

SePCAR: A Secure and Privacy-enhancing Protocol for Car Access Provision

(Extended Version)

Iraklis Symeonidis¹, Abdelrahaman Aly¹, Mustafa A. Mustafa¹, Bart Mennink², Siemen Dhooghe³, Bart Preneel¹

¹ imec-COSIC, KU Leuven, Belgium `first.last@esat.kuleuven.be`

² Radboud University, The Netherlands `b.mennink@cs.ru.nl`

³ KU Leuven, Belgium `siemen.dhooghe@student.kuleuven.be`

Abstract. We present an efficient secure and privacy-enhancing protocol for car access provision, named SePCAR. The protocol is fully decentralised and allows users to share their cars conveniently in such a way that the security and privacy of the users is not sacrificed. It provides generation, update, revocation, and distribution mechanisms for access tokens to shared cars, as well as procedures to solve disputes and to deal with law enforcement requests, for instance in the case of car incidents. We prove that SePCAR meets its appropriate security and privacy requirements and that it is efficient: our practical efficiency analysis through a proof-of-concept implementation shows that SePCAR takes only 1.55 seconds for a car access provision.

1 Introduction

As opposed to the traditional car ownership, the idea of car sharing, which allows users to share their cars conveniently, is gaining popularity. Statistics have shown that the worldwide number of users for car sharing services has grown from 2012 to 2014 by 170% (4.94 million) [1–4] with a tendency to increase by 2021 [5]. With the use of portable devices and in-vehicle telematics, physical car keys are slowly becoming obsolete. Keyless car Sharing Systems (KSSs) allow car owners to rather use their portable devices such as smartphones to distribute temporary digital car keys (access tokens) to other users and several car manufacturers (including Volvo [6], BMW [7] and Toyota [8]) have started investing in such systems. Moreover, unlike traditional car rental companies, KSSs can provide a relatively inexpensive alternative to users who need a car occasionally and on-demand [9]. The use of KSSs can also contribute to a decrease in the number of cars, effectively reducing CO₂ emissions [10] and the need for parking space [11].

In spite of these advantages, information collection in car sharing systems does not only jeopardise a system’s security, but also the users’ privacy. Uber used a tool called “Hell” to spy on their rival company drivers [12], whereas their mobile app *always* tracks their users’ locations [13]. In short, an adversary may try to eavesdrop and collect information exchanged within the KSS, tamper with the car sharing details, extract the key of a car stored in untrusted devices, generate a rogue access token to maliciously access a car or to deny having accessed a car. Regarding users’ privacy, an adversary may try to correlate and

link two car sharing requests for the same user or the car, to identify which user used which car and deduce the users' sharing preferences. These preferences can be established by collecting information about sharing patterns such as rental time, duration, pickup location, when, where and with whom someone is sharing a car. An adversary may even attempt to infer sensitive information about users such as racial and religious beliefs [14] or their health status, by identifying users who use cars for disabled passengers or visit hospitals regularly. Sensitive personal data are related to *fundamental rights and freedoms*, and merit protection regarding the collection and processing as it is articulated in the new EU General Data Protection Regulation (GDPR) [15]. A KSS may further introduce various other concerns with respect to connectivity issues [4], car key revocations in case users' device is stolen [16, 17], and the fact that malicious users may attempt to manipulate or even completely destroy potential forensic evidence on the car or their devices.

A naive way to mitigate the security and privacy concerns is to implement a peer-to-peer protocol between both the users and the car. The *car owner* can generate a temporary access token for her car using the car key and distribute it to the other user, the *consumer*, who can use the token to access the car. This approach has two main limitations: (i) the owner and the consumer may not trust each other, thus affecting the accountability of the system, and (ii) the owner has to have a copy of the car key on her personal device which is prone to get lost or stolen. These limitations can be overcome by having a centralised entity, which is trusted by both users, to perform the access token generation on behalf of the car owner. However, such a centralised entity will have to be fully trusted, which might not be realistic under real world scenarios. It can jeopardise the users' privacy as it will have access to users' booking details and car keys.

Our Contributions. We design a concrete and fully decentralised secure and privacy-enhancing protocol for car access provision, named SePCAR. The protocol provides generation and distribution of access tokens for car access provision, as well as update and revocation operations used for facilitating mutually agreed modifications of the booking details and protecting against misbehaving consumers, respectively. It internally uses secure multiparty computation to facilitate forensic evidence provision in case of car incidents or at the request of law enforcement. SePCAR is described in detail in Section 4.

We prove that the protocol fulfils the desired security and privacy requirements bound to the standards of connected cars. First, departing from Symeonidis et al. [18], we give a detailed list of security and privacy requirements in Section 2. Then, in Section 5, we prove that SePCAR meets its security and privacy requirements as long as its underlying cryptographic primitives (listed in Section 3) are secure.

Our theoretical complexity and practical efficiency analysis in Section 6 demonstrates SePCAR's competitiveness. In particular, we implemented a prototype as a proof-of-concept in C++ and we achieved a car access provision in ≈ 1.55 seconds.

Related Work. Enev et al. [19] showed that it is possible to reach high identification rates of drivers, from 87% to 99% accuracy, based on data collected by the

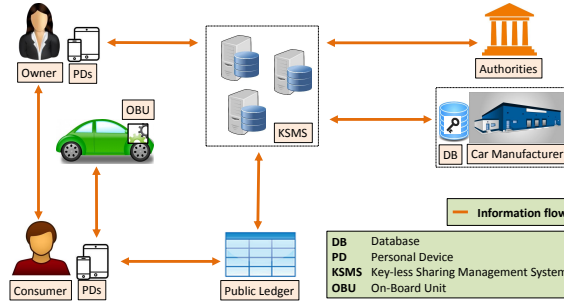


Fig. 1. System model of a physical Keyless car Sharing System (KSS) [18].

sensors of a car from 15 minutes of open-road driving. Troncoso et al. [20] proposed a pay-as-you-drive scheme that enhances the location privacy of drivers by sending only aggregated data to insurance companies. Balasch et al. [21] proposed an electronic toll pricing protocol where a car’s on-board unit calculates locally the driver’s annual toll fee while disclosing a minimum amount of location information. To mitigate colluding (dishonest) users, Floriat et al. [22] presented a privacy-preserving spot checking protocol that allows observations in public spaces. Mustafa et al. [23] proposed an anonymous electric vehicle charging protocol with billing support. The EVITA [24] and PRESERVE [25, 26] are designated projects on the design and specifications of the secure architecture of on-board units of cars. Driven by the PRESERVE instantiation, Raya et al. [27] described the need for Vehicular Public Key Infrastructure (VPKI), and Khodaei et al. [28] proposed a generic pseudonymization approach aiming to preserve the unlinkability of messages exchanged between vehicles and VPKI servers. None of these solutions provides a full-fledged keyless car sharing system, though.

Dmitrienko and Plappert [4] designed a secure two-factor authentication protocol using mobile platforms and RFID tags, e.g., smart-cards. However, in contrast to our solution, their protocol assumes a fully trusted car sharing provider which has access to the master key of smart-cards and also collects and stores all the information exchanged between the car provider and their users for every car access provision.

2 System Model and Requirements

In this section, we describe the system model and functionalities of a KSS. Moreover, we specify the threat model, the security, privacy and functional requirements which it needs to satisfy and the assumptions we will use.

System Model. We follow the KSS system model of Symeonidis et al. [18]. See also Fig. 1. *Users* are individuals who are willing to share their cars, *owners* (u_o), and use cars which are available for sharing, *consumers* (u_c), with the use of *Portable Devices* (PDs) such as smartphones. An *On-Board Unit* (OBU) is an embedded or a standalone hardware/software component [29] which is part of the secure access management system of a *car*. It has a wireless interface such as Bluetooth, NFC or LTE. The *Car manufacturer* (CM) is responsible for

generating and embedding a digital key into each car. These keys are used for car sharing and are stored in the manufacturers' *Database* (DB). The *Keyless Sharing Management System* (KSMS) is a complex of multiparty computation (MPC) *servers* that assists owners with a car access token generation, distribution, update and revocation. Each *server* individually retrieves its share of the car key, K^{car} , and the servers jointly encrypt the booking details, M^B , to generate an access token, AT^{car} . The access token is published on a *Public Ledger* (PL), which serves as a public bulletin board that guarantees the integrity of the data [30]. The booking details are typically agreed upon by owner and consumer prior to the beginning of the protocol.

Threat Model. Within the KSS the KSMS, the CM, and the PL are considered honest-but-curious entities. They will perform the protocol honestly, but they are curious to extract private information about users. Owners are passive adversaries while consumers and outsiders may be malicious. The car's OBU is trusted and equipped with Hardware Security Module (HSM) [25, 31] that supports secure key storage and cryptographic operations such as symmetric and public key encryption, following the EVITA [24] and PRESERVE [25] specifications. Users' PDs are untrusted as they can get stolen, lost or broken.

Protocol Design Requirements. The keyless car sharing system should satisfy the following security, privacy, and functional requirements, which we denote by SR , PR and FR , respectively. Here, we recall that M^B refers to the booking details, AT^{car} the access token to the car and K^{car} the car key.

- $SR1$ - *Confidentiality of M^B* . No one but the shared car, u_o and u_c should have access to M^B .
- $SR2$ - *Authenticity of M^B* . The shared car should verify the origin and integrity of M^B from u_o .
- $SR3$ - *Confidentiality of AT^{car}* . No one but the shared car and u_c should have access to AT^{car} .
- $SR4$ - *Confidentiality of K^{car}* . No one but the shared car and the car manufacturer should have access to K^{car} .
- $SR5$ - *Backward and forward secrecy of AT^{car}* . Compromise of a key used to encrypt any AT^{car} should not compromise other tokens (future and past) published on the PL of any honest consumer.
- $SR6$ - *Non-repudiation of origin of AT^{car}* . The u_o should be able to prove to u_c and the shared car that AT^{car} delivered to the car was originated by u_o .
- $SR7$ - *Non-repudiation of delivery of AT^{car}* . The u_c should be able to prove to u_o that the agreed AT^{car} was delivered to the shared car.
- $PR1$ - *Unlinkability of u_c and the car*. No one but the shared car, u_o and u_c should be able to link two booking requests of the same u_c for the car.
- $PR2$ - *Anonymity of u_c and the car*. No one but the shared car, u_o and u_c should learn the identity of u_c and the car.
- $PR3$ - *Undetectability of AT^{car} operation*. No one but the shared car, u_o and u_c (if necessary) should be able to distinguish between AT^{car} generation, update and revocation.

