Secure Free-Floating Car Sharing for Offline Cars

Alexandra Dmitrienko
Institute of Information Security
ETH Zurich, Switzerland
alexandra.dmitrienko@inf.ethz.ch

Christian Plappert
Fraunhofer SIT
Darmstadt, Germany
christian.plappert@sit.fraunhofer.de

ABSTRACT

In this paper, we present a new access control system for free-floating car sharing, which achieves a number of appealing features not available in the state-of-the-art solutions. First of all, it does not require online connection for cars, and, therefore, allows car sharing providers to expand their services to areas without reliable network coverage (e.g., with blind spots). Second, the solution is compatible to RFID cards – the most commonly deployed authentication tokens in car sharing, and can be deployed on standard mobile platforms with various hardware features. Third, it is fully compatible with off-the-shelf cars and does not require any intrusive modifications to car's internals.

These new properties can be achieved due to a novel system design which deploys two-factor authentication and combines an RFID card (the real one or emulated in software) with a "soft" authentication token stored on a mobile platform. Such a combination increases security of the solution, preserves backward compatibility to RFID technology and enables great flexibility in protection of authentication secrets on the mobile platform. To demonstrate such a flexibility, we present a platform security concept which can be instantiated in various deployment options and provides the means to achieve best possible security given available hardware.

We implemented our solution on Android and instantiated the platform security concept in three different deployment options. We evaluate security of our solution and report performance measurements.

Keywords

Car Sharing; Access Control; DESFire EV1; BLE; NFC

1. INTRODUCTION

Within the last decade, the worldwide market for car sharing has grown exponentially [1, 2] and the rapid development of car sharing solutions is drastically changing the transportation landscape, especially in metropolitan areas [3]. Car sharing membership has grown from 2012 to 2014 by 170% to 4.8 million, with an increasing tendency for 2016 [4]. Car sharing solutions are well accepted by customers due to their inherent benefits: They offer users anytime access to a pool of vehicles for short-term use and enable mobility without the costs of a private car. Although the concept of individual ownership of vehicles is not going to vanish completely, it is increasingly replaced by the car sharing's on-demand mobility approach [5].

In general, car sharing is utilized with either a station-based or a free-floating business concept. While the more traditional station-based car sharing relies on fixed stations where a car needs to be returned after the booking period to the same parking lot where it was taken from, the more flexible and faster growing free-floating model [6] allows the user to pick up and leave cars anywhere in a vendor-defined area. On the downside, however, the free-floating model requires online connection for cars and is limited to locations with reliable network coverage.

With the by now omnipresent smartphones that come with an already built-in variety of communication interfaces like NFC, Bluetooth, GSM and GPS, car sharing solutions became even more convenient for the end user. By utilizing mobile services, users can conveniently search their surroundings for bookable cars and even use their smartphone app as a car key to open booked cars (e.g., *Car2Go* car sharing solution [7]).

However, current car sharing systems suffer from various shortcomings. In particular, the more convenient free-floating car sharing requires online communication with cars during the car opening process. On the other hand, network coverage and quality of data services significantly vary in different locations, and even fully covered urban areas are known to have blind spots with poor signal reception [8]. Hence, for the sake of interoperability and reliability car sharing providers often opt for station-based usage model which is less attractive to end users. Furthermore, car sharing solutions utilizing smartphone apps to download and store electronic car keys impose additional security risks to their customers: Attackers may try to intercept electronic car keys on transit, while they are transferred from the car sharing provider to the users, or when they are stored on users' smartphones. Attackers may also attempt to hijack user accounts to be able to book cars on behalf of legitimate users. Security incidents of that kind already affected Uber [9], a car ride sharing service with more than 8 million of users around the world, – many users reported they were charged for rides they have never taken [10, 11]. Further, compromised Uber accounts were proposed for sale on dark web markets [12] for

CODASPY'17, March 22 - 24, 2017, Scottsdale, AZ, USA

ISBN 978-1-4503-4523-1/17/03…$15.00

DOI: http://dx.doi.org/10.1145/3029806.3029807

as little as $1, which indicates that the attack is rolled out on a large scale. Similar attacks are very likely to affect car sharing services, as long as service providers do not address new security threats.

The state-of-the-art approach to harden mobile platform security is to leverage isolated (secure) environments, where apps can execute security sensitive operations (e.g., encryption, signing, etc.) in sub-routines referred to as trusted applications, applets or trustlets. Such environments can be established on top of mobile secure hardware, such as processor-based security extensions [13, 14] (also referred as Trusted Execution Environments, TEEs) and dedicated secure co-processors [15] (also known as Secure Elements, SEs). However, despite the fact that mobile secure hardware is widely deployed today [16], their secure environments are controlled by various stakeholders and normally cannot be used by third party apps. While generally paid access to secure hardware is possible, the process to obtain it is cumbersome [17]. Hence, one has to consider scalable approaches to platform security which can utilize such secure hardware if accessible, while being able to provide secure alternatives otherwise.

In this paper, we aim to tackle shortcomings of state-of-the-art car sharing solutions and propose a new car sharing system which provides a unique combination of properties. It (i) supports for offline cars and, hence, can be used in locations with less reliable network connection and even without it (e.g., in underground garages). Furthermore, it (ii) accurately addresses new security threats and, at the same time, (iii) it can be used with various off-the-shelf mobile platforms with no extra requirements to hardware and installed software. Additionally, (iv) our solution provides interoperability to standards commonly used in car sharing solutions today, and can even be used with off-the-shelf cars without any intrusive modifications. In particular, we make the following contributions:

- We analyze functional and security requirements for offline car sharing systems (Section 2) and design the first smartphone-based car sharing solution for offline cars (Section 3). Our solution leverages two-factor authentication of users and separate delivery of both authenticators to clients and their isolated handling on client platforms in order to harden security against new attack vectors. It provides great flexibility in integration with mobile platforms and backward compatibility to RFID cards – the most commonly deployed authentication tokens in car sharing. Furthermore, it enables range of alternatives for protection of user authenticators on client side, which allows a car sharing provider to select the best option depending on capabilities of user's hardware and achieve the best security possible per user. We provide security analysis of our solution (Section 4).

- We implemented a proof-of-concept prototype for Android smartphones and demonstrated its flexibility by showing several alternatives for protection of user authenticators ranging from the Secure Element (SE) provider hosted on a Mifare DesFire EV1 contactless card to Mifare DesFire EV1 card emulated on top of mobile secure hardware or entirely in software when hosted by user's (trusted) smartwatch (Section 5). For system evaluation we augmented private cars with the prototyped car lock using a *car key proxy* approach which does not require any intrusive modifications to cars. Our evaluation includes performance measurements of user authentication for various instantiations of SE provider.

To summarize, our solution is the first to provide such a set of properties which improves state of the art for free floating car sharing systems with respect to security and supported functionality.

2. SYSTEM MODEL AND REQUIREMENTS

In this section we provide a high-level overview of our solution, define our system model and adversarial capabilities and analyze security and functional requirements.

2.1 High-level Overview

The core design feature of our solution is a two-factor authentication in order to get access to the cars where the authentication factors are downloaded and handled separately. In a nutshell, the user needs to present two authenticators to the car lock in order to successfully pass authentication. The first authentication factor is created during the user registration process, while the second one is downloaded during car booking. Since both authenticators are obtained in separate sessions, it is more challenging for the adversary to compromise both of them. Further, our solution also handles both authenticators on client side in isolation from each other, which even further increases the burden for the attacker, and, at the same time, enables flexibility for the defender in arranging their protection.

While two-factor authentication is widely used today, e.g., in online banking and for login verification by Internet service providers, to the best of our knowledge it is not used in car sharing applications. Moreover, our scheme is distinguishable from other two-factor authentication schemes, as it combines contactless Radio-frequency Identification (RFID) cards, which are a de-facto standard for access control and widely used in car sharing applications, with "soft" cryptographic tokens – the approach which enables a flexible integration with mobile platforms. Moreover, our solution extends the state-of-the-art in the field of access control solutions for car sharing systems by providing appealing features which are not available in alternative solutions, such as offline user authentication, compatibility with legacy cars and various deployment options.

2.2 System Model

Our system model is depicted in Figure 1 and involves the following entities: the Car Sharing Provider (CSP) $\mathcal{C}$, a car to be shared equipped with the Lock $\mathcal{L}$, and a User $\mathcal{U}$. For simplicity reasons and without loss of generality, we consider a single user and a single car in our system model, which can be easily scaled to a pool of cars and many users.

Each $\mathcal{U}$ possesses a client platform consisting of two execution environments, the mobile host $\mathcal{H}$ and a Secure Element Provider (SEP) $\mathcal{S}$, which are isolated from each other. Typically a SEP cannot communicate with other entities directly, but such a communication is mediated by the host $\mathcal{H}$ which is used as a proxy. However, depending on the deployment option, SEP may or may not have a dedicated user interface.

