SePCAR: A Secure and Privacy-Enhancing Protocol for Car Access Provision
(Full Version)

Iraklis Symeonidis[1(✉)], Abdelrahaman Aly[1], Mustafa A. Mustafa[1], Bart Mennink[2], Siemen Dhooghe[3], Bart Preneel[1]

[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 without sacrifising their security and privacy. 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 in a convenient way, 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, 18, 20] with a tendency to increase by 2021 [9]. 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. Several companies (including Volvo [56], BMW [10], Toyota [55], and Apple [54]) 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 [57]. Their use can also contribute to a decrease in the number of cars, effectively reducing CO_2 emissions [46] and the need for parking space [38].

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 [49], whereas their mobile app *always* tracks their users' location [17]. Moreover, it is possible to

reach high identification rates of drivers, from 87% to 99% accuracy, based on data collected by the sensors of a car from 15 minutes of open-road driving [21]. 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 of the same user or the car, to identify car usage patterns 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 [43] or their health status, by identifying users who use cars for disabled passengers. Sensitive personal data are related to *fundamental rights and freedoms*, and merit protection regarding the collection and processing as articulated in the new EU General Data Protection Regulation (GDPR) [12]. In addition, a KSS may introduce various other concerns with respect to connectivity issues [20], car key revocations when a user's device is stolen [26], and the fact that malicious users may attempt to manipulate or even destroy potential forensic evidence on the car or their devices.

A 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.

Related Work. Troncoso et al. [52] proposed a pay-as-you-drive scheme to enhance the location privacy of drivers by sending aggregated data to insurance companies. Balasch et al. [4] 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. For colluding (dishonest) users [4], Kerschbaum et al. [33] presented a privacy-preserving spot checking protocol that allows observations in public spaces. Mustafa et al. [37] proposed an anonymous electric vehicle charging protocol with billing support. EVITA [22] and PRESERVE [40] are designated projects on the design and specification of the secure architecture of on-board units. Driven by the PRESERVE instantiation, Raya et al. [42] described the need for a Vehicular Public-Key Infrastructure (VPKI), and Khodaei et al. [34] proposed a generic pseudonymization approach

to preserve the unlinkability of messages exchanged between vehicles and VPKI servers. None of these solutions provides a full-fledged keyless car sharing system.

Our work is closely related to the protocol proposed by Dmitrienko and Plappert [20]. They designed a centralised and secure free-floating car sharing system that uses two-factor authentication including mobile devices and RFID tags, e.g., smart-cards. However, in contrast to our solution, their protocol assumes a fully trusted car sharing provider who 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.

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 the case of car incidents or at the request of law enforcement. SePCAR is described in detail in Sect. 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. [48], we give a detailed list of security and privacy requirements in Sect. 2. Then, in Sect. 5, we prove that SePCAR meets its security and privacy requirements as long as its underlying cryptographic primitives (listed in Sect. 3) are secure. Our theoretical complexity and practical efficiency analysis in Sect. 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.

2 System Model and Requirements

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 our assumptions about the system.

System Model. We follow the KSS system model of Symeonidis et al. [48] (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); both use of *Portable Devices* (PDs) such as smartphones. An *On-Board Unit* (OBU) is an embedded or a standalone hardware/software component [31] that 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

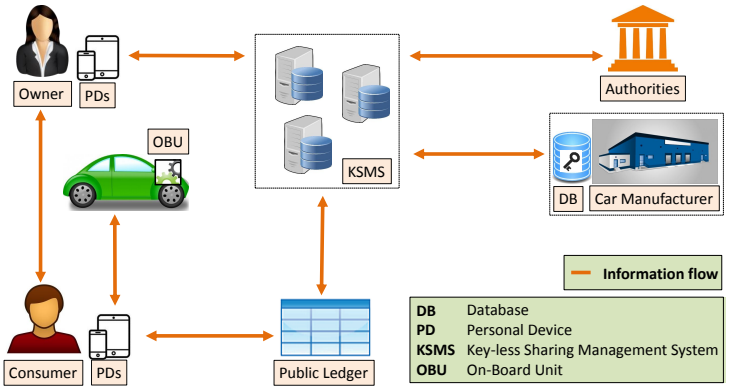

Fig. 1. System model of a physical Keyless car Sharing System (KSS) [48].

