_p - \mu(C)) \quad (2)$$

$$d_p^no = \frac{d_p - \min_{i=1}^n(d_p)}{\max_{i=1}^n(d_p) - \min_{i=1}^n(d_p)} \quad (3)$$

$$\theta_p = \exp^{-d_p^no} \quad (4)$$

Feedback Processing In this step, the feedback each participating user received is taken into account. The module calculates a aggregated feedback score α_p for each participating user p . In DTSRS, the user can receive feedback for each contribution. There, feedback refers to exactly one contribution and is then discarded. In *SPS*, users receive feedback through the rating module. Thus, received ratings originate not only from participant, but from all users of the

shared parking system. In order to avoid using a rating more than once in the calculation of a user's reputation score, *feedback processing* utilizes exactly those ratings that have been entered since the user's reputation score was last updated. For the calculations, the ratings $F_r(p)$ (1 to 5 stars) of a participant p given by rating user r are normalized to the range $[0,1]$. Only ratings of users r whose reputation score is greater than that of the rated participant p are used ($R_r > R_p$). The individual ratings are weighted according to the reputation score of the rating user and the average is calculated, which also falls within the range $[0,1]$. The following equations describe the calculation, where F is the number of rating users for each participant and FH the number of rating users whose reputation score is higher than the reputation score of the rated user.

$$F_{r,Eval}(p) = F_r(p) * R_q \quad \forall r \in 1, 2, \dots F, \text{ where } R_r > R_p \quad (5)$$

$$\alpha_p = \frac{\sum_{r=1}^{r=FH} F_{r,Eval}(p)}{FH} \quad (6)$$

A Temporal Proximity Factor In DTSRS, the proximity factor measures the distance between the participant and the sensing area, as a larger distance is more likely to result in inaccurate measurements. In *SPS*, the distance has no influence on the measurements, as anyone making a contribution must be at the location of the event. Instead, however, the temporal distance β_p has an effect on the measurements, because the more time passes, the greater the probability that the situation at the location of the event will change, which results in inaccurate measurements. For this reason, we are dealing with a temporal proximity factor σ_p in the range $[0,1]$, which is calculated using an inverse Gompertz function from the temporal distance. For the reported and reporting user the temporal distance is 0 and the proximity factor is therefore maximum at 1. For every other participant β_p describes the distance between the report and the participant's contribution in minutes.

$$\sigma_p = 1 - a * e^{-be^{-c\beta_p}} \quad (7)$$

Trust Mapping The trust of a contribution $Trust(C)$ is calculated exactly as with DTSRS. We take a weighting $\sum_{i=1}^4 W_i = 1$, the three already calculated values contribution evaluation θ_p , aggregated feedback α_p and the temporal proximity factor σ_p and as a fourth value the reputation score R_p of the participant p and calculate the weighted average of these four values:

$$Trust(C) = W_1 * \theta_p + W_2 * \alpha_p + W_3 * \sigma_p + W_4 * R_p \quad (8)$$

The reputation score of a new participant is first set to 0, meaning he or she must first earn the trust of the system.

Reputation Update When updating the reputations value of a participant p , DTSRS only uses the contribution evaluation θ_p and not the complete trust $Trust(C)$ of his contribution. The reason for not using the proximity factor σ_p is, that it cannot be influenced by the user and does not depend on the genuineness of the user. The temporal proximity factor is also not included in the recalculation of the reputation score in our module, as users further away from the location of the event take more time to reach the location, regardless of whether they are malicious or genuine.

The new reputation value R'' , as we can see in a moment, will be calculated on the basis of the old reputation value R_p . Because of that we don't use $Trust(C)$ for the calculation, as R_p would therefore have two times the impact. Mousa et. al. also argue that with the omission of R_p recent behaviour has a direct influence on the reputation score.

However, they do not provide any reason why aggregated feedback α_p should not be included in the reputation update. Since the feedback in our system is independent of the individual contributions, but clearly provides a indication about the trustworthiness of a user, we will include it.