$\mathcal{C}$ is a car sharing provider which defines access rules to cars, i.e., specifies which $\mathcal{U}$ is allowed to access which $\mathcal{L}$.

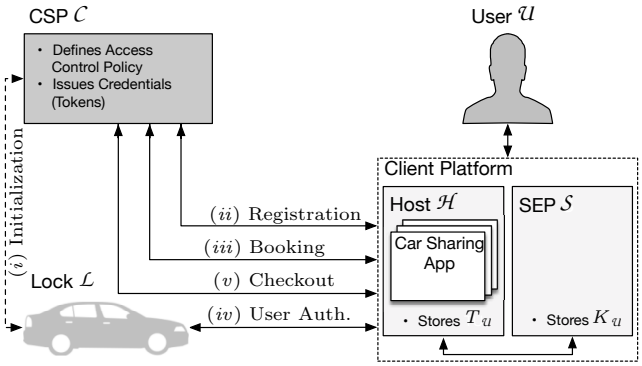

Figure 1 – System Model

$\mathcal{L}$ is either a regular car key lock interfaced with car internals or instantiated using a car key proxy approach as we describe in details in Section 5.1. $\mathcal{L}$ requires access to location and timing information, which is provided either by car's GPS sensor and timer in former case or is available on the car lock itself when latter approach is used.

Authentication factors. The two authentication factors are the user specific key K_u created during user registration and the car access token T_u which is downloaded during the car booking process. On the client side the token T_u resides within the host environment $\mathcal{H}$, while the user-specific key K_u is hosted by $\mathcal{S}$.

2.3 Adversary Model and Assumptions

We define three distinct adversary classes which will be used in Section 4 to elucidate the security of our solution in various deployment options.

Class 1 Adversary. Our first class adversary has full control over the communication between the $\mathcal{H}$, $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{L}$. This means that it can eavesdrop, modify, insert, delete, relay and re-route the protocol messages. However, it cannot compromise the communication channel between $\mathcal{U}$ and $\mathcal{H}$, $\mathcal{H}$ and $\mathcal{S}$, and any of the communication end-points.

Class 2 Adversary. Our second class adversary has all the possibilities of the first class adversary and, additionally, it can compromise user's host $\mathcal{H}$ and gain access to all information stored on it, e.g., credentials, at any time but during user registration process. Further, the communication channel between $\mathcal{H}$ and $\mathcal{S}$ can also be compromised.

Class 3 Adversary. Our third class adversary is similar to the second class, with the only difference that it can compromise $\mathcal{H}$ at any time including user registration.

We exclude relay attacks [18] on the communication between a mobile device of the user and the car lock which are not specific to our car sharing system, but due to the local communication interfaces we rely upon (Near Field Communication (NFC) and Bluetooth Low Energy (BLE)). These attacks can be mitigated by distance bounding techniques [19, 20], which can be easily incorporated into our scheme once they are available on off-the-shelf mobile platforms. Moreover, contactless RFID cards, the most commonly deployed authentication tokens in car sharing, are also susceptible to such relay attacks [21]. We also do not consider denial of service (DoS) attacks, which prevent users from accessing the service. An attacker with full control over communication channels can always disrupt communication and prevent token download, while compromised host can always prevent a car sharing app from launching or delete stored information including downloaded tokens. These attacks are not specific to our car sharing system, as they can be launched against any mobile application. Furthermore they do not provide any monetary benefits to adversaries and, hence, unlikely to be applied.

2.4 Security Requirements

Our main security objective is to prevent unauthorized access of users to the car sharing service. In particular, only a user $\mathcal{U}$ which possesses both a user key K_u and the car access token T_u should be granted access to the respective lock $\mathcal{L}$. To achieve this objective, a number of security requirements should be fulfilled which we detail in the following.

SR1: Well-established Crypto. Our first security requirement is to rely on open and well-established crypto primitives and algorithms for user authentication, since closed and proprietary systems that are not available for evaluation to a broad security community are more likely to suffer from vulnerabilities. For instance, the common practice to use proprietary protocols in immobilizer systems has lead to their successful exploitations [22, 23, 24, 25], which strongly speaks against the "security by obscurity" approach.

SR2: Confidential Credentials. Our second security requirement is to ensure the confidentiality of authentication credentials (such as cryptographic keys and passwords), and even from users themselves. Otherwise the adversary can use phishing attacks to trick users to reveal their passwords, and later on use them for impersonation. Attacks of that kind are accountable for large scale hijacking and abuse of user accounts of the car ride sharing service Uber [10, 11].

SR3: Isolation. Third, we require isolation between trusted and untrusted components on the mobile platform and that only trusted components can access credentials in clear text. Otherwise, an attacker may deploy mobile malware which can infiltrate credentials from the mobile platform. For instance, mobile banking Trojans like ZeuS/ZitMo [26] use this approach to intercept verification codes sent by banks to the users' mobile phones.

SR4: Strong Credentials. Fourth, we require authentication credentials to be randomly chosen, uniformly distributed and have sufficient length. If not fulfilled, the attacker may have significant chances to succeed in dictionary attacks against user passwords and/or even brute-force cryptographic keys. For instance, in 1998 a 321-bit RSA key used by debit/credit cards of the French bank was factored by an individual [27].

SR5: SEP invocation authorization. Fifth, we require user authorization for every invocation of code executed within SEP $\mathcal{S}$. If security sensitive code executed within SEP is not authorized by the user, it could be triggered by malware rather than by the legitimate car sharing app which can then trigger user authentication on behalf of the user and succeed in user impersonation without actually learning authentication credentials. Such attacks were shown in the past [28] on payment applications such as Google Wallet.

2.5 Functional Requirements

Apart from the security requirements discussed above, we define the following functional requirements.

FR1: Offline Authentication. State-of-the-art car sharing solutions rely on connected cars, which limits their operational area to locations with reliable network connection. To overcome this limitation, we require an offline authen-

tication of users during car (un)locking which enables car sharing services to expand to areas with less reliable or even without any network connection.

FR2: Compatibility. Compatibility significantly increases chances for a successful deployment, as it enables recycling of existing hardware, infrastructure and technologies and, hence, results in reduced time and costs of development and deployment. Our compatibility requirements include: (i) interoperability with contactless RFID cards – the most common authentication token in car sharing solutions, and (ii) the ability to utilize various off-the-shelf mobile platforms with no extra requirements to hardware or system software.

FR3: Flexible Deployment. Our last functional requirement concerns various deployment options, which should achieve the best possible level of security for available user's hardware and preferable usability properties. When available, the car sharing provider may provide different deployment options to different customers, depending on the underlying hardware of their mobile platforms.

Overall, to the best of our knowledge, no other car sharing solution can fulfill similar requirements.

3. SYSTEM DESIGN

In this section we elaborate on system design by providing protocol specification and describing our design choices and deployment alternatives for the platform security concept.

3.1 Protocol Specification

The parties which we specified in our system model (cf. Section 2.2) interact in the following use cases (cf. Figure 1): (i) system initialization, (ii) user registration, (iii) car booking, (iv) user authentication, and (v) user checkout. Below we provide protocol specifications for each use case.

3.1.1 System Initialization

During initialization the car sharing provider initializes the car locks with cryptographic material and registers all the cars in its database. In particular, $\mathcal{C}$ initializes each car lock $\mathcal{L}$ with a unique car identifier $ID_{\mathcal{L}}$ and two cryptographic keys $K^{\mathcal{L}}_{Auth}$ and $K^{\mathcal{L}}_{Enc}$ over a confidential and authenticated out-of-band channel. For instance, this step can be performed by programming the car lock via local programming interfaces before the lock is installed into the car. The triple $\{ID_{\mathcal{L}}, K^{\mathcal{L}}_{Auth}, K^{\mathcal{L}}_{Enc}\}$ is also stored by $\mathcal{C}$ in its local database.

Furthermore, $\mathcal{C}$ initializes each $\mathcal{S}$ by creating a smartcard application AID with two empty data files $FID_{\mathcal{U}}$ and $FID_{\mathcal{L}}$, which will be later on used to store the user's identifier and car's location, respectively. Access to the application is protected by an application master key K_M while administrative access to the smartcard which protects against unauthorized operations, e.g., formatting the card and creating/deleting applications, is protected by a smartcard master key. The smartcard master key is kept secret by $\mathcal{C}$ and never shared with third parties.

3.1.2 Registration

Before using the car sharing service, $\mathcal{U}$ needs to create a user account and associate it with his or her client platform. This is done during user registration which is intended to establish the user-specific key $K_{\mathcal{U}}$ shared between $\mathcal{C}$ and $\mathcal{S}$ on the client side.