- *PR4 - Forensic evidence provision.* The KSMS should be able to provide authorities with forensic evidence for an access provision to a car at law enforcement requests without violating the other users’ privacy.
- *FR1 - Offline authentication.* Access provision should be provided for locations where cars have limited (or no) network connection.

Assumptions. For SePCAR, we assume that before every evaluation, the booking details are agreed upon by owner and consumer, but that both keep these booking details confidential against external parties. SePCAR relies on PKI infrastructure [11, 25, 28], and we assume that each entity has her private/public key pair with their corresponding digital certificates distributed authentically. The communication channels are secure and authenticated among entities using SSL-TLS and NFC. For an OBU, it is reasonable to assume that the cost for any adversary to perform a physical attack while renting a car is higher than the adversary’s capabilities due to the presence of the HSM [25, 31]. The MPC servers are held by non-colluding organisations, i.e., organisations with conflicting interests such as authorities, owner unions and car manufacturers.

3 Cryptographic Building Blocks

3.1 Cryptographic Functionalities

SePCAR uses the following cryptographic building blocks. The suggested instantiations are the ones used in our proof-of-concept implementation.

- $\sigma \leftarrow \text{sign}(Sk, m)$ and $\text{true/false} \leftarrow \text{verify}(Pk, m, \sigma)$ are public key operations for signing and verification respectively. These can be implemented using RSA.
- $z \leftarrow \text{prf}(K, \text{counter})$ is a pseudorandom function (PRF) using as input a key and a counter. This function can be implemented using CTR mode with AES (as the message input is small).
- $c \leftarrow \text{enc}(Pk, m)$ and $m \leftarrow \text{dec}(Sk, c)$ are public key encryption and decryption functions. These can be implemented using RSA.
- $c \leftarrow \text{E}(K, m)$ and $m \leftarrow \text{D}(K, c)$ are symmetric key encryption and decryption functions. These can be implemented using CTR mode with AES.
- $v \leftarrow \text{mac}(K, m)$ is a symmetric key MAC function. This function can be implemented using CBC-MAC with AES.⁴
- $z \leftarrow \text{hash}(m)$ is a cryptographic hash function. This function can be implemented using SHA-2.

We will furthermore use the notation $z \leftarrow \text{query}(x, y)$ to denote the retrieval of the x th value from the y th database DB (to be defined in Section 4), and $z \leftarrow \text{query_an}(y)$ to denote the retrieval of the y th value from the public ledger PL through an anonymous communication channel such as Tor [33], aiming to anonymously retrieve a published record submitted using the $\text{publish}(y)$ function.

⁴ CBC-MAC is proven to be secure as long as it is *only* evaluated on equal-size messages (or on prefix-free messages) [32], which is the case for SePCAR. For variable length messages, one should resort to *encrypted* CBC-MAC.

3.2 Multiparty Computation

Ben-or et al. [34] (commonly referred to as BGW) and Chaum et al. [35] proved that it is possible to calculate any function with perfect security in the presence of active and passive adversaries under the information-theoretic model, as long as there is an honest majority: 1/2 for passive and 2/3 for active adversaries. The former can be achieved by assuming the use of private channels among the servers and the latter using Verifiable Secret Sharing (VSS).

Our protocol is *MPC-agnostic*, meaning that it does not depend on the solution that implements the MPC functionality, and example protocols that could be executed within our protocol are SPDZ [36], BDOZ [37] and MASCOT [38]. However, the three-party protocol for Boolean circuits that was recently introduced by Araki et al. [39] is fairly suited for our current needs, given its performance and threshold properties. Thus, this is the protocol we use in our simulation. It can perform non-linear operations with relatively high throughput and somewhat low latency (when tested on 10 Gbps connections). The scheme provides threshold security against semi-honest parties. Note that Furukawa et al. [40] further adapt the protocol to provide security against a malicious adversary.

On an incremental setup for KSMS. In essence, our protocol can support an incremental setup and deployment where an $(l > 2)$ -case of KSMS servers is trivial, e.g., using BGW [34]. The 2-party case setting could also be trivially achieved by using l -party MPC protocols such as SPDZ [36], however, the forensic properties of our setup would no longer be attainable.

Multiparty Computation Functionalities. SePCAR uses the following cryptographic functionalities for MPC:

- $[x] \leftarrow \text{share}(x)$ is used to secretly share an input. This function can be instantiated using Araki et al.’s sharing functionality.
- $x \leftarrow \text{open}([x])$ reconstructs the private input based on the secret shares.
- $[z] \leftarrow \text{XOR}([x], [y])$ outputs a secret shared bit, representing the XOR of secret shared inputs $[x]$ and $[y]$. Note that for both the arithmetic or Boolean circuits, such functionality could be implemented without requiring any communication cost.
- $[z] \leftarrow \text{AND}([x], [y])$ outputs a secret shared bit, representing the AND of two secret shared inputs $[x]$ and $[y]$. This function can be instantiated using Araki et al.’s AND operation.
- $[z] \leftarrow \text{eqz}([x], [y])$ outputs a secret shared bit, corresponding to an equality test of two secret shared inputs $[x]$ and $[y]$. This is equivalent to computing $[z] \leftarrow [x] \stackrel{?}{=} [y]$ where $z \in \{0, 1\}$.
- $[C] \leftarrow \text{E}([K], [M])$ secretly computes a symmetric encryption from a secret shared key $[K]$ and a secret shared message $[M]$. Several protocols have been proposed to achieve this, e.g., [41, 42]. We include a succinct review on how to implement AES below.
- $[V] \leftarrow \text{mac}([K], [M])$ secretly computes a MAC from a secret shared key $[K]$ and a secret shared message $[M]$.

Table 1. Notations.

Symbol	Description
$\text{KSMS}, S_i, \text{PL}, \text{CM}, u_o, u_c$	Set of KSMS servers, the i th server for $i \in \{1 \dots l\}$, Public Ledger, Car Manufacturer, owner, consumer
$ID^B, ID^{u_o}, ID^{u_c}, ID^{car}$	ID of booking, u_o, u_c , car
$CD^{u_c}/AC^{u_c}, L^{car}$	Set of conditions/access rights under which u_c is allowed to access a car, car's location
$DB^{\text{CM}} / DB^{S_i}$	Database that CM holds with $(ID^{u_o}, ID^{car u_o}, K^{car u_o})$ / that S_i holds with $(ID^{u_o}, [ID^{car u_o}], [K^{car u_o}])$ for all owners (u_o 's) and their registered cars
\bar{D}^{u_o}	Car records $(ID_x^{u_o}, [ID_y^{car u_o}], [K_y^{car u_o}])$ of the x th u_o for the y th car extracted (query) from DB^{S_i} , where $ \bar{D}^{u_o} = n$
\bar{D}^{car}	The matched (eqz output) y th car key $([0] \dots [0][1][0] \dots [0])$, where $ \bar{D}^{car} = n$
$Pk^x / Sk^x, Cert^{u_c}$	Public/private key pair of the KSS entity x , certificate of u_c
M^B	Booking details, i.e., $\{\text{hash}(Cert^{u_c}), ID^{car}, L^{car}, CD^{u_c}, AC^{u_c}, ID^B\}$
$\sigma^{u_o}, \sigma^{car}$	Signature (sign output) of M^B with Sk^{u_o} , and $\{M^B, TS_{Access}^{car}\}$ with Sk^{car}
$K^{car}, K^{u_c}, K_1^{u_c}/K_2^{u_c}$	Symmetric key of the car, u_c 's master key, u_c 's session keys generated by (prf output) K^{u_c} and <i>counter/counter</i> + 1
M^{u_c}, AT^{u_c}	Concatenation of M^B with σ^{u_o} , a secure access token as the encryption (E output) of M^{u_c} with K^{car}
C^{S_i}	Ciphertext (enc output) of session keys $\{[K_1^{u_c}], [K_2^{u_c}]\}$ with Pk^{S_i}
$[C^{u_c}]$	Ciphertext (E output) of $\{[AT^{u_c}], [ID^{car}]\}$ with $[K_1^{u_c}]$
$C^B, [C^B]$	Message digest (mac output) of M^B with $K_2^{u_c}$, and $[M^B]$ with $[K_2^{u_c}]$
$TS_i^{Pub}, TS_{Access}^{car}$	Time-stamp of u_c accessing the shared car, a record published (publish) on the PL submitted by S_i

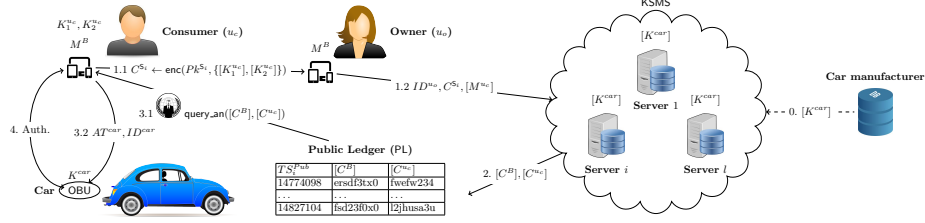


Fig. 2. SePCAR high level overview.

On the secure equality test. Various protocols have been proposed to implement the equality tests (previously referred eqz functionality). Common approaches provide either constant rounds or a logarithmic number of them on the bit size of its inputs, which could be proven more efficient for sufficiently small sizes. Furthermore, they also offer different security levels, i.e., perfect or statistical security. We refer the reader to the constructions presented in [43–45] for further details on their implementation and inner working. We assume the use of logarithmic depth comparison circuits.

On AES over MPC. AES has been the typical functionality used for benchmarking MPC protocols during the last few years. This and its usability on MPC based applications have motivated faster and leaner MPC implementations of the cipher. As it was previously stated, they consider the case where the computational parties hold a secret shared key K and a secret shared message M . The product of the operation is a secret shared AES encrypted ciphertext. We refer the reader to [42, 46–48] for further details and treatment on the state of the art. Note that in this paper we assume the use of Damgård and Keller [46] with some minor code optimisations.

4 SePCAR

In this section, we provide a detailed description of SePCAR. For simplicity and without loss of generality, we consider a single owner, consumer and a shared car. The description straightforwardly scales to a larger set of owners, consumers, and cars. Table 1 lists the notation used in the paper and Fig. 2 illustrates the high-level overview of SePCAR.

SePCAR consists of four steps: *session keys generation and data distribution*, *access token generation*, *access token distribution and verification* and *car access*. We will discuss these steps in detail in the remainder of the section, with a general overview picture given in Fig. 8 in Appendix A. Before discussing these steps, we first discuss a few *prerequisite* steps which have to be performed. After the discussion of the fourth (and last) step, we complete the section with an overview of the possible operations after SePCAR: *access token update and revocation*.