(MPC) *servers* that assists owners with 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 [36]. 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 a Hardware Security Module (HSM) [40, 53] that supports secure key storage and cryptographic operations such as symmetric and public-key encryption, following the EVITA [22] and PRESERVE [40] 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 [48], 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 CM 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 u_c.
- *SR6 - Non-repudiation of origin of AT^{car}.* The u_o should not be able to deny it has agreed to the terms of M^B, and participated in providing the respective AT^{car}.
- *SR7 - Non-repudiation of delivery of AT^{car}.* The u_c should not be able to deny it has obtained and used the AT^{car} to open the car (once it has done so).
- *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 the transaction details of an access provision to a car at the request of law enforcement 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 a PKI infrastructure [40], and we assume that each entity has her private/public-key pair with their corresponding digital certificates. The communication channels are secure and authenticated among entities using SSL-TLS and NFC. OBU is equipped with a HSM [40, 53], and it is designed to resist deliberate or accidental physical destruction (i.e., black box). The MPC servers are held by non-colluding organisations, i.e., organisations with conflicting interests such as authorities, car owner unions and car manufacturers.

3 Cryptographic Building Blocks

This section specifies, the cryptographic functionalities that are used across this paper, as well as the MPC functionalities and cryptographic building blocks.

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

- $\sigma \leftarrow \mathsf{sign}(Sk, m)$ and $\mathsf{true/false} \leftarrow \mathsf{verify}(Pk, m, \sigma)$ are public-key operations for signing and verification respectively. These can be implemented using RSA as defined in the PKCS #1 v2.0 specifications [29].

- $z \leftarrow \mathsf{prf}(K, counter)$ is a pseudo-random function (PRF) that uses as input a key and a counter. This function can be implemented using CTR mode with AES (as the message input is small).
- $c \leftarrow \mathsf{enc}(Pk, m)$ and $m \leftarrow \mathsf{dec}(Sk, c)$ are public-key encryption and decryption functions. These can be implemented using RSA as defined in the RSA-KEM specifications [30].
- $c \leftarrow \mathsf{E}(K, m)$ and $m \leftarrow \mathsf{D}(K, c)$ are symmetric key encryption and decryption functions. These can be implemented using CTR mode with AES.
- $v \leftarrow \mathsf{mac}(K, m)$ is a symmetric key MAC function. This function can be implemented using CBC-MAC with AES.[4]
- $z \leftarrow \mathsf{hash}(m)$ is a cryptographic hash function. This function can be implemented using SHA-2 or SHA-3.

We will furthermore use the notation $z \leftarrow \mathsf{query}(x, y)$ to denote the retrieval of the xth value from the yth database DB (to be defined in Sect. 4), and $z \leftarrow \mathsf{query\_an}(y)$ to denote the retrieval of the yth value from the PL through an anonymous communication channel such as Tor [51], aiming to anonymously retrieve a published record submitted using the $\mathsf{publish}(y)$ function.

Multiparty Computation. Ben-or et al. [8] (commonly referred to as BGW) 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.

Our protocol is *MPC-agnostic*, meaning that it does not depend on the solution that implements the MPC functionality; example protocols that could be executed within our protocol are SPDZ [16] or MASCOT [32]. However, the three-party protocol for Boolean circuits that was introduced by Araki et al. [3, 23] is fairly suited for our current needs, given its performance and threshold properties. Hence, we use this protocol 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 and malicious parties. Note that Furukawa et al. [23] further adapt the protocol to provide security against a malicious adversary.

On an incremental setup for KSMS. Our protocol can support an incremental setup and deployment where an ($l > 2$)-case of KSMS servers is trivial, e.g., using BGW [8]. The 2-party case setting could also be achieved with MPC protocols such as SPDZ [16], however, the forensic properties of our setup would no longer be attainable.