The user registration procedure consists of two phases. In the first phase, out-of-band communication is used to exchange data between $\mathcal{U}$ and $\mathcal{C}$. In particular, $\mathcal{U}$ submits his or her personal data (such as user name, e-mail address, post address, etc.), identifies payment method and submits the scanned copy of the driving license to the car sharing provider. In return, he or she receives either the pre-programmed smartcard initialized with the user-specific key $K_{\mathcal{U}}$, or one time password $OTP_{\mathcal{U}}$ which will be later on used to provision $K_{\mathcal{U}}$ into SEP remotely. We do not specify any particular way to establish such an out-of-band channel since similar registration procedures are commonly utilized by mobile applications and well-established techniques exist. For instance, user-specific information can be submitted from $\mathcal{U}$'s PC to $\mathcal{C}$ over a web form, a pre-initialized smartcard can be picked up in the office of a car sharing provider or sent per post[1], while the one-time password can be delivered via one time accessible link sent by email.

The second phase of *registration* (cf. Figure 2) is only necessary if the customer has received the one time password $OTP_{\mathcal{U}}$ in the first phase. It is initiated by $\mathcal{U}$ who sends his credentials *creds* consisting of the user name $ID_{\mathcal{U}}$ and $OTP_{\mathcal{U}}$, to the mobile host $\mathcal{H}$ (step 1), which, in turn, stores $ID_{\mathcal{U}}$ for future use and forwards *creds* over the established Transport Layer Security (TLS) channel to the car sharing provider $\mathcal{C}$ (steps 2-3). Upon receive, $\mathcal{C}$ verifies *creds* and, if correct, it generates the user-specific key $K_{\mathcal{U}}$ and reconfigures $\mathcal{U}$'s SEP to use $K_{\mathcal{U}}$ as an application master key for the AID application. In particular, $\mathcal{C}$ first sends a `select_app(`AID`)` command (step 4) and then authenticates with the application master key K_M (step 5). Afterwards, it selects the file with $FID_{\mathcal{U}}$ (step 6) and writes the identifier $ID_{\mathcal{U}}$ into it (step 7). Finally, it sends a command to change the application master key to $K_{\mathcal{U}}$ (step 8).

Note that the communication between $\mathcal{C}$ and $\mathcal{S}$ is always mediated by $\mathcal{H}$, which is omitted in Figure 2 and in other protocol figures for brevity. Further, depending on the deployment options which we discuss in Section 3.2, steps 1-3 of the protocol may involve SEP $\mathcal{S}$ instead the host $\mathcal{H}$[2]. Moreover, a successful authentication of an entity to SEP implies that all the subsequent communication is protected with a freshly generated session key, which is not explicitly shown to simplify protocol figures. For instance, in Figure 2 steps 6-8 are performed in a channel secured by the session key derived from the authentication in step 5. Finally, all communication between $\mathcal{H}$ and $\mathcal{C}$ is performed via the TLS channel established at the beginning of the protocol run.

3.1.3 Car Booking

After registration users are allowed to book cars for a self-defined period of time. Booking is done by using the *car booking protocol* depicted in Figure 3, within which $\mathcal{U}$ retrieves the car access token $T_{\mathcal{U}}$ and stores it on $\mathcal{H}$.

The protocol is initiated by $\mathcal{U}$, who indicates to $\mathcal{H}$ that he or she would like to perform car booking (step 1). In turn, $\mathcal{H}$ establishes a TLS session to $\mathcal{C}$ and sends user identifier $ID_{\mathcal{U}}$ (steps 2-3) to $\mathcal{C}$. Next, $\mathcal{C}$ authenticates the client by ensuring his or her SEP $\mathcal{S}$ has the knowledge of $K_{\mathcal{U}}$: First, it selects the car sharing application AID (step 4), then authenticates with the key $K_{\mathcal{U}}$ (step 5), then selects the file $FID_{\mathcal{U}}$ (step 6) and makes sure it can read the user identifier $ID'_{\mathcal{U}}$ from it

[1]This is a state-of-the art approach to deliver access cards used in car sharing today

[2]In particular, in a deployment option where SEP $\mathcal{S}$ features its own user interface

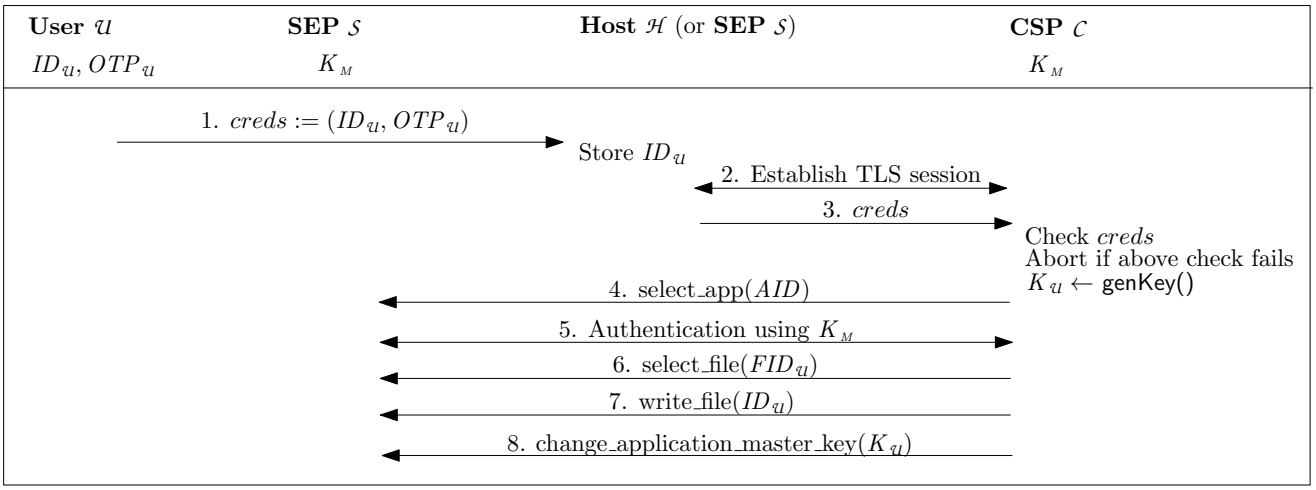

Figure 2 – Registration Protocol (second phase)