Prerequisite. Before SePCAR can commence, two prerequisite steps need to take place: *car key distribution* and setting the details for the *car booking*.

Car key distribution takes place immediately after the x th owner, $ID_x^{u_o}$, has registered her y th car, $ID_y^{car_{u_o}}$, with the KSMS. The KSMS forwards $ID_y^{car_{u_o}}$ to the Car Manufacturer (CM) to request the symmetric key, $K_y^{car_{u_o}}$, of the car. The CM retrieves $K_y^{car_{u_o}}$ from its DB, DB^{CM} and generates l secret shares of $K_y^{car_{u_o}}$ and $ID_y^{car_{u_o}}$, denoted by $[K_y^{car_{u_o}}]$ and $[ID_y^{car_{u_o}}]$, respectively. Then, it forwards each share to the corresponding KSMS server, i.e., S_i . Upon receipt of the shares, each S_i stores ID^{u_o} together with the shares, $[ID_y^{car_{u_o}}]$ and $[K_y^{car_{u_o}}]$, in its local DB, DB^{S_i} . The representations of the DB of CM and S_i are shown in Fig. 3. For simplicity, in some parts of SePCAR we will use ID^{u_o} , ID^{car} and K^{car} instead of $ID_x^{u_o}$, $ID_y^{car_{u_o}}$ and $K_y^{car_{u_o}}$.

$$DB^{CM} = \begin{pmatrix} ID_1^{u_o} & ID_1^{car_{u_o}} & K_1^{car_{u_o}} \\ \vdots & \vdots & \vdots \\ ID_x^{u_o} & ID_y^{car_{u_o}} & K_y^{car_{u_o}} \\ \vdots & \vdots & \vdots \\ ID_m^{u_o} & ID_n^{car_{u_o}} & K_n^{car_{u_o}} \end{pmatrix} \quad DB^{S_i} = \begin{pmatrix} ID_1^{u_o} & [ID_1^{car_{u_o}}] & [K_1^{car_{u_o}}] \\ \vdots & \vdots & \vdots \\ ID_x^{u_o} & [ID_y^{car_{u_o}}] & [K_y^{car_{u_o}}] \\ \vdots & \vdots & \vdots \\ ID_m^{u_o} & [ID_n^{car_{u_o}}] & [K_n^{car_{u_o}}] \end{pmatrix}$$

Fig. 3. The car manufacturer CM (left) and the DB of the i th server S_i (right).

Car booking allows u_o and u_c to agree on the booking details, i.e., $M^B = \{\text{hash}(Cert^{u_c}), ID^{car}, L^{car}, CD^{u_c}, AC^{u_c}, ID^B\}$, where $\text{hash}(Cert^{u_c})$ is the hash of the digital certificate of u_c , L^{car} is the pick-up location of the car, CD^{u_c} is the set of conditions under which u_c is allowed to use the car (e.g., restrictions on locations, time period), AC^{u_c} are the access control rights under which u_c is allowed to access the car and ID^B is the booking identifier. Recall that it is assumed that an owner and a consumer agree on the booking details beforehand.

Step 1: Session Keys Generation and Data Distribution. u_c generates two symmetric session keys, $K_1^{u_c}$ and $K_2^{u_c}$. Key $K_1^{u_c}$ will be used by each S_i

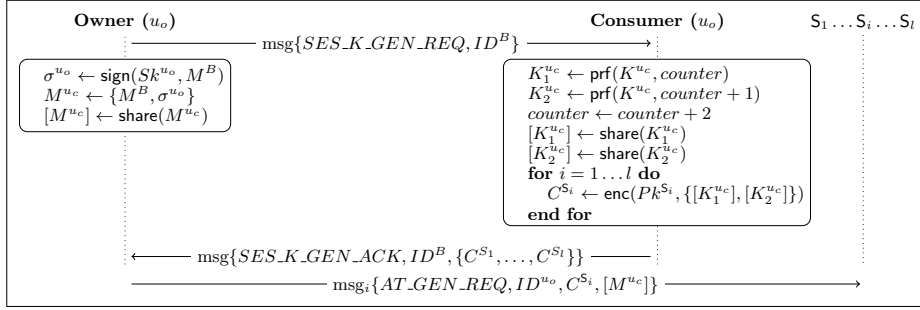


Fig. 4. Step 1: session keys generation and data distribution.

to encrypt the access token, such that only u_c has access to it. $K_2^{u_c}$ will be used to generate an authentication tag which will allow u_c to verify that the access token contains M^B which was agreed upon during the *car booking*. In addition, u_o sends the necessary data to each S_i , such that the access token can be generated. In detail, as it is shown in Fig. 4, u_o sends a session-keys-generation request, *SES_K_GEN_REQ*, along with ID^B to u_c . Upon receipt of the request, u_c generates $K_1^{u_c}$ and $K_2^{u_c}$ using the $\text{prf}()$ function instantiated by u_c 's master key, i.e., K^{u_c} and $counter$ and $counter+1$. Then, u_c transforms these into l secret shares, $[K_1^{u_c}]$ and $[K_2^{u_c}]$, one for each S_i in such a way that none of the servers will have access to the keys but that they can jointly evaluate functions using these keys securely. Then, it encrypts $[K_1^{u_c}]$ and $[K_2^{u_c}]$ with the public key of each S_i , $C^{S_i} = \text{enc}(Pk^{S_i}, \{[K_1^{u_c}], [K_2^{u_c}]\})$, such that only the corresponding S_i can access the corresponding shares. Finally, u_c forwards to u_o an acknowledgment message, *SES_K_GEN_ACK*, along with ID^B and $\{C^{S_1}, \dots, C^{S_l}\}$.

While waiting for the response of u_c , the owner u_o signs M^B with her private key, i.e., $\sigma^{u_o} = \text{sign}(Sk^{u_o}, M^B)$. In a later stage, the car will use σ^{u_o} to verify that M^B has been approved by u_o . Then u_o transforms $M^{u_c} = \{M^B, \sigma^{u_o}\}$ into l secret shares, i.e., $[M^{u_c}]$. Upon receipt of the response of u_c , u_o forwards to each S_i an access-token-generation request, *AT_GEN_REQ*, along with ID^{u_o} , the corresponding C^{S_i} and $[M^{u_c}]$.

Step 2: Access Token Generation. The servers generate an access token and publish it on the PL. In detail, as it is shown in Fig. 5, upon receipt of *AT_GEN_REQ* from u_o , each S_i uses the ID^{u_o} to extract $[K^{car}]$ from DB^{S_i} as follows. Initially, each S_i uses ID^{u_o} to retrieve the list of identities of all cars and car key shares related to the set of records that correspond to u_o . The result is stored in a vector \vec{D}^{u_o} of size $n \times 3$, i.e.,

$$\vec{D}^{u_o} = \begin{pmatrix} ID^{u_o} & [ID_1^{car u_o}] & [K_1^{car}] \\ \vdots & \vdots & \vdots \\ ID^{u_o} & [ID_y^{car u_o}] & [K_y^{car}] \\ \vdots & \vdots & \vdots \\ ID^{u_o} & [ID_n^{car u_o}] & [K_n^{car}] \end{pmatrix},$$

where n is the number of cars which u_o has registered with the KSS.

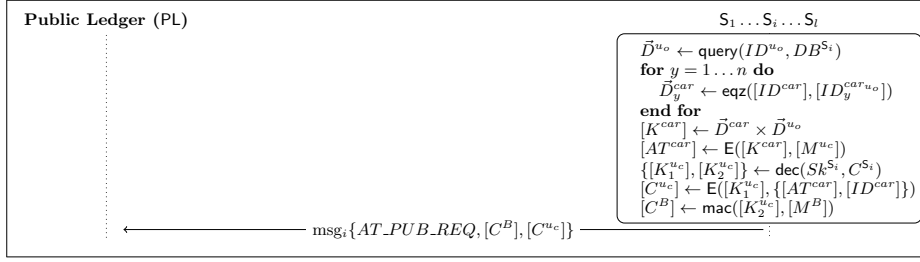


Fig. 5. Step 2: access token generation.

To retrieve the record for the car to be shared, each S_i extracts $[ID^{car}]$ from $[M^{u_c}]$ and performs a comparison with each of the n records of \vec{D}^{u_o} using the $\text{eqz}()$ function. The comparison outcomes 0 for mismatch and 1 for identifying the car at position y . The result of each iteration is stored in a vector \vec{D}^{car} with size $1 \times n$, i.e.,

$$\vec{D}^{car} = \begin{pmatrix} 1 \\ [0] \dots [0] [1] [0] \dots [0] \end{pmatrix} .$$

Each S_i then multiplies \vec{D}^{car} and \vec{D}^{u_o} to generate a third vector of size 1×3 , i.e.,

$$\vec{D}^{car} \times \vec{D}^{u_o} = \left(ID^{u_o} \ [ID_y^{car u_o}] \ [K_y^{car u_o}] \right) ,$$

from which the share of the car's secret key, $[K^{car}]$, can be retrieved. Then, the KSMS servers S_i collaboratively encrypt $[M^{u_c}]$ using the retrieved $[K^{car}]$ to generate an access token for the car in a shared form, $[AT^{car}]$.

As AT^{car} and ID^{car} need to be available only to u_c , a second layer of encryption is performed using $K_1^{u_c}$. To retrieve the shares of the session keys, $\{[K_1^{u_c}], [K_2^{u_c}]\}$, each S_i decrypts C^{S_i} using its private key. Then, the servers encrypt $[AT^{car}]$ and $[ID^{car}]$ with $[K_1^{u_c}]$ to generate $[C^{u_c}]$. In addition, they generate an authentication tag, $[C^B]$, using the $\text{mac}()$ function with $[K_2^{u_c}]$ and $[M^B]$ as inputs. Finally, each S_i sends to PL an access-token-publication request, AT_PUB_REQ , along with $[C^B]$ and $[C^{u_c}]$.

Step 3: Access Token Verification and Distribution. The PL publishes the shares of the encrypted access token which are then retrieved by u_c . Once retrieved, u_c can obtain the access token and use it to access the car. In detail, as it is shown in Fig. 6, upon receipt of AT_PUB_REQ , PL publishes $[C^B]$, $[C^{u_c}]$ and TS^{Pub} , which is the time-stamp of publishing the encrypted token. Then PL sends an acknowledgement of publication, AT_PUB_ACK , along with TS_i^{Pub} to at least one S_i which forwards it to u_o who, in turn, forwards it to u_c .