We update the reputation value consecutively with the contributions evaluation θ_p and the aggregated feedback α_p . For both values there is a threshold τ_θ and τ_α . If the previously calculated values θ_p or α_p of a participant p is higher than τ_θ or τ_α , the reputation value is increased by $e_{+,\theta}$ or $e_{+,\alpha}$. If it is lower, the reputation value is decreased by $e_{-,\theta}$ or $e_{-,\alpha}$. $e_{-,\theta} > e_{+,\theta}$ and $e_{-,\alpha} > e_{+,\alpha}$ applies to punish adversaries more severely. The update is described by the following equations:

$$R'_p = \begin{cases} R_p + e_{+,\theta} & \text{if } \theta_p \geq \tau_\theta \\ R_p - e_{-,\theta} & \text{if } \theta_p < \tau_\theta \end{cases} \quad (9)$$

$$R''_p = \begin{cases} R' + e_{+,\alpha} & \text{if } \alpha \geq \tau_\alpha \\ R' - e_{-,\alpha} & \text{if } \alpha < \tau_\alpha \end{cases} \quad (10)$$

If the value is above 1 or below 0 after the update, it is reset to the corresponding value (1 or 0).

5.2.5 Sanctioning and Compensation

With these basic systems it is now possible for the system to punish adversaries and compensate the injured parties.

After a report has been received, the reporting user will first be provided with a new parking space, regardless of whether it later turns out that he was in fact the fraudster. For this purpose, the system selects the nearest parking space available for the entire duration of the reservation of the original parking space. The system

now books this parking space for the reporting user, takes over the costs, and sends the information about it to his smartphone. The System however, will not pay the price which is officially offered for the replacement parking space, but will instead pay the average price of parking spaces in the area. This is calculated as the average price of all bookings of the last x months in a maximum distance of y kilometers. On registration, all users agree to make their parking space available as a replacement parking space for the average parking fee, if required. The reporting user can now use the replacement parking space and the owner of the replacement parking space will be fairly remunerated.

After a replacement parking space has been provided, the already explained verification and reputation modules are used to determine who the adversary is.

After the real offender is identified, he will be punished. The punishment of fraudsters is primarily enforced by a fine. The penalty will be deducted from his credit balance. This penalty will also be used to cover the costs of the fraud. Costs arise on the one hand when compensating the injured party and on the other hand when paying the inspectors. The penalty is calculated from the sum of the inspectors' credit bonuses and the remuneration of the landlord of the replacement car park. Thus there are no costs for the system when the offender is successfully identified.

5.2.6 Example

Figure ?? on page ?? shows fraud within the system and how to deal with it. The procedure is explained in more detail below.

User u_1 books a parking space that belongs to user u_2 in advance. The *SPS* transfers the parking fee from u_1 to u_2 , while deducting the system fee and providing it to *SP*. However, when u_1 arrives to occupy the parking space, he realizes that another user (u_3) is blocking the parking place. u_1 then uses the reporting system to report u_3 with his license plate. The system receives the report and then assigns other independent users with the task of verifying the report. $u_{i1}...u_{i3}$ are the first to accept the verification request and head to the location of the event. In the meantime, u_1 will be provided with a new parking space (parking space of u_4). He can park on it free of charge for the same time period he booked parking spot of u_2 . The *SP* will cover the parking fee f_p for now and add it to the balance of u_4 . After arriving at the location of the event, $u_{i1}...u_{i3}$ confirm the licence plate of u_3 and thus the report made by u_1 . They get paid with the verification fee f_v by the *SP*. The reputation module evaluates all contributions, which leads to deciding u_3 as the adversary. The reputation scores $R_{u_1}, R_{u_2}, R_{u_{i1}}, R_{u_{i2}}, R_{u_{i3}}$ are updated, leading to a score decrease for u_3 and an increase for the other users. In the end the *SPS* deducts the fine from the balance of u_3 which is the sum of f_p and $f_v * 3$. Thus paying for the inspectors