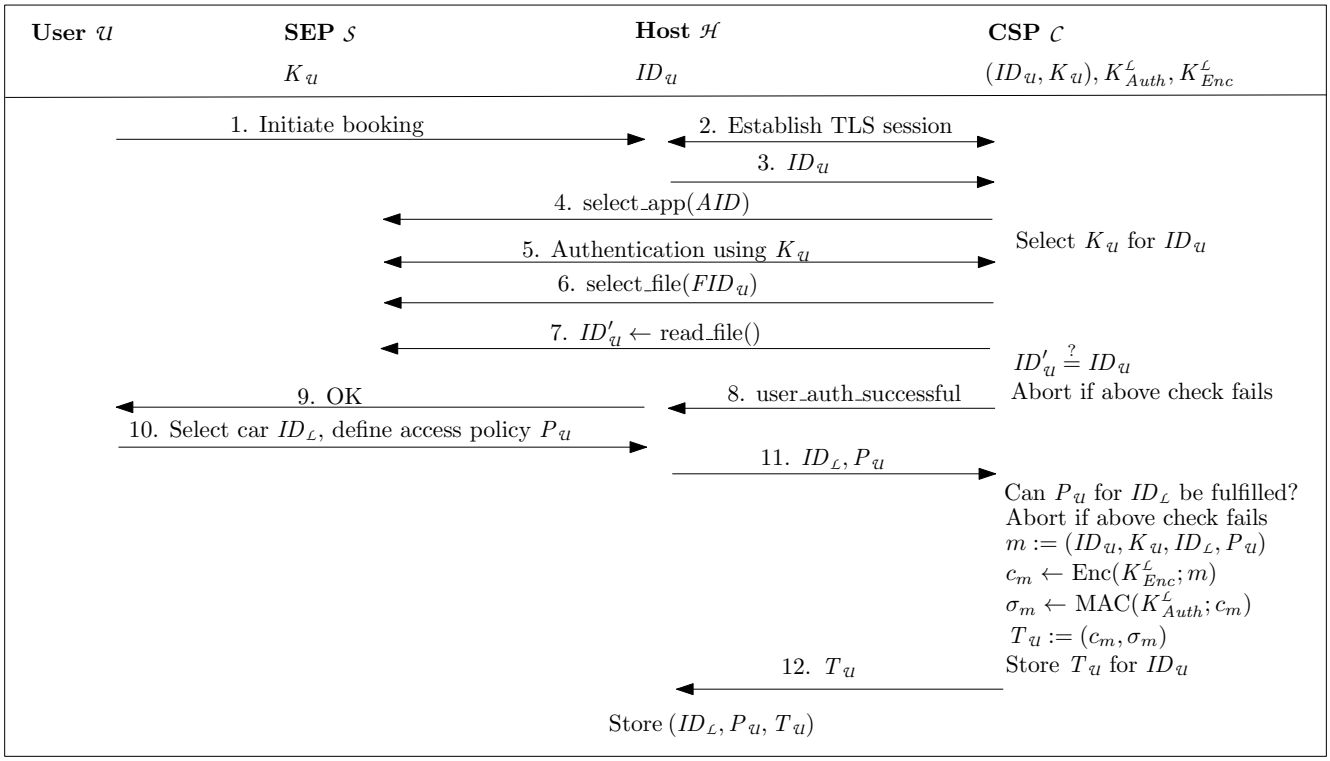

Figure 3 – Car Booking Protocol

(step 7). After a successful validation that the received $ID_{\mathcal{U}}$ from step 3 and the obtained $ID'_{\mathcal{U}}$ from step 7 are equal, $\mathcal{U}$ is notified of the successful user authentication (step 8-9). Once authenticated, $\mathcal{U}$ can book the car as follows: He selects the car with the identifier $ID_{\mathcal{L}}$ in the mobile app and specifies the desired access policy $P_{\mathcal{U}}$[3], which are then forwarded by $\mathcal{H}$ to $\mathcal{C}$ (steps 10-11). After $\mathcal{C}$ has verified that the desired access policy can be fulfilled for the respective car, it generates a message m which includes the user identifier $ID_{\mathcal{U}}$, the user-specific key $K_{\mathcal{U}}$, the car identifier $ID_{\mathcal{L}}$, and the access policy $P_{\mathcal{U}}$. The message m is then encrypted with the encryption key $K^{\mathcal{L}}_{Enc}$ and its signature σ_m is calculated using the key $K^{\mathcal{L}}_{Auth}$ and Message Authentication Code (MAC) algorithm. The resulting access token $T_{\mathcal{U}}$ consisting of the cipher c_m and the signature σ_m is sent to $\mathcal{H}$ (step 12), which stores it along with the car identifier $ID_{\mathcal{L}}$ and the policy $P_{\mathcal{U}}$.

[3]Typically access policy includes validity period of booking, but generally may include more sophisticated statements, e.g., maximum distance, location area, etc.

3.1.4 User Authentication

User authentication is required in order to unlock the car and lock it again after usage. With the *user authentication protocol* $\mathcal{U}$ can proof to $\mathcal{L}$ that he or she is in the possession of both authenticators, $T_{\mathcal{U}}$ and $K_{\mathcal{U}}$. Furthermore, $\mathcal{L}$ writes the car's location on $\mathcal{U}$'s SEP $\mathcal{S}$ and the timestamp so that they can be reported to $\mathcal{C}$ later on.

The protocol for *user authentication* is depicted in Figure 4.

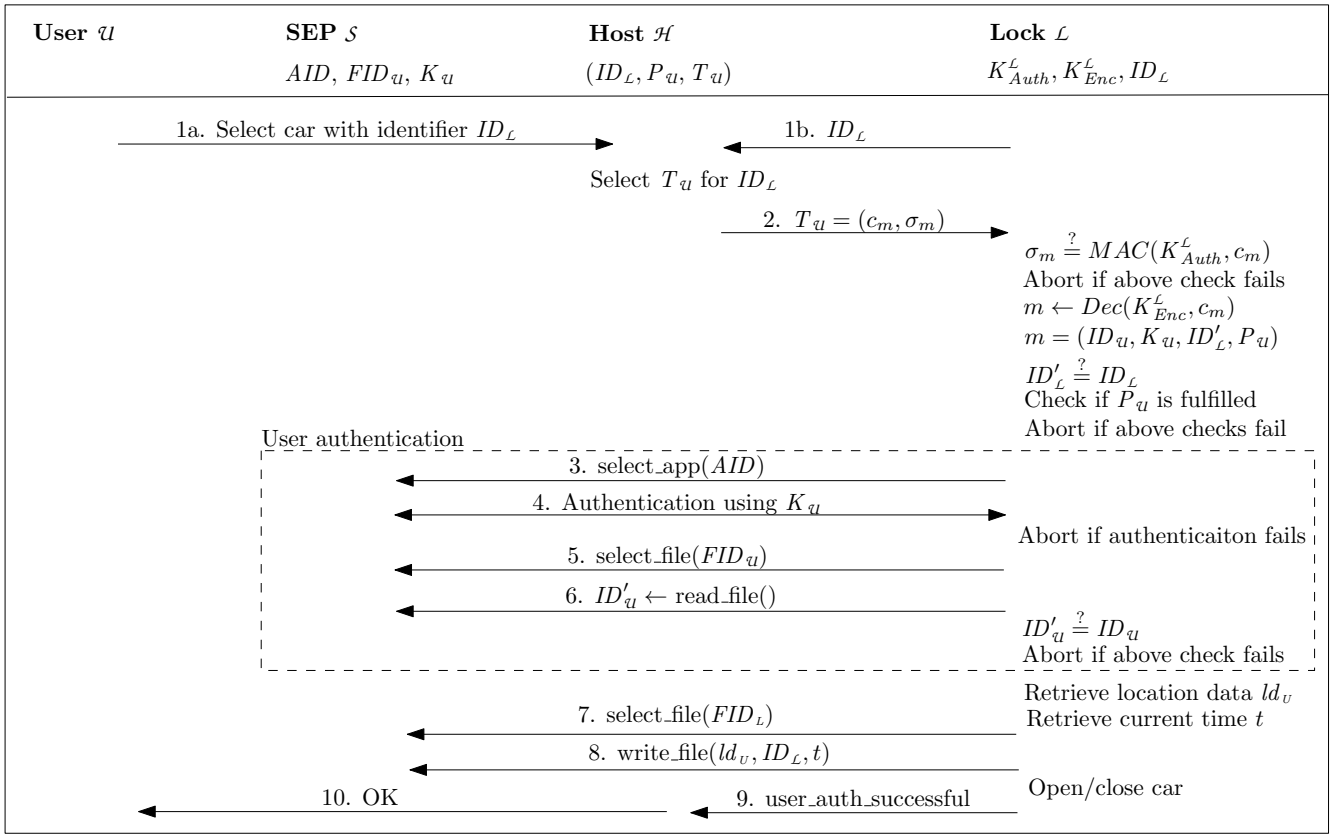

Figure 4 – User Authentication Protocol

Depending on the underlying communication technology used for the communication between $\mathcal{L}$ and $\mathcal{H}$, it may have an active or a passive character. In particular, if BLE is used, the protocol is initiated by $\mathcal{U}$ who presses the respective button in the application, while the host infers identifier $ID_{\mathcal{L}}$ of the car to be booked based on $\mathcal{U}$'s input (step 1a). In case of NFC, however, the authentication is initiated by $\mathcal{L}$ which sends its identifier $ID_{\mathcal{L}}$ to the host as soon as the user taps car's NFC reader with the smartphone (step 1b).

Starting from step 2, protocols for NFC and BLE versions are identical: $\mathcal{H}$ fetches the corresponding access token $T_{\mathcal{U}} = (c_m, \sigma_m)$ from its memory and sends it to $\mathcal{L}$ (step 2). Upon receive, the token is verified by $\mathcal{L}$ as follows: First, a MAC is calculated over c_m using the key $K^{\mathcal{L}}_{Auth}$ and the result is compared with σ_m from the token. If verification succeeds, the cipher c_m is decrypted using $K^{\mathcal{L}}_{Enc}$ key. Furthermore, the lock checks if its identifier $ID_{\mathcal{L}}$ matches with the identifier $ID'_{\mathcal{L}}$ contained within the token and if the policy conditions $P_{\mathcal{U}}$ are satisfied. If all the checks pass, the lock initiates authentication with $\mathcal{U}$'s SEP $\mathcal{S}$ to ensure the possession of $K_{\mathcal{U}}$. In particular, $\mathcal{L}$ sends `select_app` request to $\mathcal{S}$ in order to select the car sharing application AID (step 3), then runs authentication protocol with the key $K_{\mathcal{U}}$ (step 4), selects file $FID_{\mathcal{U}}$ and reads user identifier $ID'_{\mathcal{U}}$ from it (steps 5-6). $ID'_{\mathcal{U}}$ read from the file is then compared with $ID_{\mathcal{U}}$ from the token, and if identical, the authentication is successful. Thereby $\mathcal{L}$ retrieves location information ld_L from $\mathcal{L}$'s GPS and writes it along with $ID_{\mathcal{L}}$ and the current time t into the file FID_L of $\mathcal{S}$ (steps 7-8). Finally, the car opens (resp. closes) and the authentication status is sent to $\mathcal{H}$ (step 9) and a feedback is provided to $\mathcal{U}$ (step 10).

3.1.5 Checkout

With the *checkout protocol* depicted in Figure 5 $\mathcal{U}$ finalizes sharing the car. Thereby, $\mathcal{C}$ retrieves the last location of the car from the user's platform, then updates the car list with the new location information and makes the respective car available again to other customers.