Upon receipt of AT_PUB_ACK , u_c uses TS_i^{Pub} to anonymously retrieve $[C^{u_c}]$ and $[C^B]$ from PL, such that PL cannot identify u_c . Then, u_c uses the $\text{open}()$ function to reconstruct C^B and C^{u_c} using the retrieved shares. Next, u_c verifies the authentication tag C^B locally using the $\text{mac}()$ function with $K_2^{u_c}$ and M^B as inputs. In case of successful verification, u_c is assured that the token contains the same details as the ones agreed during *car booking*. Then, u_c decrypts C^{u_c} using $K_1^{u_c}$ to obtain the access token and the car identity, $\{AT^{car}, ID^{car}\}$.

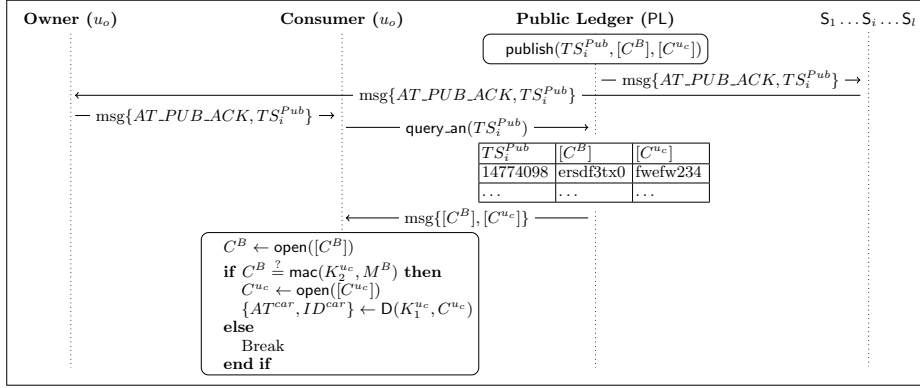


Fig. 6. Step 3: access token distribution and verification.

Step 4: Car Access. The consumer uses the access token to obtain access to the car. In detail, u_c sends $\{AT^{car}, ID^{car}, Cert^{u_c}\}$ to the car using a secure and close range communication channel such as NFC or Bluetooth (see Fig. 7). Upon receipt, the car's OBU obtains $M^{u_c} = \{M^B, \sigma^{u_o}\}$ by decrypting AT^{car} with K^{car} . It then performs three verifications. It checks if the access attempt satisfies the conditions specified in M^B . Then, it verifies σ^{u_o} to be assured that the booking details, M^B , have not been modified and have been indeed approved by the car owner. Finally, it verifies the identity of u_c . For the last verification, as the OBU receives $Cert^{u_c}$ (along with the $\text{hash}(Cert^{u_c})$ in M^B), it can use any challenge-response protocol based on public/private key [49] and RFIDs [4]. If any of these verifications fails, the OBU terminates the car access process and denies access to the car. Otherwise, it grants u_c access to the car, signs $\{M^B, TS_{Access}^{car}\}$, where TS_{Access}^{car} is the time-stamp of granting the access and asynchronously sends $\text{msg}\{\sigma_{Access}^{car}, TS_{Access}^{car}\}$ to u_o .

Access Token Update and Revocation. Upon an agreement between u_o and u_c to update or revoke an access token, SePCAR can be performed as described in steps 1-3. The values of an update request can be changed according to new booking details, \hat{M}^B , whereas for revocation, each of the parameters in \hat{M}^B can receive a predefined value indicating the revocation action. However, there are occasions when u_o may need to enforce an update or revocation of an access token. To prevent u_c from blocking such operations, SePCAR should be executed only by u_o , without the involvement of u_c . More detailed, u_o generates session

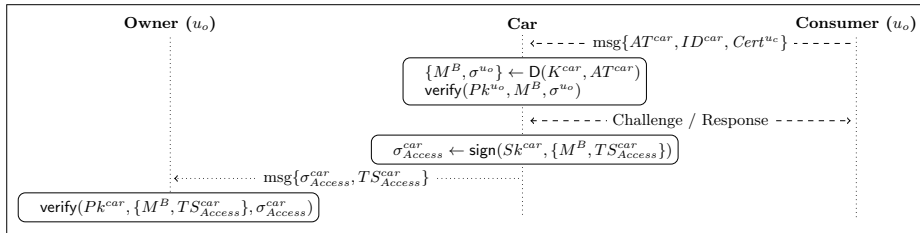


Fig. 7. Step 4: car access. Dashed lines represent close range communication.

keys, requests an access token, queries the PL, and sends the token to the car using long range asynchronous communication channel such as LTS.

5 Security and Privacy Analysis

We prove that SePCAR satisfies the security and privacy requirements of Section 2, provided that its underlying cryptographic primitives are sufficiently secure. Below theorem statement and proof are informal; a formal description of the security models and the stand-alone proof are given in Appendix B.

Theorem 1. *If communication takes place over private channels, the MPC is statistically secure,*

- the signature scheme `sign` is multi-key existentially unforgeable [50],
- the pseudorandom function `prf` is multi-key secure [51],
- the public key encryption scheme `enc` is multi-key semantically secure [52],
- the symmetric key encryption scheme `E` is multi-key chosen-plaintext secure [53],
- the MAC function `mac` is multi-key existentially unforgeable [50], and
- the hash function `hash` is collision resistant [54],

then SePCAR fulfils the security and privacy requirements of Section 2.

Note that, indeed, for each of the keyed cryptographic primitives we require security in the *multi-key* setting, as these are evaluated under different keys. For example, `sign` is used by all owners, each with a different key; `enc` is used for different keys, each for a different party in the KSMS, and `E` and `mac` are used for independent keys for every fresh evaluation of the protocol. We refer to Bellare et al. [52] for a discussion on generalizing semantic security of public key encryption to multi-key security; the adaptation straightforwardly generalizes to the other security models.

Proof (sketch). We treat the security and privacy requirements, and discuss how these are achieved from the cryptographic primitives, separately. We recall that consumer and owner have agreed upon the booking details prior to the evaluation of SePCAR, hence they know each other.

SR1 - Confidentiality of M^B . In one evaluation of the protocol, u_c , u_o , and the shared car learn the booking details by default or design. The KSMS servers only learn shares of the booking data, and under the assumption that the MPC is statistically secure, nothing about the booking data is revealed during the MPC. The outcomes of the MPC are C^B and C^{u_c} satisfying

$$C^B = \text{mac}(K_2^{u_c}, M^B), \tag{1}$$

$$C^{u_c} = \text{E}(K_1^{u_c}, \{\text{E}(K_y^{car u_o}, \{M^B, \sigma^{u_o}\}), ID^{car}\}), \tag{2}$$

both of which reveal nothing about M^B to a malicious outsider due to the assumed security of `mac`, `E`, and the independent uniform drawing of the keys $K_1^{u_c}$ and $K_2^{u_c}$. The nested encryption `E` does not influence the analysis due to the mutual independence of the keys $K_1^{u_c}$ and $K_y^{car u_o}$.

SR2 - Authenticity of M^B . An owner who initiates the access token generation and distribution, first signs the booking details using its private key before sending those to the KSMS in shares. Therefore, once the car receives the token and obtains the booking details, it can verify the owner's signature on the booking details. In other words, the car can verify the source of the booking details, the owner and their integrity. Suppose, to the contrary, that a malicious consumer can get access to a car of an owner u_o . This particularly means that it created a tuple (M^B, σ^{u_o}) such that $\text{verify}(Pk^{u_o}, M^B, \sigma^{u_o})$ holds. If σ^{u_o} is new, this means that u_c forges a signature for the secret signing key Sk^{u_o} . This is impossible by assumption that the signature scheme is existentially unforgeable. On the other hand, if (M^B, σ^{u_o}) is old but the evaluation is fresh, this means a collision $\text{hash}(Cert^{u_c}) = \text{hash}(Cert^{u_c'})$, which happens with negligible probability as hash is collision resistant.

SR3 - Confidentiality of AT^{car} . The access token is generated by the KSMS servers obliviously (as the MPC is statistically secure), and only revealed to the public in encrypted form, through C^{u_c} of (8). Due to the uniform drawing of the key $K_1^{u_c}$ (and the security of the public-key encryption scheme used to transmit this key), only the legitimate user can decrypt and learn the access token. It shares it with the car over a secure and private channel.

SR4 - Confidentiality of K^{car} . Only the car manufacturer and the car itself hold copies of the car key. The KSMS servers learn these in shared form, hence learn nothing about it by virtue of the statistical security of the MPC. Retrieving a car key from encryptions made under this key constitutes a key recovery attack, which in turn allows to break the chosen-plaintext security of the symmetric key encryption scheme.

SR5 - Backward and forward secrecy of AT^{car} . The access token is published on the public ledger as C^{u_c} of (8), encrypted under symmetric key $K_1^{u_c}$. Every honest consumer generates a fresh key $K_1^{u_c}$ for every new evaluation, using a pseudorandom function prf that is secure, i.e., that is indistinguishable from a random function. This implies that all session keys are drawn independent and uniformly at random. In addition, the symmetric encryption scheme \mathbf{E} is multi-key secure. Concluding, all encryptions C^{u_c} are independent and reveal nothing of each other. (Note that nothing can be said about access tokens for malicious users who may deviate from the protocol and reuse one-time keys.)

SR6 - Non-repudiation of origin of AT^{car} . The car, who is a trusted identity, verifies the origin through verification of the signature, $\text{verify}(Pk^{u_o}, M^B, \sigma^{u_o})$. The consumer u_c verifies the origin through the verification of the MAC function, $C^B \stackrel{?}{=} \text{mac}(K_2^{u_c}, M^B)$. Note that the consumer does not effectively verify AT^{car} , but rather C^B , which suffices under the assumption that the MPC servers evaluate their protocol correctly. In either case, security fails only if the asymmetric signature scheme or the MAC function are forgeable.

SR7 - Non-repudiation of delivery of AT^{car} . The owner can verify correct delivery through the verification of the message sent by the car to the owner, $\text{verify}(Pk^{car}, \{M^B, TS_{Access}^{car}\}, \sigma_{Access}^{car})$ at the end of the protocol. Security breaks only if the signature scheme is forgeable.

PR1 - Unlinkability of u_c and the car. The only consumer-identifiable data is in the consumer’s certificate included in the booking details. Note that these are agreed upon between the consumer and the owner, so the owner learns the identity of the consumer by default. Beyond that, the consumer only communicates with the car, which is supposed to learn the consumer’s identity so that it can perform proper access control. The consumer consults the public ledger over an anonymous channel. The booking details are transferred to and from the KSMS, but these are encrypted and do not leak by virtue of their confidentiality (security requirement SR1).