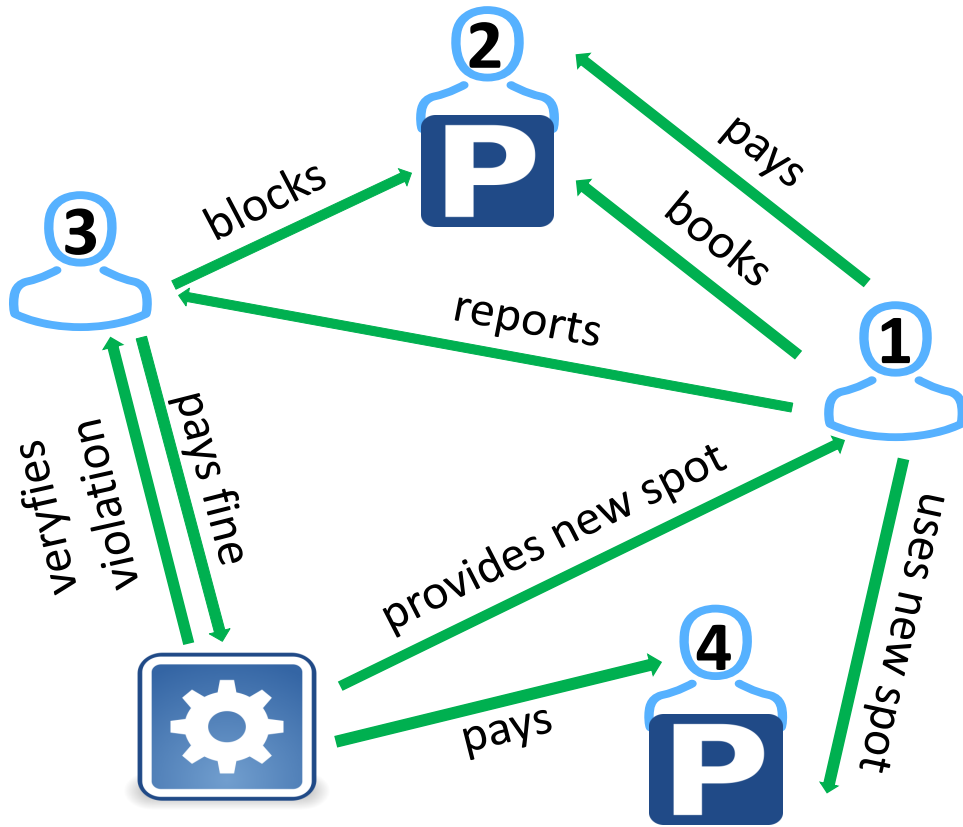


Figure 1: A possible example for the handling of rule violations

service and the replacement parking space for u_1 .

Of course, it is also conceivable that user 3 has not occupied the parking space at all, but user 1 still reports him. In this case, the verifiers will report that the parking space is actually free. Now the system penalizes user 1 with a sum that covers the costs for the newly allocated parking space (of user 4) and the costs for the verifying users.

Another possibility is that user 3 has actually occupied the parking lot but has already left until the users arrive for verification. In this case, like above, the system would identify user 1 as the adversary, although user 3 violated the rules. It also imposes the fine on User 1. He now has the opportunity to challenge the fine. If he does, the confidence score comes into play. If both users are new users or neither of them has a particularly bad or good score, no fair decision can really be made and the costs incurred are borne by the system. In all other cases, when one user is clearly more trustworthy than another, the system decides in favor of the more trustworthy user.

6 Security Analysis

In the following, the design of the system is evaluated using the security features set up in the requirement analysis. We look at each point individually. But first we will note a few premise that we believe apply to our system. They will help us perform the security analysis.

Without established data and numbers on fraud in our system, we cannot work with probabilities or the like to show that our design meets the specified requirements. As far as possible, we will therefore work without such data. In places where this is no longer sufficient, we will show that the error probability of our methods approaches 0 with increasing size of the system.

6.1 Fraud Detection

Fraud detection includes two components: the simple recognition of the fact of fraud as well as the ability to correctly identify the adversary.

Fraud recognition Fraud recognition is exclusively achieved by the report system. As soon as a user detects a rule violation, he can report it and the system is informed. It is sufficient to recognize fraud by which another party would be harmed, because this would not be any different without the Shared Parking System. A parking offence only causes problems if another user, who is authorized to park, wants to park on it and with our system the other user now has the ability to report it. Thus, parking spaces with a shared parking system always provide a better possibility for fraud detection than without it.

Adversary identification The ability to correctly identify the adversary is provided by the verification and reputation module.

First, we show that our reputation module can generally distinguish between malicious and genuine users, while also defending itself against the attacks described by class 2 fraud cases.

Our module is based on the DTSRS from [?], in which Mousa et. al experimentally evaluated their system using different simulations and obtained the following results. They showed that their system "accurately assesses the quality of participants' contributions" and "clearly identifies adversaries even if the number of colluding adversaries reaches 60% of the total number of participants" and furthermore defends itself against corruption, On-Off and collusion attacks.

Since we only map our system parameters to the parameters of Participatory Sensing, we can adopt the evaluation and the resulting properties of DTSRS. We assume that the majority of users in our system are non-malicious (??), which means an adversary or a collaboration of adversaries can control at maximum 50% of users. This remains below the 60% with which DTSRS can still successfully deal with. In the following we discuss how the *SPS* can either fend off attacks on a reputation system presented in the adversary model or mitigate them in the environment of our system in some other way.