$\mathcal{U}$ initiates the checkout procedure by triggering $\mathcal{H}$ to send the user identifier $ID_{\mathcal{U}}$ and car identifier $ID_{\mathcal{L}}$ to $\mathcal{C}$ over the established TLS session (steps 1-3). In turn, $\mathcal{C}$ authenticates the user (step 4) by executing steps similar to steps 3-6 of the *user authentication* protocol. If successful, $\mathcal{C}$ selects the file FID_L on SEP (step 5) and retrieves location data ld_L of the car, its identifier $ID'_{\mathcal{L}}$ and the timestamp t via `read_file` operation (step 6). After ensuring that the car identifier retrieved from the file matches the one received from $\mathcal{H}$ at step 3, $\mathcal{C}$ stores location ld_L and time t for the respective car and deletes the token $T_{\mathcal{U}}$. $\mathcal{H}$ is then notified about the successful checkout (step 7), deletes also $T_{\mathcal{U}}$ and provides feedback to $\mathcal{U}$ (step 8).

Note that if the user has no online connection at the moment he stopped using the service, he can lock the car in offline mode by executing user authentiation protocol and run the checkout protocol later on whenever the online connection becomes available.

3.2 Design Choices and Deployment Options

Our platform security concept relies on two isolated environments on the client side, host $\mathcal{H}$ and SEP $\mathcal{S}$, which handle two authentication factors of the user in separation.

Design Choices To achieve greater interoperability across

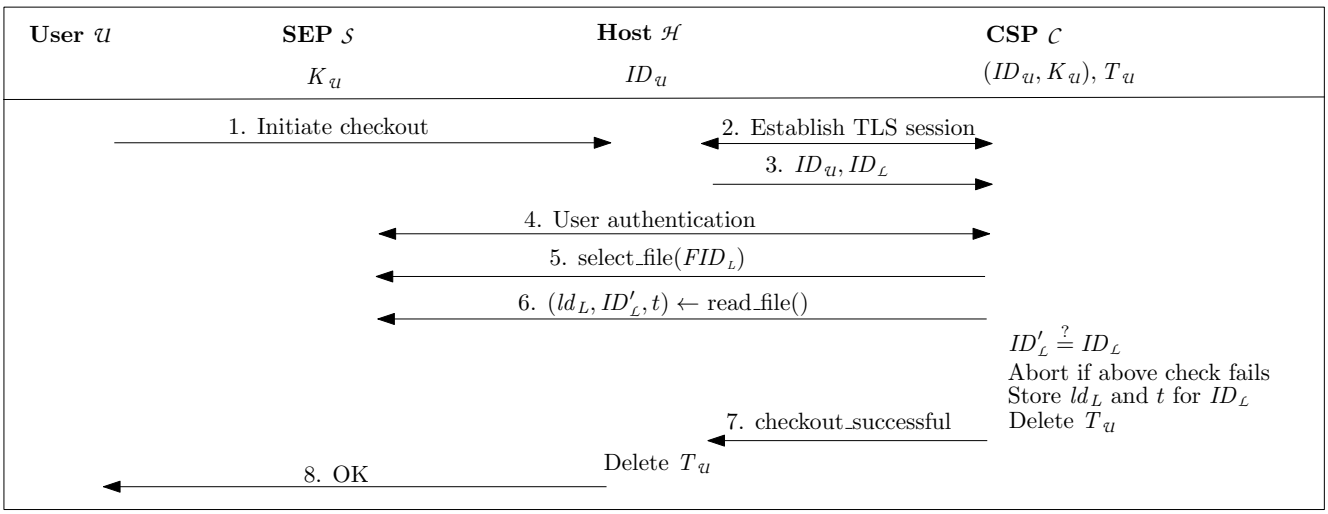

Figure 5 – Checkout Protocol

various mobile platforms, we opted to instantiate SEP $\mathcal{S}$ using the Java card environment, which is a well-standardized execution environment and therefore is widely supported on mobile platforms. This choice enables various deployment options ranging from embedded and removable secure elements (such as UICC-based and ASSD cards [15, 29]) to Java cards emulated in software. Furthermore, software-emulated Java cards can be deployed either on a mobile platform or even provided by external parties, e.g., by user-controlled wearable devices or cloud-based SE providers [30]. Moreover, Java cards can emulate cryptographic RFID cards (such as Mifare and Legic cards) by executing corresponding applets, which allows us to achieve compatibility to contactless RFID cards – the most commonly used authentication token in car sharing solutions, and to enable an additional deployment option based on a contactless smartcard.

Deployment Options Taken into account our security and functional requirements (cf. Section 2), we opt for the following three deployment options: (i) based on a Secure Element, (ii) a contactless smartcard, and (iii) emulated on user's smartwatch. All these options provide strong isolation between SEP $\mathcal{S}$ and host $\mathcal{H}$ and do not require online connection for their communication (and, hence, can achieve offline user authentication). These options can be supported by the car sharing provider simultaneously and be selected on per-user basis depending on capabilities of customer's hardware.

4. SECURITY ANALYSIS

In this section we provide informal security analysis of our solution by showing how the security requirements specified in Section 2.4 are fulfilled. The summary of our analysis is provided in Table 1 which indicates tolerance of various deployment options (cf. Section 3.2) to different adversary classes (cf. Section 2.3).

Well-established crypto. Our protocols rely only on well-established crypto primitives and algorithms. In particular, we use TLS in order to authenticate $\mathcal{C}$ to $\mathcal{H}$ and to protect their subsequent communication. TLS is widely used to secure client-server applications in the web and its security properties are widely studied and well understood. Further, communication with SEP $\mathcal{S}$ relies on standards ISO/IEC 7816-4/7816-8 developed for contactless identification cards. Moreover, software tokens generated by $\mathcal{C}$ are encrypted using Advanced Encryption Standard (AES), and authenticated using SHA-256 algorithm (cf. Section 5.2). These algorithms and standards represent the state of the art, and even if recognized as vulnerable in the future, they can easily be replaced with more secure versions.

Table 1 – Summary of the Security Analysis

Adversary Classes	*Class 1*	*Class 2*	*Class 3*
Secure Element	+	-	-
Contactless SmartCard	+	+	-
Smartwatch	+	+	+

Confidential credentials. Our system relies on the following credentials : (i) master key K_M and (ii) one-time password (OTP) used during user registration, (iii) user key $K_{\mathcal{U}}$ utilized to authenticate the user to CSP $\mathcal{C}$ during car booking, and to SEP $\mathcal{S}$ during checkout and authentication protocols[4], (iv) credentials $K^{\mathcal{L}}_{Enc}$ and $K^{\mathcal{L}}_{Auth}$ leveraged to encrypt and authenticate the token $T_{\mathcal{U}}$ during car booking and to decrypt and verify it during user authentication protocol.

Confidentiality of the master key K_M is ensured by keeping it secret from untrusted parties – it is only available to the SEP $\mathcal{S}$ and CSP $\mathcal{C}$. Similarly, confidentiality of keys $K^{\mathcal{L}}_{Enc}$ and $K^{\mathcal{L}}_{Auth}$ are ensured by making them available only to $\mathcal{C}$ and $\mathcal{L}$, which are trusted. The user key $K_{\mathcal{U}}$ is available to CSP $\mathcal{C}$ and to SEP $\mathcal{S}$ on client side – the end points which cannot be compromised in any of the adversary models. Further, $K_{\mathcal{U}}$ is always protected on transit, i.e., when transferred from $\mathcal{C}$ to SEP $\mathcal{S}$ during user registration, it is protected by the session key established after successful authentication of $\mathcal{C}$ with the master key K_M. Further, it is delivered from $\mathcal{C}$ to $\mathcal{L}$ in the encrypted and authenticated token. Whenever used for user authentication (in booking, checkout and authentication

[4]While our adversary model assumes that the car lock is trusted, one could consider approach to increase resilience to car lock compromise by using two distinct keys, e.g., $K^1_{\mathcal{U}}$ and $K^2_{\mathcal{U}}$, for authentication to CSP $\mathcal{C}$ and SEP $\mathcal{S}$, respectively.

protocols), it is utilized in a challenge-response manner and, unlike passwords, never leaves the platform.

The way to ensure confidentiality of $OTP_{\mathcal{U}}$ differs in different deployment options. For the deployment options 1 and 2, where SEP $\mathcal{S}$ is hosted either by the secure element or by the external contactless smartcard, the $OTP_{\mathcal{U}}$ is given to the host via the user interface (UI) provided by the operating system. Hence, this approach is secure against adversaries of classes 1 and 2, where the adversary is not allowed to compromise the host during user registration. For the third deployment option, where SEP SE is instantiated using user's smartwatch, $OTP_{\mathcal{U}}$ is provided to the (trusted) SEP $\mathcal{S}$ directly through the user interface available on the smartwatch instead of giving it to the host $\mathcal{H}$, which makes this deployment option secure against adversaries of classes 1-3.

To resist password-related attacks, such as social engineering and phishing, our solution relies on authentication keys unknown to users, with the only exception of OTP which is used during registration process. To protect the OTP from uncareful users, we represent it in the form of the QR-code so that it is never shown to users in clear text.