[4] CBC-MAC is proven to be secure as long as it is *only* evaluated on equal-size messages (or on prefix-free messages) [7], which is the case for SePCAR. For variable length messages, one should resort to *encrypted* CBC-MAC or replace the key for the last block [28].

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

- $[x] \leftarrow \mathsf{share}(\mathsf{x})$ is used to secretly share an input. This function can be instantiated using Araki et al.'s sharing functionality.
- $x \leftarrow \mathsf{open}([x])$ reconstructs the private input based on the secret shares.
- $[z] \leftarrow \mathsf{XOR}([x], [y])$ outputs a secret shared bit, representing the XOR of secret shared inputs $[x]$ and $[y]$. Note that for both arithmetic or Boolean circuits, such functionality could be implemented without requiring any communication cost.
- $[z] \leftarrow \mathsf{AND}([x], [y])$ 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.
- $[z] \leftarrow \mathsf{eqz}([x], [y])$ 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 $z \in \{0, 1\}$.
- $[C] \leftarrow \mathsf{E}([K], [M])$ secretly computes a symmetric encryption from a secret shared key $[K]$ and a secret shared message $[M]$. We include a succinct review on how to implement AES below.
- $[V] \leftarrow \mathsf{mac}([K], [M])$ secretly computes a MAC from a secret shared key $[K]$ and a secret shared message $[M]$.

On the secure equality test. Various protocols have been proposed to implement the equality tests (previously referred to an eqz functionality). Common approaches provide either constant rounds or a logarithmic number of them in 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 [11, 13, 35]. In this paper we assume the use of any logarithmic depth construction, which matches the current state of the art.

On AES over MPC. AES has been the typical functionality used for benchmarking MPC protocols during the last few years. This fact and its usability for MPC based applications have motivated faster and leaner MPC implementations of the cipher. As it was previously stated, they consider the case where the MPC 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 [2, 14, 15, 27]. Note that in this paper we assume the use of the methods proposed by Damgård and Keller [14] with some minor code optimisations.

4 SePCAR

This section provides 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.

Table 1. Notation.

Symbol	Description
KSMS, S_i, PL, CM, u_o, u_c	Set of KSMS servers, the ith 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^{CM} / $DB^{\mathsf{S}i}$	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
$\vec{D}^{u_o}$	Car records ($ID_x^{u_o}$, $[ID_y^{car_{u_o}}]$, $[K_y^{car_{u_o}}]$) of the xth u_o for the yth car extracted (query) from $DB^{\mathsf{S}i}$, where $\lvert\vec{D}^{u_o}\rvert = n$
$\vec{D}^{car}$	The matched (eqz output) yth car key ($[0]^1 \cdots [0][1]^y [0] \cdots [0]^n$), where $\lvert\vec{D}^{car}\rvert = n$
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, {$\mathsf{hash}(Cert^{u_c})$, ID^{car}, L^{car}, CD^{u_c}, AC^{u_c}, ID^B}
σ^{u_o}, σ^{car}_{Access}	Signature (sign output) of M^B with Sk^{u_o}, and $\{M^B, TS^{car}_{Access}\}$ 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 be (prf output) K^{u_c} and $counter/counter+1$
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 $Pk^{\mathsf{S}i}$
$[C^{u_c}]$	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^{car}_{Access}	Time-stamp of u_c accessing the shared car, a record published ($\mathsf{publish}$) on the PL submitted by S_i

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. 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 xth owner, $ID_x^{u_o}$, has registered her yth car, $ID_y^{car_{u_o}}$, with the KSMS. The KSMS forwards $ID_y^{car_{u_o}}$ to the 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,

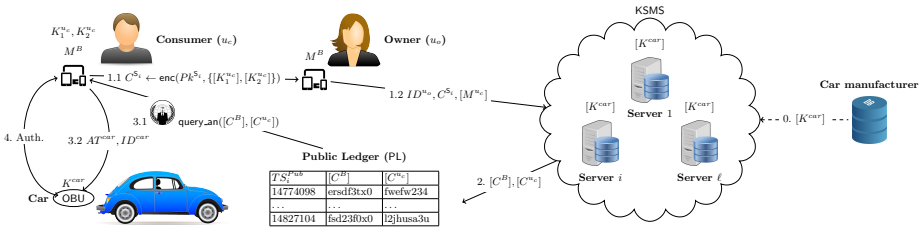

Fig. 2. SePCAR high level overview.