1. **Corruption attack:** DTSRS was shown to be resilient to the corruption attack, and our system inherits this property from DTSRS.
2. **On-off attack:** DTSRS was shown to be resilient to the on-off attack, and our system inherits this property from DTSRS.
3. **Re-entry attack:** As Mousa et. al. states in [?] "this attack was not considered by any of the current reputation based trust systems in participatory sensing" and thus no automated solution has been found in state-of-the-art reputation systems. If we want to protect our system against this type of fraud, we could add a manual function: the registration credential information would have to be checked manually either in general during registration (for example, with proof of identity card) or during registration of a user's license plate (in cooperation with the authorities). This ensures that people can only register with their real identity and thus only once.
4. **Collusion attack:** DTSRS was shown to be resilient to the Collusion attack, and our system inherits this property from DTSRS.
5. **Sybil attack:** For the sybil attack the same applies as for the re-entry attack.
6. **Reputation lag exploitation:** In our system there is a time interval between the individual events of a user which cannot be influenced by the user himself. For malicious behavior as inspector, the user must first be selected as such. If parking spaces are blocked, the system is only informed when the user is reported, etc. It is not possible for a user to launch a large number of attacks in a very short time. Thus, the results of a fraud are already accounted for in the reputation score of the user when the next fraud case is evaluated.
7. **GPS spoofing attacks:** We do not use gps functions when receiving contributions because we already know the position of an event. If an inspector checks a fraud case, it is a parking space that is not available.

The system already knows the position of this parking space. If an inspector does not travel to the location of the incident and simply guesses a result of his verification, it is called a corruption attack.

8. **Unfair ratings:** DTSRS implemented methods to mitigate the unfair rating attack, and our system inherits this property from DTSRS.
9. **Bad mouthing or negative discrimination attack:** "Bad mouthing [...] attacks are not addressed by any of existing trust systems in participatory sensing" explained in [?]. However, the effect of this attack is also mitigated in our system. In contrast to DTSRS and Participatory Sensing, users cannot arbitrarily give feedback in our system. To rate another user, you must first book or rent a parking space. While renting out one cannot freely choose the target of his bad mouthing attack. While renting one has to pay the parking price for every single rating he'd like to issue. In addition, the ratings of each individual user are publicly available and other users can easily recognize if the adversary discriminates against individual users. In addition, users who are unfairly rated badly will also return a bad rating in most cases.
10. **Ballot stuffing or positive discrimination attack:** Ballot stuffing is similar to bad mouthing. Even if two users always rate each other highly, fees must be paid to the operator for each related rental transaction.

Because the reputation module can correctly distinguish between malicious and genuine users and it is used to check each report, we can decide whether the reporting or the reported user is the adversary. The trust of a contribution used for this is actually better than simply comparing the trust scores of the conflicting users, since in addition to the trust score, the contributions of the other users for this event and the proximity factor are also included there.

As shown, the module recognizes the adversaries of class 1 cases (reported user is a adversary) and recognizes and protects itself against class 2 cases. The exceptional cases of class 3 are briefly discussed below:

6.2 Fraud Punishment

The ability of fraud punishment is based on the ability of fraud detection. If a adversary is correctly identified, the system is able to penalize him. By registering, users have agreed to the rules of the system and can therefore be fined if they violate those rules. In the simplest case, the system receives the money by deducting it from the offender's credit balance and adding it to the system's credit balance. It is therefore obvious that the system receives the entire penalty

and can use it for other purposes. If the adversary's credit balance becomes negative through deduction and is not recharged within a certain period of time, the operator receives the penalty in the form of an invoice for contractual penalty to the user.

6.3 Fraud Prevention

The system achieves fraud prevention through two methods. On the one hand, adversaries are deterred and on the other hand, fraud is uncovered and adversaries are thus eliminated.

adversaries are deterred by the public rating system and the well-known possibility of reporting. Anyone who intends to cheat considers twice whether they want to accept the resulting disadvantages, such as a poor rating or exclusion from the system. In addition, there is the fine you have to pay if you are discovered to be a adversary. Since these fines are invariably higher than the cost of using the system properly, fraud is not profitable.