Isolation. The approach to establish isolation between SEP $\mathcal{S}$ and the host $\mathcal{H}$ varies depending on the deployment approach. Specifically, our deployment option 1 (based on the secure element) provides hardware-based isolation, while deployment options 2 and 3 (a smart card and a smartwatch) provide isolation via physical separation. All these options are secure against attacks of all adversary classes.

Strong credentials. Our solution relies on strong cryptographic secrets which are randomly chosen, uniformly distributed and have sufficient length. In particular, we use AES-128 for symmetric and RSA-2048 for asymmetric operations (cf. Section 5 for more details). For OTP used in the user registration, we use a randomly generated 128 bit string. The string is encoded in a QR code which is to be scanned with the smartphone's (smartwatch's) camera during user registration.

Authorized SEP invocation. Generally, SEP invocation authorization requires secure user interface (UI) to indicate user's intention to invoke protected credentials. Because secure elements do not feature their own UI, in our first deployment option (using secure element) SEP invocation is authorized using UI provided by the operating system. Hence, this deployment option can only tolerate class 1 adversary, which assumes that the attacker cannot compromise the mobile host $\mathcal{H}$ at any time. In the second deployment option using a contactless card, the user needs to put a smartcard in proximity to the smartphone's NFC reader in order to enable communication between the host $\mathcal{H}$ and the SEP $\mathcal{S}$. Hence, we use physical proximity of the card to the reader as indication that invocation is user-authorized. This approach is secure against adversary classes 1-3, as none of the adversaries can enforce proximity of the card to the NFC reader on behalf of the user.

In the third deployment option using a smartwatch, SEP invocation is confirmed using its own UI. Because such an authorization is provided directly to SEP, it is protected against adversaries of all classes.

To summarize, the first deployment option using secure element is secure against class 1 adversary, the second one based on contactless smartcard can tolerate 2nd adversarial class, while the third one utilizing a smartwatch can resist all three adversary classes.

5. PROTOTYPING AND EVALUATION

In this section, we briefly describe our car sharing prototype including hardware and software components and report performance evaluation results.

5.1 Prototype Hardware

For prototyping the mobile host we used Android smartphones Samsung S4 (GT-I9505) and Google Nexus 5 (LG D821) running Android 5.0 and 5.1 (Lollipop), respectively. For the first SE provider deployment option we equipped the S4 with an Giesecke & Devrient Mobile Security Card (MSC) [31] which acts as Secure Element and runs a DESFire applet. For the second deployment option using a contactless smartcard we utilized a Mifare DESFire EV1 smartcard. The third deployment alternative was instantiated using a Samsung Galaxy Gear SM-V700 smartwatch running Android 4.2 as an external SE provider.

The CSP $\mathcal{C}$ was hosted by a 64 bit Ubuntu Trusty Tahr (14.04.3 LTS) server running on a quadcore processor (Intel Xeon @ 3.5 GHz) with 16 GB of RAM.

To build a car lock prototype, we opted for a car key proxy approach which enabled compatibility with off-the-shelf cars. In particular, our car lock $\mathcal{L}$ is realized as telematics box with the car key fob inside[5], which offers a communication interface with the host $\mathcal{H}$. $\mathcal{L}$ is locked inside the car and communicates wirelessly with the smartphone. Once the authentication is successful, $\mathcal{L}$ activates the servo motor to push the buttons on the key fob, which results in car opening.

We built the telematics box using Lego Midstorms NXT 2.0, which holds inside the following hardware components: (*i*) an Arduino Mega Board [33] with either an BLE or NFC shield [34, 35] attached, (*ii*) a TowerPro MG995 servo [36], (*iii*) power supply in form of a 9-volt battery for the board and a 6 V-battery (4 AA batteries) for the servo and (*iv*) the respective car's key. We made the corresponding wiring diagram, step-by-step building instructions for the Lego frame and the photo of the resulting prototype available in [37].

We used two private cars, BMW 118i and BMW X4 xDrive30d, to perform evaluation of our prototype. Both cars feature Go button which can be used to start ignition without using the actual key. This feature requires the car key fob to be inside the vehicle – the requirement which is fulfilled in our case, as the car key is locked inside the telematics box placed inside the car.

5.2 Prototype Software

As summarized in Table 2, our prototype software consists of four modules and includes 21296 LOC in total excluding comments, blank lines and code from the external libraries.

Primitives and Parameter Sizes. The MAC scheme (for token authentication) is implemented based on SHA-256 with a digest size of 256 bits. For the symmetric encryption scheme AES in Cipher Block Chaining (CBC) mode with random padding is used with a key length of 128 Bit. Also the one-time password used in the user registration is a 128 bit

[5]Note that many deployed car sharing solutions already lock car keys inside the vehicle (typically in a safe [32] which can be unlocked with the smartcard). All the associated risks are covered by insurances.

Table 2 – Software Modules.

Module	*Language*	*Dependencies*	*LOC*
App	Android	JCAndroid [38], GMaps [39], ZXing [40]	5966
CSP	Java, PHP, CSS	BCC API [41], MySQL Connector [42], ZXing [40], MyEdit [43], JSCal [44]	7501
Lock	C/C++	AVR Crypto Lib [45], Servo Lib [46], NFC Lib [47], BLE Lib [48]	4212
DESFire Applet	Java	DESFire Applet Base [49]	3617
Total LOC			**21296**

BCC: Bouncy Castle Crypto, GMaps: Google Maps

string. To realize communication with the SEP we followed DesFire EV1 standard [50] using AES encryption and CMAC for message authentication.

Host $\mathcal{H}$. Depending on the deployment of SEP, the smartphone should run at least Android 4.3 (alias Jelly Bean) with API level 18 to support the BLE service [51] for the "external emulation" deployment option or at least Android 4.4 (alias KitKat) with API level 19 to support the Host APDU Service [52] necessary for the "contactless smartcard" deployment variant.

The car sharing app is implemented as an Android applicationwhereas the core functionality that drives the respective protocols is implemented as a Software Development Kit (SDK). For the car search we used the Google Maps API [39] as an external map provider to display both the user's and respective cars' positions on a map screen.

Furthermore, to decode the OTP from the QR code during user registration we used the third party library ZXing [40].

Secure Element Provider $\mathcal{S}$. In order to achieve a general compatibility with the commonly deployed DESFire EV1 standard throughout our deployment options, we enhanced a third party implementation of DESFire applet [49] to support crypto primitives like AES and CMAC. The DESFire EV1 applet is then either hosted by the G&D MSC which is inserted into the smartphone for the first deployment option or hosted by the Android-based Java card emulator JCAndroid [38] running on the smartwatch for the third deployment option. Communication with the MSC is done via SEEK for Android's MSC SmartcardService [53] which includes an MSC interface to access the G&D card over special reads and writes to the file system – the approach which requires neither root privileges nor changes to system software.

Since communication with $\mathcal{S}$ is compliant with the DESFire EV1 standard, we could easily adapt our system to support the second deployment option which uses a real DESFire EV1 card as SE provider.

In all deployment approaches we wrapped DESFire's native commands [50] in ISO/IEC 7816 Application Protocol Data Unit (APDU) commands and transmitted them to the DESFire application where they are then further processed.

Car Sharing Provider $\mathcal{C}$. The CSP server is implemented in Java and uses the *Bouncy Castle Crypto API (v1.46)* [41] for cryptographic operations, e.g., encryption, decryption or signing, as well as to establish the TLS communication with $\mathcal{H}$. Moreover, we used a Java-based *MySQL Connector (v5.1.18)* [42] to connect to the MySQL database (v5.5.46) running on the same platform and again the ZXing [40] library to encode the OTP to a QR code during user registration.

$\mathcal{C}$ further provides an administrative interface for the car sharing provider for displaying and managing the pool of cars and their status which uses the MyEdit [43] and JSCal [44] libraries to conveniently edit MySQL tables and utilize calender functionality.

Lock $\mathcal{L}$. Car lock software is implemented as an Arduino C/C++ application which depends on the *AVR Crypto Library* [45] and the libraries [47, 48] for NFC and BLE shield, respectively. Additionally, the servo shield library [46] is used to control the servo. For the local communication between the car lock and client's host both, BLE and NFC are supported. While BLE is more convenient for the user since he or she can open the car over a distance, NFC is provided for backwards compatibility to solely smartcard-based systems.

5.3 Performance Evaluation

We measured the time required to complete user authentication towards the car in each deployment variant. Measurements were taken 100 times and average values as well as standard deviation are reported.

The overall measurement path is divided into five intervals, where (*i*) Host2SE interval covers communication between the car sharing app running on the Host $\mathcal{H}$ and the SEP $\mathcal{S}$, (*ii*) Host2Lock interval corresponds to the communication between $\mathcal{H}$ and the Lock $\mathcal{L}$, and (*iii*) Lock2Car interval includes the time needed by the servo motor to press the respective key button on the car key fob and, additionally the actual wireless communication between car key and respective car. We depict these intervals in Figure 6.