PR2 - Anonymity of u_c and the car. The reasoning is identical to that of PR1.

PR3 - Undetectability of AT^{car} operation. Access token generation, update, or revocation is performed using the same steps and the same type of messages sent to the KSMS and PL. Hence, outsiders and system entities can not distinguish which operation has been requested.

PR4 - Forensic evidence provision. In the case of disputes, the information related to a specific transaction (and only this information) may need to be reconstructed. This reconstruction can be done only if the KSMS servers collude and reveal their shares. In our setting, these servers have competing interests, thus they would not collude unless law authorities enforce them to do so. Due to the properties of threshold secret sharing, the private inputs can be reconstructed by a majority coalition. This is, if the KSMS consists of three parties, it suffices two of such parties to reconstruct the secrets (for semi-honest and malicious cases).

FR1 - Offline authentication. Note that steps 1-3 of the protocol require a network connection, but step 4, car access, is performed using close range communication and with no need of a network connection. The decryption and verification of the access token can be performed by the car offline (it has its key K^{car} and the owner’s public key Pk^{uo} stored). Sending the confirmation signature σ_{Access}^{car} can also be done offline. \square

6 Performance Evaluation

In this section, we provide a theoretical complexity and practical efficiency analysis of SePCAR.

Theoretical Complexity. The complexity of multiparty protocols is typically measured by the number of communication rounds produced by non-linear operations, as linear operations can usually be done without any information exchange and are virtually free of charge. We refer the reader to [55] for an extended review on complexity analysis on MPC. In one evaluation of SePCAR, the non-linear operations performed by the KSMS servers are (i) the retrieval of the car key by means of using multiple calls of the `eqz` functionality using the ID^{car} and their counterparts in \vec{D}^{car} as parameters, and (ii) two evaluations of the encryption scheme `E` and one evaluation of `mac`.

For (i) the evaluations of the `eqz` functionality, we consider a multiplicative depth of $\lceil \log(|ID^{car}|) \rceil + 1$, where $|ID^{car}|$ is the amount of bits in ID^{car} . Note that we can parallelize the `eqz` call for all \vec{D}^{car} entries. Therefore, the bulk of the overhead of extracting the car key comes from implementing the equality test in logarithmic depth [44]. Besides executing the `eqz` tests, we also have to perform an extra communication round since we need to multiply the result of each equality test with its corresponding car key. The total number of communication rounds for (i) is thus $\lceil \log(|ID^{car}|) \rceil + 1$.

For (ii) the two evaluations of the encryption scheme `E` and the single evaluation of `mac` we use, as mentioned in Section 3, CTR mode with AES and CBC-MAC with AES, respectively. Note that in a single AES evaluation the number of non-linear operations equals the number of S-Boxes evaluated in these functions, but many can be parallelized. Denote by ν the number of communication rounds needed to encrypt a single 128-bit block using AES. The two evaluations of CTR mode can be performed in parallel, and cost $2 \cdot \nu$ rounds. The evaluation of CBC-MAC is inherently sequential and costs $\left\lceil \frac{|M^B|}{128} \right\rceil \cdot \nu$ communication rounds.

The total amount of communication rounds can thus be expressed as

$$\left(\lceil \log(|ID^{car}|) \rceil + 1 \right) + 2 \cdot \nu + \left\lceil \frac{|M^B|}{128} \right\rceil \cdot \nu . \quad (3)$$

Practical Efficiency. Our protocol, as well as above computation, is agnostic towards the underlying multiparty protocol. In our experiments we have incorporated the 3-party semi-honest protocol by Araki et al. [39], given its relative efficiency of AES calls compared to alternatives such as Sharemind [56] and others [57–59]. The upshot of our experiments is that SePCAR needs only 1.55 seconds for a car access provision. We elaborate on our simulation below, following the steps of Section 4. A detailed segregation of the time into the different steps is given in Table 2.

Step 1. Recall that step 1 handles the preparation and sharing of the booking details and generation of keys. For `enc` we use RSA with 2048 bit keys ($\approx 2\text{ms}$) and for `sign` we use RSA with SHA-2 with a 512 bit output ($\approx 50\mu\text{s}$). The `prf` is implemented using AES in CTR mode ($\approx 2\mu\text{s}$). For all these functions we use OpenSSL [60]. The `share` function is implemented by the sharing primitive introduced by Araki et al. [39].

Step 2. In this step, the KSMS servers retrieve the car key and perform the corresponding encryption and other subroutines linked to generating the MAC. We consider the following message configuration size: `hash(Certuc)` of 512 bits, `IDcar` of 32 bits, `Lcar` of 64 bits, `CDuc` of 96 bits, `ACuc` of 8 bits, `IDB` of 32 bits and `σ^{uc}` of 512 bits. The booking details `MB` are of size 768 bits (including padding) and the final access token `ATuc` is of size 1408 bits (including padding). For the `dec` function we use RSA with 2048 bit keys ($\approx 2\text{ms}$). The symmetric encryption `E` is implemented in CTR mode and the `mac` in CBC mode. As mentioned before, the functions `E`, `mac`, and `eqz` use the basic primitives from

Table 2. Performance of SePCAR, where time is averaged over 1000 runs.

Phase	Description	Time (in sec)
Step 1	Sharing the booking details and keys	0.220 ± 0.027
Step 2	Extracting car key and making access token	1.274 ± 0.032
Step 3	Verifying the access token	$0.055 (+1 \text{ Tor } [62])$
Total		$1.551 \pm 0.043 (+1 \text{ Tor})$

Araki et al. [39], and we use the multiparty AES implementation of Damgård and Keller [46].

Step 2 also includes the MPC. Using Damgård and Keller [46], a single S-Box evaluation takes 5 communication rounds. A single evaluation of AES consists of 20 sequential evaluations of an S-Box, where we included the key expansion and took into account that parallelizable S-Boxes do not add up to the number of communication rounds, and can thus be encrypted in $\nu = 100$ communication rounds. From (3) we obtain that in our simulation the total number of communication rounds is

$$(5 + 1) + 2 \cdot 100 + 6 \cdot 100 = 806 .$$

We remark that key expansion for different keys only needs to be performed once, and for multiple evaluations of SePCAR for the same car the round complexity reduces.

Step 3. In this step the consumer retrieves, reconstructs, and verifies the assigned access token. The PL is implemented using SQLite. The implementation of `open` again follows the basic primitive given by Araki et al. [39], and `mac` is implemented using AES in CBC mode ($\approx 13\mu\text{s}$).

Step 4. The final step consists of a challenge-response protocol between u_c and the car, but it does not directly affect the performance of SePCAR and we omit it from our implementation.

Environment Settings. We implemented a realistic simulation for SePCAR in C++ and evaluated it using a machine equipped with an Intel *i7*, 2.6Ghz CPU and 8GB of RAM.⁵ The communication within the KSMS was simulated using socket calls and latency parameters. We used Araki et al. [39] to simulate the LAN latency ($\approx 0.13\text{ms}$) and Ramamurthy et al. [61] for Wi-Fi ($\approx 0.50\text{ms}$). We did not assume any specific network configuration for our experimentation.

7 Conclusion

SePCAR is proven to be secure and privacy-enhancing, efficiently performing in ≈ 1.55 seconds for a car access provision. We presented a formal analysis of the security and privacy requirements of our protocol and we provided a prototype as proof-of-concept. As future work, we plan to extend SePCAR to support additional operations such as booking and payment.

⁵ The implementation can be obtained from <https://bitbucket.org/Siemen11/sepcar>.

References

1. Shaheen, S., Cohen, A.: Innovative mobility carsharing outlook. University of Berkeley, California (2013)
2. Deloitte: Smart mobility: Reducing congestion and fostering faster, greener, and cheaper transportation options. <https://www.mysql.com/> Accessed April, 2017.
3. ACEA: Carsharing: Evolution, Challenges and Opportunities. <https://goo.gl/NTec4l> Accessed April, 2017.
4. Dmitrienko, A., Plappert, C.: Secure free-floating car sharing for offline cars. In: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, CODASPY 2017, Scottsdale, AZ, USA, March 22-24, 2017. 349–360
5. Bert, J., Collie, B., Gerrits, M., Xu, G.: What’s ahead for car sharing?: The new mobility and its impact on vehicle sales. (2016)
6. Volvo: Volvo Keyless Cars. <https://youtu.be/FF6JtS3y1xA> Accessed Nov., 2016.
7. BMW: DriveNow Car Sharing. <https://drive-now.com/> Accessed November, 2016.
8. USA TODAY: Toyota will test keyless car sharing. <http://tinyurl.com/hl8m6a7> Accessed November, 2016.
9. Wielinski, G., Trépanier, M., Morency, C.: Electric and hybrid car use in a free-floating carsharing system. *International Journal of Sustainable Transportation* **11**(3) (2017) 161–169
10. Shaheen, S.A., Cohen, A.P.: Car sharing and personal vehicle services: worldwide market developments and emerging trends. *Int. Journal of Sustainable Transportation* **7**(1) (2013) 5–34
11. Naphade, M.R., Banavar, G., Harrison, C., Paraszczak, J., Morris, R.: Smarter cities and their innovation challenges. *IEEE Computer* **44**(6) (2011) 32–39
12. The Guardian: Hell of a ride: even a PR powerhouse couldn’t get Uber on track. <https://goo.gl/UclihE> Accessed April, 2017.
13. Daring Fireball: Regarding Ubers New Always Location Tracking. <https://goo.gl/L1Elve> Accessed April, 2017.
14. reddit: Identifying Muslim cabbies from trip data and prayer times. <https://goo.gl/vLrW1s> Accessed April, 2017.
15. Council of the EU Final Compromised Resolution: General Data Protection Regulation. <http://www.europarl.europa.eu> Accessed Feb, 2015.
16. statista: Number of smartphone users in the United States from 2010 to 2021. <https://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us/> Accessed April, 2017.
17. GOV.UK: Reducing mobile phone theft and improving security. <https://www.gov.uk/government/publications/reducing-mobile-phone-theft-and-improving-security> Accessed April, 2017.
18. Symeonidis, I., Mustafa, M.A., Preneel, B.: Keyless car sharing system: A security and privacy analysis. In: IEEE International Smart Cities Conference, ISC2 2016, Trento, Italy, September 12-15, 2016. 1–7
19. Enev, M., Takakuwa, A., Koscher, K., Kohno, T.: Automobile driver fingerprinting. *PoPETs* **2016**(1) (2016) 34–50
20. Troncoso, C., Danezis, G., Kosta, E., Balasch, J., Preneel, B.: PriPAYD: Privacy-Friendly Pay-As-You-Drive Insurance. *IEEE Trans. Dependable Sec. Comput.* **8**(5) 742–755
21. Balasch, J., Rial, A., Troncoso, C., Preneel, B., Verbauwhede, I., Geuens, C.: PrETP: Privacy-Preserving Electronic Toll Pricing. In: 19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings. 63–78
22. Kerschbaum, F., Lim, H.W.: Privacy-preserving observation in public spaces. In: Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part II. (2015) 81–100