As explained in the paragraph 'Fraud Dedection', adversaries are discovered at the latest in the long run. On one hand, all other users notice from the adversary's rating that there is something wrong. The adversary is thus eliminated by the fact that nobody wants to book a parking space of his any more and he will no longer be able to book a parking space from users with manual confirmation, because most users only conclude contracts with users from whom they assume that everything will go as planned, i.e. users with a good rating. On the other hand, adversaries are eliminated because they are excluded by the system. Anyone who has been discovered too often as a adversary or whose reputation score falls below a certain threshold will be banned.

6.4 Fraud Compensation

Also, fraud compensation can easily be performed once the system has the ability of correctly identifying the fraudster. Further up we have shown that the system has this ability. Once the fraudster has been clearly identified, it is also clear that the other party to the conflict is the injured party which must be compensated.

But even before this is clear, the reporting user was provided with a new parking space if required. The costs are initially borne by the system. However, once fraudsters and injured parties have been assigned, the system will recover these costs, as shown in the 'Fraud Punishment' section. This means that there are no costs for the system if the adversary can be clearly identified. So if the reporting user was really the injured party, he got a replacement parking space without paying anything for it; he was compensated. If the reporting user was

the adversary, however, he had to pay a fine afterwards, meaning that he paid for his replacement parking space by himself.

7 Implementation

The system is implemented according to the client-server model. Users utilize a smartphone as a client, while the operator provides a server on which a central database is hosted. Also running on the server is a web service that can be requested by the clients via a REST API. The web service forwards the request to the database system and also returns the response back to the clients. An Android app was developed for the clients using Java. The web service is also written in Java and runs on a Tomcat server, while a MySQL server has been set up for the database system. In total, the implementation consists of three relatively independent parts, which will be examined in more detail below. The final realization of the *SPS* is of course not part of this work or the implementation, but the last part of this section should give some comments on a possible realization. These include, for example, how to deal with the problem described at the end of section x of adversaries not registered in the system.

7.1 Database

A relational database management system was used to store user and parking data. We chose the open source system MySQL, which, as the name suggests, uses SQL as the query language. The actual database system, whose ER diagram is depicted in figure x, consists of the tables user, car, parkingspace, parkingoffer, parkingtrade and rating.

7.2 Java Web Service

A web service in Java was developed to access the database. It runs on a Tomcat server and offers a RESTful interface. This interface is used by the android app to access the database. The java open source framework Jersey is used to develop the REST API. This framework helps to address JAX-RS, the Java API for RESTful Web Service. For this purpose, annotations are used in the code that can be found in the java package javax.ws.rs. These annotations are used to specify the path and the request method, which are both part of an http request, by which a java method may be addressed.

7.3 Java Android App

The main part of the implementation was the development of the android app, which is also written with java. The app uses Gradle as a build automation system. The development of the UI largely adheres to the guidelines set by Google regarding material design. This ensures that new users can find their way around the app in no time. The app itself has no internal database and mainly uses JSON objects to manage the user and parking data, since the data is returned in this form by the Web service. These objects are only stored in the random access memory and the web service is queried again if necessary.

7.4 Comments on the realization of the *SPS*

Creation of a basic user structure The problem of many shared parking systems mentioned in section x is that they could hardly generate any users at all. One of the reasons for this is probably that such a system must first overcome a certain barrier before it becomes attractive for users. Nobody registers with a system to lease parking spaces if there is no one renting them out. And no private user takes the trouble to insert a parking space that nobody will ever book. The pure C2C model is actually not really suited for the launch of such a system. However, businesses can help to overcome this barrier. Various businesses such as hotels, cinemas, doctors or supermarkets usually only need their parking spaces for a certain period of the day. If one has already contacted some of the above mentioned before launching the system and they decide to insert their parking spaces, the system already launches with a large supply of parking spaces. This supply can then also ensure that demand increases and the system further grows.

Solution for adversaries not registered in the system In section x the problem has already been raised that it is not possible to punish parking offenders directly via the *SPS* if they are not part of the system. However, there is also a solution that can be implemented when realizing the system.

Without the *SPS*, adversaries in private parking spaces can only be punished by requesting a towing service and having the parking space cleared. There are, however, two problems: On the one hand, all costs that the adversary has to pay are transferred to the towing service and cannot be used by the *SPS* to pay for a replacement parking space or inspectors. On the other hand, it can happen that the adversary has already left the parking space when the towing service arrives and the caller has to cover the costs.