There are two more intervals which are omitted in the figure for brevity: (*iv*) the accumulated time of computations taken on each platform and (*v*) the accumulated discovery time necessary to establish a (BLE or NFC) connection between $\mathcal{H}$ and $\mathcal{L}$, and $\mathcal{H}$ and $\mathcal{S}$, respectively. Thereby, computations are already included into measurements (*i*) (resp. (*ii*)) and thus are not additionally added up into the overall authentication time. The results are summarized in Table 3.

Intervals *(iii)* and *(v)* unfortunately cannot be influenced by our implementation. While *(v)* is induced by BLE/NFC discovery performed during connection establishment between smartphone and Lock $\mathcal{L}$, *(iii)* results from the Lock2Car communication, which is due to our key proxy approach and includes the whole time of the key-to-car authentication. In total, intervals *(iii)* and *(v)* sum up into a fixed time ranging from 1.01 to 1.57 s, depending on the deployment option.

The time intervals which can be influenced by our implementation are *(i)*, *(ii)* and *(iv)*, which contribute 1.87

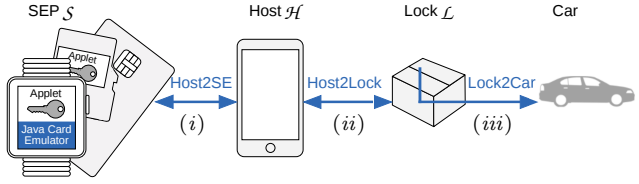

Figure 6 – Measurement Intervals.

to 3.82 s into a measurement path. This time can be improved through further optimizations and by other factors, e.g., through using newer hardware which could improve computation time.

To summarize, the overall authentication time ranges from 2.88 over 4.22 to 5.39 s depending on the deployment option.The longer intervals introduced by the ASSD and Smartwatch deployment option are induced almost entirely by the Host2SE communication and are caused by the limited capabilities of the respective hardware the SE is hosted on and/or the overhead introduced by the SEEK service. Besides hard- and software optimizations, the time required for user authentication can be further improved through integration of the car lock into the car and thus removing the interval (iii) from the measurement path.

Table 3 – User Authentication Performance
Values are given in seconds and show both average as well as standard deviation. Fixed time cannot be influenced by our implementation, while improvable time can be further optimized.

	ASSD Card	*Smartcard*	*Smartwatch*
(i) Host2SE	2,15 (±0,10)	0,13 (±0,01)	1,51 (±1,55)
(ii) Host2Lock	1,67 (±0,24)	1,74 (±0,08)	1,69 (±0,37)
(iii) Lock2Car	1,00	1,00	1,00
(iv) Computations	0,26 (±0,23)	0,13 (±0,04)	0,19 (±0,35)
(v) Discovery	0,57 (±0,01)	0,01 (±0,01)	0,02 (±0,02)
Fixed	1,57	1,01	1,02
Improvable	3,82	1,87	3,20
Total	5,39	2,88	4,22

6. RELATED WORK

The main body of the related work in the domain of car sharing is concentrated on such topics as process optimization, car relocation strategies and innovative competitive solutions, but typically does not cover security aspects of car sharing access control systems. For instance, Zhu et al. [54] propose a novel optimization approach to determine the depot location in station-based car sharing systems. Furthermore, in the context of the free-floating model, Formentin et al. [55] address the problem of future bookings by predicting the distance of the nearest available vehicle at a given future instant. Moreover, relocation strategies and algorithms in the free-floating business model are addressed in [56] and [57] that try to distribute cars in the business area according of the predicted booking demand. Finally, Shao et al. [58] present a dynamic car sharing system, that solves traffic congestion by reducing empty seats traveling. Thereby, the participants' smartphones continuously share driver's traveling information with a central server that analyzes and dynamically matches users with similar traveling needs.

The closest to our work is a car sharing solution [59] where the vehicle is equipped with an telematics box that is wired to the car's network and communicates on the one hand with the smartphone via short range wireless communication interfaces (NFC, Bluetooth) and on the other hand with a backend system via the Internet connection which opens the car. However, this solution is different from ours in that it requires Internet connection for cars and needs car modifications in order to attach the telematics box to the car's network.

In the industry sector, NFC-based immobilizer systems supporting car sharing were proposed independently by the automotive component suppliers Valeo [60] and Continental [61]. They rely on UICC-based SEs of local network operators for the protection of electronic car keys. In contrast, we aimed to avoid stakeholder-owned solutions, as those can only be used by customers of respective network operators. Furthermore, there are no details available in public domain on design and provided features, hence it is not possible to directly compare them with our solution regarding functionality and security. A virtual key service based on the Keyzee App [62] is powered by a Joint Venture of D'Ieteren and Continental, and is already in use by a car-sharing fleet in Monaco [63]. Similar to our telematics box, the Keyzee app works with an in the car integrated BLE box that connects with the smartphone in order to open or close the car [62]. However, in contrast to our solution this box intrusively interferes with the car's bus system [64] and thus invalidates the manufacturer's guarantees.

Related to our work are also smartphone-based access control solutions which were not initially developed for car sharing, but might be adapted to the new use case. With this respect, Arnosti et al. [65] introduced an access control solution which utilizes a microSD-based SE to store security-sensitive information on an NFC-enabled smartphone. However, it relies on an online connection to a central server and is only applicable to smartphones that have the corresponding hardware features (microSD slot). Further, Dmitrienko et al. [66] presented a generic access control system that enables the secure storage of access credentials for different resources on a smartphone. The system's generic approach makes it both applicable for digital and physical resources, like electronic resources or doors. If applied in a car sharing scenario, this solution would provide offline authentication, similar to ours. However, the solution relies on proprietary protocols which are not compatible to smartcards and does not provide various deployment options. Moreover, Abu-Saymeh et al. [67] proposed a framework that quantifies security of the mobile device based on user activities and behavior to secure NFC transactions. According to the identified security level, different authentication methods can be enforced by the system. In their work, authors took significantly different approach to address threats specific to mobile platforms and opted to detect suspicious activity rather than deploy a security architecture for protection of authentication secrets.

7. CONCLUSION

In this work, we propose a car sharing system for free-floating cars that overcomes shortcomings of current state-of-the-art solutions. It supports such new features as (*i*) offline authentication of users, (*ii*) compatibility to RFID cards – the commonly used authentication tokens in car sharing, and (*iii*) compatibility with legacy cars. Moreover, we present (*iv*) a mobile platform security concept for protection of electronic car keys on user's platforms which provides a flexibility to leverage various types of underlying hardware on client platforms and achieve best possible security given available hardware.

To summarize, our solution combines modern technologies in an intelligent way and, as a result, achieves new attractive features not available in state-of-the-art solutions which increase flexibility and security while preserving backward compatibility.

References

[1] Scott Le Vine, Alireza Zolfaghari, and John Polak. *Carsharing: Evolution, Challenges and Opportunities*. http://www.acea.be/publications/article/sag-report-22-carsharing-evolution-challenges-and-opportunities. 2014.

[2] Metro Vancouver. *The Metro Vancouver car share study – Technical report*. http://www.metrovancouver.org/services/regional-planning/PlanningPublications/MetroVancouverCarShareStudyTechnicalReport.pdf. 2014.

[3] Elliot Martin and Susan Shaheen. *The impact of carsharing on household vehicle ownership*. http://www.uctc.net/access/38/access38_carsharing_ownership.pdf. 2011.

[4] Susan Shaheen and Adam Cohen. *Innovative mobility carsharing outlook*. http://innovativemobility.org/wp-content/uploads/2016/02/Innovative-Mobility-Industry-Outlook_World-2016-Final.pdf. 2016.

[5] Shannon Bouton et al. *Urban mobility at a tipping point*. http://www.mckinsey.com/insights/sustainability/urban_mobility_at_a_tipping_point. 2015.

[6] bcs - Bundesverband CarSharing e.V. *Auf dem Weg zu einer neuen Mobilitätskultur - mehr als eine Million CarSharing-Nutzer*. http://carsharing.de/sites/default/files/uploads/ueber_den_bcs/pdf/bcs_jahresbericht_2014_final.pdf. 2014.

[7] car2go Deutschland GmbH. *How does it work?* https://www.car2go.com/en/berlin/how-does-car2go-work/. 2016.

[8] Which? The consumer interests company. *Mobile phone coverage map*. http://www.which.co.uk/reviews/mobile-phone-providers/article/mobile-phone-coverage-map.

[9] *Uber: Sign up to drive or tap and ride*. https://www.uber.com/.

[10] Alison Griswold. *Looks like Uber got hacked*. http://www.slate.com/blogs/moneybox/2015/02/27/uber_hack_50_000_drivers_may_be_affected_in_2014_security_breach.html. 2015.

[11] Joseph Cox. *Uber users say they're being charged for trips they didn't take*. http://motherboard.vice.com/read/uber-users-say-theyre-being-charged-for-trips-they-didnt-take. 2015.

[12] Joseph Cox. *Stolen Uber customer accounts are for sale on the dark web for $1*. Motherboard. 2015.