23. Mustafa, M.A., Zhang, N., Kalogridis, G., Fan, Z.: Roaming electric vehicle charging and billing: An anonymous multi-user protocol. In: IEEE SmartGridComm. (Nov 2014) 939–945
24. EVITA: E-safety Vehicle Intrusion Protected Applications (EVITA). <http://www.evita-project.org/> Accessed November, 2016.
25. PRESERVE: Preparing Secure Vehicle-to-X Communication Systems (PRESERVE). <https://www.preserve-project.eu/> Accessed November, 2016.
26. Stotz, J., Bißmeyer, N., Kargl, F., Dietzel, S., Papadimitratos, P., Schleiffer, C.: Security requirements of vehicle security architecture, preserve-deliverable 1.1 (2011)
27. Raya, M., Papadimitratos, P., Hubaux, J.: Securing vehicular communications. IEEE Wireless Commun. **13**(5) 8–15
28. Khodaei, M., Jin, H., Papadimitratos, P.: Towards deploying a scalable & robust vehicular identity and credential management infrastructure. CoRR **abs/1601.00846**
29. INVERS: Make Mobility Shareable. <https://invers.com/> Accessed April, 2017.
30. Micali, S.: Algorand: The efficient and democratic ledger. arXiv preprint arXiv:1607.01341 (2016)
31. Trusted Computing Group: TPM 2.0 Library Profile for Automotive-Thin. <http://tinyurl.com/jrklfj> Accessed June, 2016.
32. Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. **61**(3) 362–399
33. Tor Project: Protect your privacy. Defend yourself against network surveillance and traffic analysis. <https://www.torproject.org/> Accessed April, 2017.
34. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, ACM (1988) 1–10
35. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: STOC, ACM (1988) 11–19
36. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: CRYPTO '12. Volume 7417 of LNCS., Springer (2012) 643–662
37. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: EUROCRYPT '11. Volume 6632 of LNCS., Springer (2011) 169–188
38. Keller, M., Orsini, E., Scholl, P.: MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24–28, 2016. (2016) 830–842
39. Araki, T., Furukawa, J., Lindell, Y., Nof, A., Ohara, K.: High-throughput semi-honest secure three-party computation with an honest majority. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16
40. Furukawa, J., Lindell, Y., Nof, A., Weinstein, O.: High-throughput secure three-party computation for malicious adversaries and an honest majority. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II. 225–255
41. Albrecht, M., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. Cryptology ePrint Archive, Report 2016/687 (2016)
42. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer (2015) 430–454
43. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: TCC 2006. Volume 3876 of LNCS., Springer (2006) 285–304

44. Lipmaa, H., Toft, T.: Secure equality and greater-than tests with sublinear online complexity. In: ICALP (2). (2013) 645–656
45. Catrina, O., Hoogh, S.D.: Improved primitives for secure multiparty integer computation. In: SCN. (2010) 182–199
46. Damgård, I., Keller, M.: Secure multiparty AES. In: International Conference on Financial Cryptography and Data Security, Springer (2010) 367–374
47. Damgård, I., Keller, M., Larraia, E., Miles, C., Smart, N.: Implementing AES via an Actively/Covertly Secure Dishonest-Majority MPC Protocol. In: Security and Cryptography for Networks. Volume 7485 of LNCS., Springer (2012) 241–263
48. Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: MPC-Friendly Symmetric Key Primitives. Cryptology ePrint Archive, Report 2016/542 (2016)
49. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Cryptography **2**(2) 107–125
50. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2) 281–308
51. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4) (1986) 792–807
52. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. 259–274
53. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997. 394–403
54. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: FSE. Volume 3017 of LNCS., Springer (2004) 371–388
55. Cramer, R., Damgård, I., Nielsen, J.B.: Secure Multiparty Computation and Secret Sharing. Cambridge University Press (2015)
56. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A framework for fast privacy-preserving computations. In: Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings. (2008) 192–206
57. Talviste, R.: Applying secure multi-party computation in practice. Ph.D dissertation (2016)
58. Laur, S., Talviste, R., Willemson, J.: From oblivious AES to efficient and secure database join in the multiparty setting. In: Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings. (2013) 84–101
59. Launchbury, J., Diatchki, I.S., DuBuisson, T., Adams-Moran, A.: Efficient lookuptable protocol in secure multiparty computation. SIGPLAN Not. **47**(9) (September 2012) 189–200
60. OpenSSL: Cryptography and SSL/TLS Toolkit. <https://www.openssl.org/> Accessed April, 2017.
61. Ramamurthy, H., Prabhu, B., Gadh, R., Madni, A.M.: Wireless industrial monitoring and control using a smart sensor platform. IEEE sensors journal **7**(5) (2007) 611–618
62. Tor: TorMETRICS. <https://metrics.torproject.org/torperf.html> Accessed April, 2017.
63. Stinson, D.R.: Some observations on the theory of cryptographic hash functions. Des. Codes Cryptography **38**(2) (2006) 259–277
64. Rogaway, P.: Formalizing human ignorance. In: Progress in Cryptology - VIETCRYPT 2006, First International Conference on Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers. (2006) 211–228

A SePCAR Complete Representation.

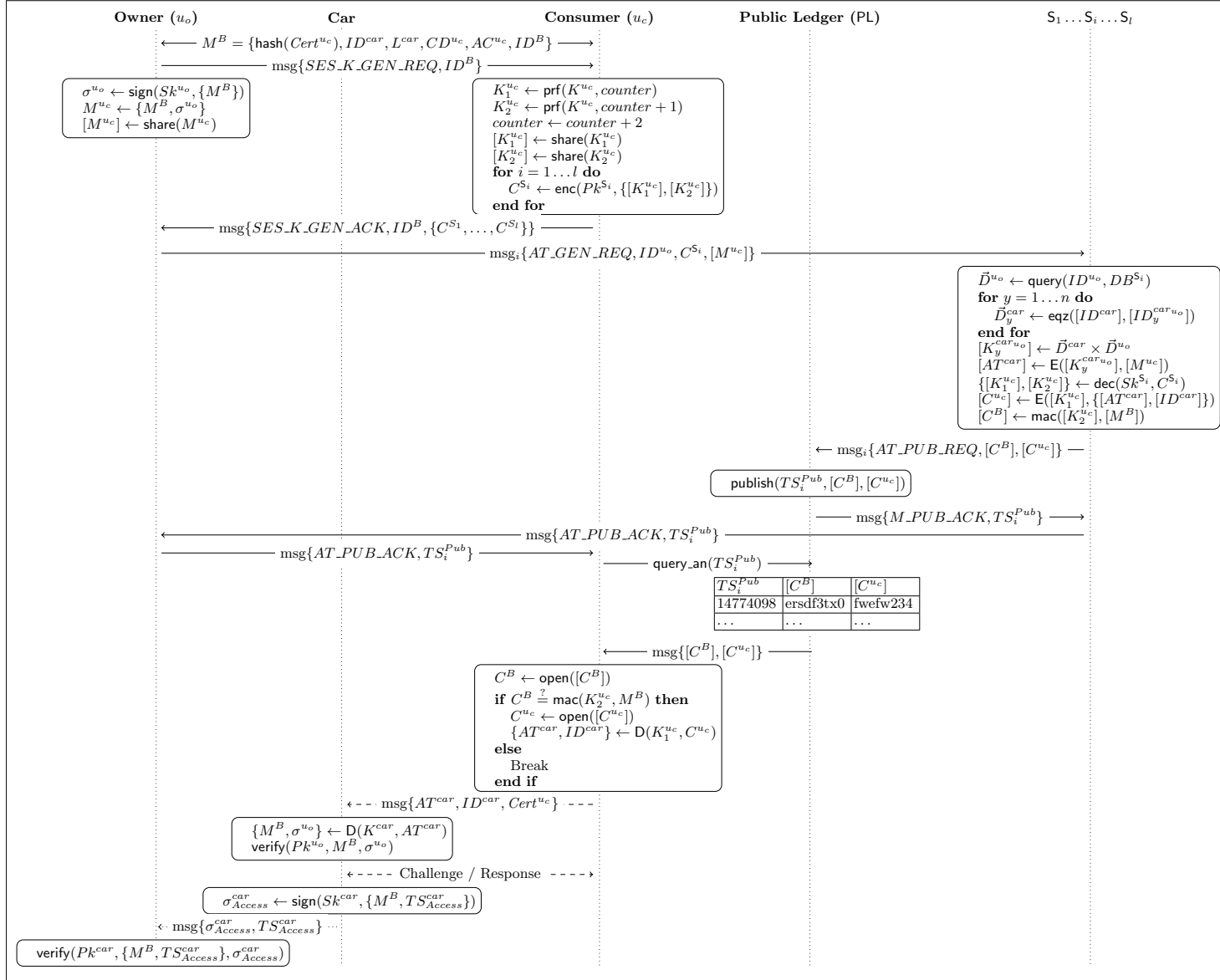


Fig. 8. SePCAR complete representation.

B Extended Security and Privacy Analysis

We prove that SePCAR satisfies the security and privacy requirements of Section 2, provided that its underlying cryptographic primitives are sufficiently secure. In Section B.1 we describe the security models of the cryptographic primitives. Then, the formal reasoning is given in Section B.2.

B.1 Cryptographic Primitives

The security definitions for signature schemes and MAC functions are inspired by Goldwasser et al. [50], for pseudorandom functions by Goldreich et al. [51], for public key encryption by Bellare et al. [52], and for symmetric key encryption by Bellare et al. [53].

We will, in fact, need security of the cryptographic primitives in the *multi-key* setting, as these are evaluated under different keys. For example, `sign` is used by all owners u_o , each with a different key; `enc` is used for different keys, each for a different party in the KSMS, and `E` and `mac` are used for independent keys for every fresh evaluation of the protocol. We refer to Bellare et al. [52] for a discussion on generalizing semantic security of public key encryption to multi-key security; the adaptation straightforwardly generalizes to the other security models.