However, the situation that several towing services are competing with each other in larger cities can be taken advantage of. It should be possible to negotiate a deal with one of the towing services stating that all towing operations under the

SPS will be carried out by this service provider. In return, the service provider pays part of its extremely high towing fee back to the *SPS*, which can then be used to pay for the replacement parking space and the inspectors. In addition, service provider should not charge any costs in the event that the adversary has already left the car park. In return, every report, that contains a license plate which is not registered in the system, is forwarded directly to the towing service. In this way, both parties can benefit from the deal.

8 Future Work

To conclude, we will present some further aspects of this topic which either could not be elaborated within the time frame of the study or which would exceed the scope of this work. These can be carried out in further work.

Extending the functionality of the app

Testing the usability

Testing the performance

Testing the reputation system

Using a Trusted Platform Module as Trust System

9 Conclusion

10 Acknowledgments

I would like to thank Prof. Dr. Alexandra Dmitrienko and the city of Würzburg, who both supported me in creating this work.

References

- [1] Gildas Avoine, Felix Gärtner, Rachid Guerraoui, and Marko Vukolić. Gracefully degrading fair exchange with security modules. In *European Dependable Computing Conference*, pages 55–71. Springer, 2005.

- [2] Zinaida Benenson, Felix C Gärtner, and Dogan Kesdogan. *Secure Multi-Party Computation with Security Modules*. RWTH, Department of Computer Science, 2004.
- [3] Institute for Transportation and Development Policy. What is shared parking? 2014.
- [4] Muftah Fraifer and Mikael Fernström. Investigation of smart parking systems and their technologies. In *Thirty Seventh International Conference on Information Systems. IoT Smart City Challenges Applications (ISCA 2016), Dublin, Ireland*, pages 1–14, 2016.
- [5] MY Idna Idris, YY Leng, EM Tamil, NM Noor, Z Razak, et al. Car park system: A review of smart parking system and its technology. *Information Technology Journal*, 8(2):101–113, 2009.
- [6] Victoria Transport Policy Institute. Shared parking. 2015.
- [7] Oanh Tran Thi Kim, Nguyen Dang Tri, Vandung Nguyen, Nguyen H. Tran, and Choong Seon Hong. A shared parking model in vehicular network using fog and cloud environment. *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 321–326, 2015.
- [8] Hayam Mousa, Sonia Benmokhtar, Omar Hasan, Lionel Brunie, Osama Younes, and Mohiy Hadhoud. A reputation system resilient against colluding and malicious adversaries in mobile participatory sensing applications. In *The 14th Annual IEEE Consumer Communications & Networking Conference (CCNC 2017)*, 2017.
- [9] Hayam Mousa, Sonia Ben Mokhtar, Omar Hasan, Osama Younes, Mohiy Hadhoud, and Lionel Brunie. Trust management and reputation systems in mobile participatory sensing applications: A survey. *Computer Networks*, 90:49–73, 2015.
- [10] Mehrdad Nojoumian and Douglas R Stinson. Efficient sealed-bid auction protocols using verifiable secret sharing. In *International Conference on Information Security Practice and Experience*, pages 302–317. Springer, 2014.
- [11] Susan Shaheen. Smart parking management field test: A bay area rapid transit (bart) district parking demonstration. 2005.
- [12] Chaoyi Shao, Hai Yang, Yi Zhang, and Jintao Ke. A simple reservation and allocation model of shared parking lots. *Transportation Research Part C: Emerging Technologies*, 71:303 – 312, 2016.

- [13] Donald C Shoup. Cruising for parking. *Transport Policy*, 13(6):479–486, 2006.
- [14] Haohan Xiao, Meng Xu, and Ziyu Gao. Shared parking problem: A novel truthful double auction mechanism approach. *Transportation Research Part B: Methodological*, 109:40–69, 2018.
- [15] Su Xiu Xu, Meng Cheng, Xiang T.R. Kong, Hai Yang, and George Q. Huang. Private parking slot sharing. *Transportation Research Part B: Methodological*, 93:596 – 617, 2016.
- [16] Jingwei Yu, Qian Wan, and Zhibin Li. Optimal distribution model of shared parking space in large shopping malls at night. In *IOP Conference Series: Earth and Environmental Science*, volume 153, page 032006. IOP Publishing, 2018.
- [17] Wei Zhang and Shuaian Wang. Pricing of shared-parking lot: An application of hotelling model. In Giuseppe De Pietro, Luigi Gallo, Robert J. Howlett, Lakhmi C. Jain, and Ljubo Vlacic, editors, *Intelligent Interactive Multimedia Systems and Services*, pages 310–317, Cham, 2019. Springer International Publishing.