[13] Tiago Alves and Don Felton. "TrustZone: Integrated hardware and software security". In: *Information Quarterly* (2004).

[14] Jerome Azema and Gilles Fayad. *M-Shield Mobile security technology: Making wireless secure*. http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf. 2008.

[15] Certgate. *Certgate products. cgCard*. http://www.certgate.com/wp-content/uploads/2012/09/20131113_cgCard_Datasheet_EN.pdf. 2012.

[16] Jan-Erik Ekberg, Kari Kostiainen, and N Asokan. "The untapped potential of trusted execution environments on mobile devices". In: *IEEE Security & Privacy* (2014).

[17] Alexandra Dmitrienko et al. "Market-driven code provisioning to mobile secure hardware". In: *Financial Cryptography and Data Security*. 2015.

[18] Yvo Desmedt, Claude Goutier, and Samy Bengio. "Special uses and abuses of the Fiat-Shamir passport protocol". In: *Advances in Cryptology - CRYPTO. Annual International Cryptology Conference*. 1987.

[19] Nils Ole Tippenhauer and Srdjan Čapkun. "ID-based secure distance bounding and localization". In: *European Conference on Research in Computer Security*. 2009.

[20] Aanjhan Ranganathan, Boris Danev, and Srdjan Capkun. "Proximity verification for contactless access control and authentication systems". In: *Annual Computer Security Applications Conference*. 2015.

[21] G. P. Hancke. "Practical attacks on proximity identification systems". In: *IEEE Symposium on Security and Privacy*. 2006.

[22] Sebastiaan Indesteege et al. "A practical attack on KeeLoq". In: *Advances in Cryptology - EUROCRYPT. International Conference on the Theory and Applications of Cryptographic Techniques*. 2008.

[23] Markus Kasper et al. "Breaking KeeLoq in a flash: on extracting keys at lightning speed". In: *2nd International Conference on Cryptology in Africa (AFRICA-CRYPT'09)*. 2009.

[24] Aurélien Francillon, Boris Danev, and Srdjan Čapkun. "Relay attacks on passive keyless entry and start systems in modern cars". In: *Network and Distributed System Security Symposium (NDSS)*. 2011.

[25] Roel Verdult, Flavio Garcia, and Josep Balasch. "Gone in 360 seconds: Hijacking with Hitag2". In: *21st USENIX Security Symposium*. 2012.

[26] Virus News. *Teamwork: How the ZitMo Trojan bypasses online banking security*. http://www.kaspersky.com/about/news/virus/2011/Teamwork_How_the_ZitMo_Trojan_Bypasses_Online_Banking_Security. 2011.

[27] Cedric Ingrand. *French credit card hacker convicted*. http://www.theregister.co.uk/2000/02/26/french_credit_card_hacker_convicted/.

[28] M. Roland, J. Langer, and J. Scharinger. "Applying relay attacks to Google Wallet". In: *International Workshop on Near Field Communication (NFC)*. 2013.

[29] Press Release, Giesecke & Devrient. *G&D makes mobile terminal devices even more secure with new version of smart card in microSD format*. https://www.gi-de.com/en/about_g_d/press/press_releases/G&D-Makes-Mobile-Terminal-Devices-Secure-with-New-MicroSD-Card-g3592.jsp. 2010.

[30] Ramya Jayaram Masti, Claudio Marforio, and Srdjan Capkun. "An architecture for concurrent execution of secure environments in clouds". In: *ACM Cloud Computing Security Workshop*. 2013.

[31] Giesecke & Devrient Secure Flash Solutions. *The Mobile Security Card SE 1.0 offers increased security.* http://www.gd-sfs.com/the-mobile-security-card/mobile-security-card-se-1-0/.

[32] DB AG. *So einfach ist Flinkster.* https://www.flinkster.de/index.php?id=450&&f=3. 2016.

[33] Arduino MEGA 2560. *A microcontroller board based on the ATmega2560.* https://www.arduino.cc/en/Main/ArduinoBoardMega2560.

[34] BLE Shield. *A Bluetooth Low Energy extension for Arduino.* http://redbearlab.com/bleshield/.

[35] NFC Shield. *A NFC extension for Arduino.* http://www.seeedstudio.com/wiki/NFC_Shield_V2.0.

[36] ServoDatabase.com. *TowerPro MG995 servo.* http://www.servodatabase.com/servo/towerpro/mg995.

[37] crjp. *Car2X lock prototype.* https://github.com/crjp/car2x-lock-prototype. 2016.

[38] Technische Universität Darmstadt. *JCAndroid.* http://jcandroid.org/.

[39] Google Inc. *Google Maps Android API.* https://developers.google.com/maps/documentation/android-api/?hl=de.

[40] CZXing. *ZXing project.* https://github.com/zxing.

[41] Legion of the Bouncy Castle Inc. *The legion of the Bouncy Castle.* https://www.bouncycastle.org/index.html.

[42] Oracle Corporation. *MySQL connectors.* https://dev.mysql.com/downloads/connector/.

[43] Platon Group and Contributors. *Instant MySQL table editor and PHP code generator.* http://www.phpmyedit.org.

[44] Mihai Bazon. *DHTML calendar JSCal.* http://dynarch.com/mishoo/calendar.epl.

[45] Daniel Otte. *AVR-Crypto-Lib.* http://www.das-labor.org/wiki/AVR-Crypto-Lib/en.

[46] Michael Margolis. *Servo library for Arduino.* https://github.com/arduino/Arduino/tree/master/libraries/Servo.

[47] Adafruit Industries & Seeed Studio. *Adafruit-PN532.* https://github.com/adafruit/Adafruit-PN532/.

[48] RedBearLab. *RBL BLEShield.* https://github.com/RedBearLab/BLEShield/tree/master/Arduino/libraries/RBL_BLEShield.

[49] Jorge Prado Casanovas and Gauthier Vandamme. *Java Card Desfire emulation.* https://code.google.com/p/java-card-desfire-emulation/.

[50] NXP Semiconductors. *NXP.* Tech. rep. 2015.

[51] Google Inc. *BluetoothGattService.* http://developer.android.com/reference/android/bluetooth/BluetoothGattService.html.

[52] Google Inc. *Host APDU service.* http://developer.android.com/reference/android/nfc/cardemulation/HostApduService.html.

[53] Giesecke & Devrient GmbH. *Secure Element Evaluation Kit for the Android platform.* http://seek-for-android.github.io/.

[54] X. Zhu et al. "Optimization approach to depot location in car sharing systems with big data". In: *IEEE International Congress on Big Data*. 2015.

[55] S. Formentin, A.G. Bianchessi, and S.M. Savaresi. "On the prediction of future vehicle locations in free-floating car sharing systems". In: *Intelligent Vehicles Symposium (IV), 2015 IEEE*. 2015.

[56] Patrick Briest and Christoph Raupach. "The car sharing problem". In: *Annual ACM Symposium on Parallelism in Algorithms and Architectures*. 2011.

[57] S. Weikl and K. Bogenberger. "Relocation strategies and algorithms for free-floating car sharing systems". In: *2012 15th International IEEE Conference on Intelligent Transportation Systems*. 2012.

[58] Jianhua Shao and Chris Greenhalgh. "DC2S: a dynamic car sharing system". In: *ACM SIGSPATIAL International Workshop on Location Based Social Networks*. 2010.

[59] G. Alli et al. "Green Move: Towards next generation sustainable smartphone-based vehicle sharing". In: *Sustainable Internet and ICT for Sustainability (SustainIT), 2012*. 2012.

[60] NFC World. *Orange and Valeo demonstrate NFC car key concept.* http://www.nfcworld.com/2010/10/07/34592/orange-and-valeo-demonstrate-nfc-car-key-concept/. 2010.

[61] Telecom. *Deutsche Telekom and automotive supplier Continental demonstrated car keys.* http://www.telekom.com/innovation/connectedcar/81840. 2011.

[62] Keyzee. *Carsharing - Keyzee in a few words.* http://sandbox.keyzee.eu/en/autopartage/. 2012.

[63] Continental. *Virtual key service for car-sharing companies: D'Ieteren and Continental form joint venture.* http://www.continental-corporation.com/www/pressportal_com_en/themes/press_releases/3_automotive_group/interior/press_releases/pr_2015_03_12_jointventure_en.html.

[64] OTA Keys. *OTA keys. Easy to use, easy to install!* https://www.youtube.com/watch?v=9Iv9VFyErb0. 2015.

[65] Christof Arnosti, Dominik Gruntz, and Marco Hauri. "Secure physical access with NFC-enabled smartphones". In: *International Conference on Advances in Mobile Computing and Multimedia*. 2015.

[66] Alexandra Dmitrienko et al. "SmartTokens: Delegable Access Control with NFC-Enabled Smartphones". In: *International Conference on Trust and Trustworthy Computing*. 2012.

[67] D. Abu-Saymeh, D. e. D. I. Abou-Tair, and A. Zmily. "An application security framework for Near Field Communication". In: *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. 2013.