In below definitions, for a function f , we define by $\text{Func}(f)$ the set of all functions with the exact same interface as f_K . We denote a random drawing by $\xleftarrow{\$}$.

Definition 1. Let $\mu \geq 1$. Consider a signature scheme $\text{sign} = (\text{keygen}, \text{sign}, \text{verify})$. For any adversary \mathcal{A} , we define its advantage in breaking the μ -multikey existential unforgeability as

$$\text{Adv}_{\text{sign}}^{\mu\text{-euf}}(\mathcal{A}) = \Pr \left((Pk^1, Sk^1), \dots, (Pk^\mu, Sk^\mu) \xleftarrow{\$} \text{keygen} : \mathcal{A}^{\text{sign}(Sk^i, \cdot)}(Pk^i) \text{ forges} \right),$$

where “forges” means that \mathcal{A} outputs a tuple (i, M, σ) such that $\text{verify}(Pk^i, M, \sigma) = 1$ and M has never been queried to the i -th signing oracle. We define by $\text{Adv}_{\text{sign}}^{\mu\text{-euf}}(q, t)$ the supremum over all adversaries making at most q queries and running in time at most t .

Definition 2. Let $\mu \geq 1$. Consider a pseudorandom function $\text{prf} = (\text{keygen}, \text{prf})$. For any adversary \mathcal{A} , we define its advantage in breaking the μ -multikey pseudorandom function security as

$$\text{Adv}_{\text{prf}}^{\mu\text{-prf}}(\mathcal{A}) = \left| \Pr \left(K^1, \dots, K^\mu \xleftarrow{\$} \text{keygen} : \mathcal{A}^{\text{prf}(K^i, \cdot)} = 1 \right) - \Pr \left(\$^1, \dots, \$^\mu \xleftarrow{\$} \text{Func}(\text{prf}) : \mathcal{A}^{\$^i} = 1 \right) \right|.$$

We define by $\text{Adv}_{\text{prf}}^{\mu\text{-prf}}(q, t)$ the supremum over all adversaries making at most q queries and running in time at most t .

Definition 3. Let $\mu \geq 1$. Consider a public-key encryption scheme $\text{enc} = (\text{keygen}, \text{enc}, \text{dec})$. For any adversary \mathcal{A} , we define its advantage in breaking the μ -multikey semantic security as

$$\text{Adv}_{\text{enc}}^{\mu\text{-pke}}(\mathcal{A}) = \left| \Pr \left((Pk^1, Sk^1), \dots, (Pk^\mu, Sk^\mu) \stackrel{\$}{\leftarrow} \text{keygen} : \mathcal{A}^{\mathcal{O}_0}(Pk^i) = 1 \right) - \Pr \left((Pk^1, Sk^1), \dots, (Pk^\mu, Sk^\mu) \stackrel{\$}{\leftarrow} \text{keygen} : \mathcal{A}^{\mathcal{O}_1}(Pk^i) = 1 \right) \right|,$$

where \mathcal{O}_b for $b \in \{0, 1\}$ gets as input a tuple (i, m_0, m_1) with $i \in \{1, \dots, \mu\}$ and $|m_0| = |m_1|$ and outputs $\text{enc}_{Pk^i}(m_b)$. We define by $\text{Adv}_{\text{enc}}^{\mu\text{-pke}}(t)$ the supremum over all adversaries running in time at most t .

Definition 4. Let $\mu \geq 1$. Consider a symmetric-key encryption scheme $\text{E} = (\text{keygen}, \text{E}, \text{D})$. For any adversary \mathcal{A} , we define its advantage in breaking the μ -multikey chosen-plaintext security as

$$\text{Adv}_{\text{E}}^{\mu\text{-ske}}(\mathcal{A}) = \left| \Pr \left(K^1, \dots, K^\mu \stackrel{\$}{\leftarrow} \text{keygen} : \mathcal{A}^{\text{E}(K^i, \cdot)} = 1 \right) - \Pr \left(\$^1, \dots, \$^\mu \stackrel{\$}{\leftarrow} \text{Func}(\text{E}) : \mathcal{A}^{\$^i} = 1 \right) \right|.$$

We define by $\text{Adv}_{\text{E}}^{\mu\text{-ske}}(q, t)$ the supremum over all adversaries making at most q queries and running in time at most t .

Definition 5. Let $\mu \geq 1$. Consider a MAC function $\text{mac} = (\text{keygen}, \text{mac})$. For any adversary \mathcal{A} , we define its advantage in breaking the μ -multikey existential unforgeability as

$$\text{Adv}_{\text{mac}}^{\mu\text{-mac}}(\mathcal{A}) = \Pr \left(K^1, \dots, K^\mu \stackrel{\$}{\leftarrow} \text{keygen} : \mathcal{A}^{\text{mac}(K^i, \cdot)} \text{ forges} \right),$$

where “forges” means that \mathcal{A} outputs a tuple (i, M, σ) such that $\text{mac}(K^i, M) = \sigma$ and M has never been queried to the i -th MAC function. We define by $\text{Adv}_{\text{mac}}^{\mu\text{-mac}}(q, t)$ the supremum over all adversaries making at most q queries and running in time at most t .

Finally, we consider the hash function hash to be collision-resistant. We denote the supremal probability of any adversary in finding a collision for hash in t time by $\text{Adv}_{\text{hash}}^{\text{col}}(t)$. The definition is, acknowledgeably, debatable: for any hash function there exists an adversary that can output a collision in constant time (namely one that has a collision hardwired in its code). We ignore this technicality for simplicity and refer to [54, 63, 64] for further discussion.

B.2 Analysis

We prove that SePCAR satisfies the security and privacy requirements of Section 2 provided that its underlying cryptographic primitives are sufficiently secure.

Theorem 2. *Suppose that communication takes place over private channels, the MPC is statistically secure, hash is a random oracle, and*

$$\begin{aligned} & \text{Adv}_{\text{sign}}^{\mu_o + \mu_{\text{car}} - \text{euf}}(2q, t) + \text{Adv}_{\text{prf}}^{\mu_c - \text{prf}}(2q, t) + \text{Adv}_{\text{enc}}^{l - \text{pke}}(t) + \\ & \text{Adv}_{\text{E}}^{q + \mu_{\text{car}} - \text{ske}}(2q, t) + \text{Adv}_{\text{mac}}^{q - \text{mac}}(q, t) + \text{Adv}_{\text{hash}}^{\text{col}}(t) \ll 1, \end{aligned}$$

where μ_o denotes the maximum number of u_o s, μ_c the maximum number of u_c s, μ_{car} the maximum number of cars, l the number of servers in the KSMS, q the total times the protocol gets evaluated, and t the maximum time of any adversary.

Then, SePCAR fulfills the security and privacy requirements of Section 2.

Proof. Recall from Section 2 that u_o s and CM are honest-but-curious whereas u_c s and outsiders may be malicious and actively deviate from the protocol. Cars are trusted.

Via a hybrid argument, we replace the pseudorandom functions $\text{prf}(K^{u_c}, \cdot)$ by independent random functions \mathcal{S}^{u_c} . This step is performed at the cost of

$$\text{Adv}_{\text{prf}}^{\mu_c - \text{prf}}(2q, t), \quad (4)$$

as in every of the q evaluations of SePCAR there are two evaluations of a function prf , and there are at most μ_c instances of these functions. As we assume that the MPC is performed statistically secure, we can replace the KSMS by a single trusted authority (with l interfaces) that is trusted, perfectly evaluates the protocol, and does not reveal/leak any information. Assuming that the public-key encryption reveals nothing, which can be done at the cost of

$$\text{Adv}_{\text{enc}}^{l - \text{pke}}(t), \quad (5)$$

we can for simplicity replace it with a perfectly secure public-key encryption ρ^{KSMS} to the KSMS directly (an encryption does not reveal its origin and content, and only KSMS can magically decrypt), therewith eliminating the fact that KSMS has l interfaces and has to perform multiparty computation. Now, as the pseudorandom functions are replaced by random functions, the keys to the symmetric encryption scheme E are all independently and uniformly distributed, and as the public-key encryption scheme is secure, these keys never leak. Therefore, we can replace the symmetric encryption functionalities by perfectly random invertible functions, $\pi^{\text{car}u_o}$ for the cars and unique π^{u_c} 's for every new encryption under u_c 's session keys, at the cost of

$$\text{Adv}_{\text{E}}^{q + \mu_{\text{car}} - \text{ske}}(2q, t), \quad (6)$$

as there are $q + \mu_{\text{car}}$ different instances involved and at most $2q$ evaluations are made in total. Note that this means that, instead of randomly drawing $K_1^{u_c} \leftarrow \mathcal{S}^{u_c}$, we now randomly draw $\pi^{u_c} \xleftarrow{\mathcal{S}} \text{Func}(\text{E})$.

We are left with a simplified version of SePCAR, namely one where the KSMS is replaced by a single trusted authority, the pseudorandom functions are replaced by independent random drawings (u_c uses \mathcal{S}^{u_c} which generates fresh outputs for every call), public-key encryptions are replaced with a perfectly secure public-key encryption function ρ^{KSMS} , and symmetric-key encryptions are

replaced by perfectly random invertible functions $\pi^{car_{u_o}}$ and π^{u_c} . The simplified protocol is given in Figure 9. Here, the derivation of the car key (or, formally, the random function corresponding to the encryption) from the database is abbreviated to $\pi^{car_{u_o}} \leftarrow \text{query}(ID^{u_o}, DB^{\text{KSMS}})$ for conciseness.

We will now treat the security and privacy requirements, and discuss how these are achieved from the cryptographic primitives, separately. We recall that u_c and u_o have agreed upon the booking details prior to the evaluation of SePCAR, hence they know each other by design.

SR1 - Confidentiality of M^B . In one evaluation of the protocol, u_c , u_o , the trusted KSMS, and the shared car learn the booking details by default or design. The booking details only become public through the values C^B and C^{u_c} satisfying

$$C^B = \text{mac}(K_2^{u_c}, M^B), \quad (7)$$

$$C^{u_c} = \pi^{u_c}(\{\pi^{car_{u_o}}(\{M^B, \sigma^{u_o}\}), ID^{car}\}). \quad (8)$$

The latter value reveals nothing about M^B as π^{u_c} is randomly generated for every evaluation, whereas the former value reveals nothing about M^B as $K_2^{u_c}$ is randomly generated for every evaluation. The nested encryption $\pi^{u_c} \circ \pi^{car_{u_o}}$ does not influence the analysis due to the mutual independence of the two functions.

SR2 - Authenticity of M^B . An owner who initiates the access token generation and distribution, first signs the booking details using its private key before sending those to the KSMS in shares. Therefore, once the car receives the token and obtains the booking details, it can verify u_o 's signature on the booking details. In other words, the car can verify the source of M^B , u_o , and its integrity. Suppose, to the contrary, that a malicious consumer can get access to a car of an u_o . This particularly means that it created a tuple (M^B, σ^{u_o}) such that $\text{verify}(Pk^{u_o}, M^B, \sigma^{u_o})$ holds. If σ^{u_o} is new, this means that u_c forges a signature for the secret signing key Sk^{u_o} . Denote the event that this happens by

$$E_1 : \mathcal{A} \text{ forges } \text{sign}(Sk^{u_o}, \cdot) \text{ for some } Sk^{u_o}. \quad (9)$$

On the other hand, if (M^B, σ^{u_o}) is old but the evaluation is fresh, this means a collision $\text{hash}(Cert^{u_c}) = \text{hash}(Cert^{u_c'})$. Denote the event that this happens by

$$E_2 : \mathcal{A} \text{ finds a collision for hash}. \quad (10)$$

We thus obtain that a violation of SR2 implies $E_1 \vee E_2$.

SR3 - Confidentiality of AT^{car} . The access token is generated by the KSMS obviously (as it is trusted), and only revealed to the public in encrypted form, through C^{u_c} of (8). Due to the uniform drawing of π^{u_c} (and the security of ρ^{KSMS} used to transmit this function), only the legitimate user can decrypt and learn the access token. It shares it with the car over a secure and private channel.

SR4 - Confidentiality of K^{car} . By virtue of our hybrid argument on the use of the symmetric-key encryption scheme, $E_{K^{car}}$ got replaced with $\pi^{car_{u_o}}$, which itself is a keyless random encryption scheme. As the key is now absent, it cannot leak.

SR5 - Backward and forward secrecy of AT^{car} . The access token is published on PL as C^{u_c} of (8), encrypted using π^{u_c} . Every honest u_c generates a uniformly randomly drawn function π^{u_c} for every new evaluation. Therefore, all encryptions C^{u_c} are independent and reveal nothing of each other. (Note that nothing can be said about access tokens for malicious users who may deviate from the protocol and reuse one-time keys.)

SR6 - Non-repudiation of origin of AT^{car} . The car, who is a trusted identity, verifies the origin through verification of the signature, $\text{verify}(Pk^{u_o}, M^B, \sigma^{u_o})$. The consumer u_c verifies the origin through the verification of the MAC function, $C^B \stackrel{?}{=} \text{mac}(K_2^{u_c}, M^B)$. Note that u_c does not effectively verify AT^{car} , but rather C^B . In either case, security fails only if the asymmetric signature scheme or the MAC function are forgeable. The former is already captured by event E_1 in (9). For the latter, denote the event that this happens by

$$E_3 : \mathcal{A} \text{ forges } \text{mac}(K_2^{u_c}, \cdot) \text{ for some } K_2^{u_c} . \quad (11)$$

We thus obtain that a violation of SR6 implies $E_1 \vee E_3$.

SR7 - Non-repudiation of delivery of AT^{car} . u_o can verify correct delivery through the verification of the message sent by the car to the him/her, $\text{verify}(Pk^{car}, \{M^B, TS_{Access}^{car}\}, \sigma_{Access}^{car})$ at the end of the protocol. Security breaks only if the signature scheme is forgeable. Denote the event that this happens by

$$E_4 : \mathcal{A} \text{ forges } \text{sign}(Sk^{car}, \cdot) \text{ for some } Sk^{car} . \quad (12)$$

We thus obtain that a violation of SR7 implies E_4 .

PR1 - Unlinkability of u_c and the car. The only consumer-identifiable data is in u_c 's certificate included in the booking details. Note that these are agreed upon between u_c and u_o , so u_o learns the identity of u_c by default. Beyond that, u_c only communicates with the car, which is supposed to learn u_c 's identity so that it can perform proper access control. u_c consults PL over an anonymous channel. The booking details are transferred to and from the KSMS, but these are encrypted and do not leak by virtue of their confidentiality (security requirement SR1).

PR2 - Anonymity of u_c and the car. The reasoning is identical to that of PR1.

PR3 - Undetectability of AT^{car} operation. Access token generation, update, or revocation is performed using the same steps and the same type of messages sent to the KSMS and PL. Hence, outsiders and system entities can not distinguish which operation has been requested.

PR4 - Forensic evidence provision. In the case of disputes, the information related to a specific transaction (and only this information) may need to be reconstructed. This reconstruction can be done only if the KSMS servers collude and reveal their shares. In our setting, these servers have competing interests, thus they would not collude unless law authorities enforce them to do so. Due to the properties of threshold secret sharing, the private inputs can be reconstructed by a majority coalition. This is, if the KSMS consists of three parties, it suffices two of such parties to reconstruct the secrets (for semi-honest and malicious cases).

FR1 - Offline authentication. Note that steps 1-3 of the protocol require a network connection, but step 4, car access, is performed using close range communication and with no need of a network connection. The decryption and verification of the access token can be performed by the car offline (it has its $\pi^{car_{u_o}}$ and u_o 's public key Pk^{u_o} stored). Sending the confirmation signature σ_{Access}^{car} can also be done offline.

Conclusion. In conclusion, SePCAR operates securely as long as the costs of (4-6), together with the probability that one of the events (9-12) occurs, are sufficiently small:

$$\text{Adv}_{\text{prf}}^{\mu_c\text{-prf}}(2q, t) + \text{Adv}_{\text{enc}}^{l\text{-pke}}(t) + \text{Adv}_{\text{E}}^{q+\mu_{\text{car}}\text{-ske}}(2q, t) + \Pr(\text{E}_1 \vee \text{E}_2 \vee \text{E}_3 \vee \text{E}_4) \ll 1.$$

By design, the probability that event $\text{E}_1 \vee \text{E}_4$ occurs is upper bounded by $\text{Adv}_{\text{sign}}^{\mu_o+\mu_{\text{car}}\text{-euf}}(2q, t)$, the probability that event E_3 occurs is upper bounded by $\text{Adv}_{\text{mac}}^{q\text{-mac}}(q, t)$, and the probability that E_2 occurs is upper bounded by $\text{Adv}_{\text{hash}}^{\text{col}}(t)$. We thus obtain

$$\Pr(\text{E}_1 \vee \text{E}_2 \vee \text{E}_3 \vee \text{E}_4) \leq \text{Adv}_{\text{sign}}^{\mu_o+\mu_{\text{car}}\text{-euf}}(2q, t) + \text{Adv}_{\text{mac}}^{q\text{-mac}}(q, t) + \text{Adv}_{\text{hash}}^{\text{col}}(t),$$

which completes the proof. \square

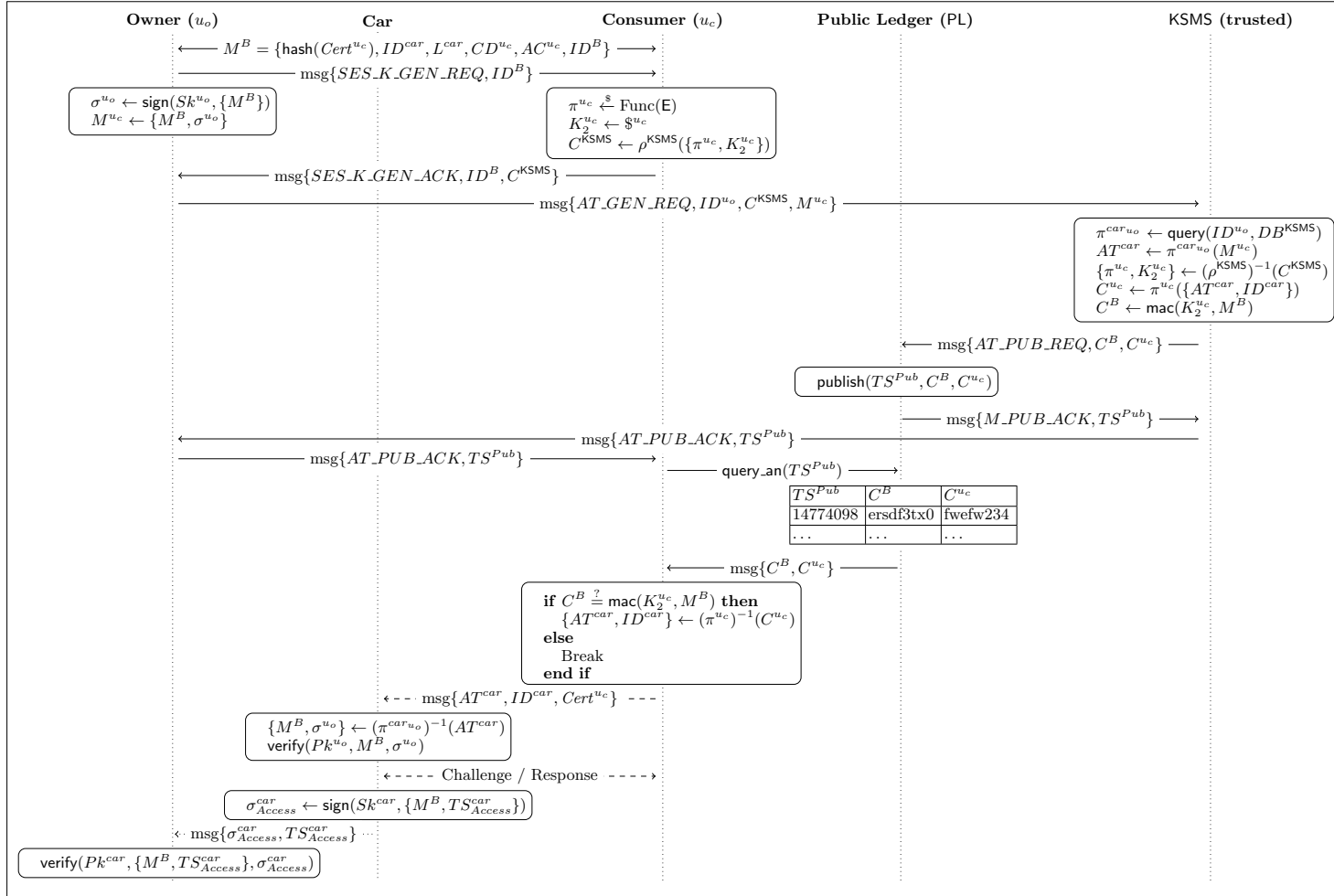


Fig. 9. Simplified representation of SePCAR for the proof of Theorem 2.