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

Fig. 3. The DB of CM (left) and the DB of the ith server S_i (right).

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 $K_y^{car_{u_o}}$.

Car booking allows u_o and u_c to agree on the booking details, i.e., $M^B = \{\mathsf{hash}(Cert^{u_c}), ID^{car}, L^{car}, CD^{u_c}, AC^{u_c}, ID^B\}$, where $\mathsf{hash}(Cert^{u_c})$ 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 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 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 $\mathsf{prf}()$ 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, $C^{\mathsf{S}_i} = \mathsf{enc}(Pk^{\mathsf{S}_i}, \{[K_1^{u_c}], [K_2^{u_c}]\})$, such that only the corresponding S_i can

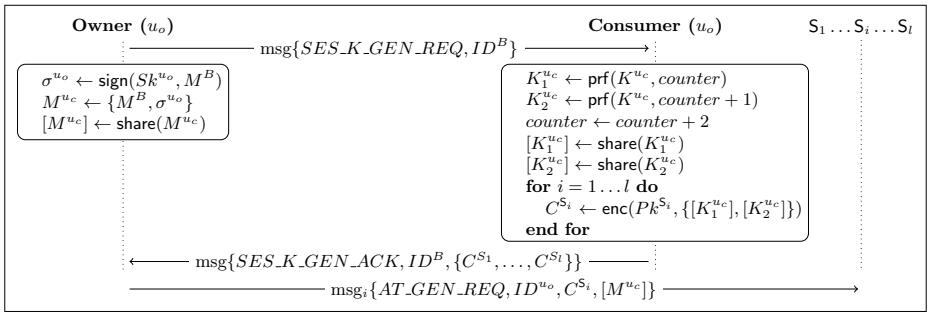

Fig. 4. Step 1: session keys generation and data distribution.

access the corresponding shares. Finally, u_c forwards to u_o an acknowledgment message, *SES_K_GEN_ACK*, along with ID^B and $\{C^{\mathsf{S}_1}, \ldots, C^{\mathsf{S}_l}\}$.

While waiting for the response of u_c, the owner u_o signs M^B with her private key, i.e., $\sigma^{u_o} = \mathsf{sign}(Sk^{u_o}, M^B)$. 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 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.

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 $\mathsf{eqz}()$ 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 $\vec{D}^{car}$ of length n, i.e.,

$$\vec{D}^{car} = \left(\overset{1}{[0]} \cdots [0] \overset{y}{[1]} [0] \cdots \overset{n}{[0]} \right) .$$

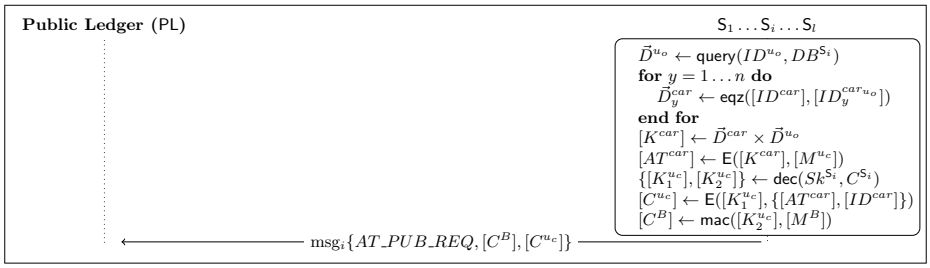

Fig. 5. Step 2: access token generation.

Each S_i then multiplies $\vec{D}^{car}$ and $\vec{D}^{u_o}$ to generate a third vector of length 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 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 $\mathsf{mac}()$ 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 Distribution and Verification. 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 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 the publication of the encrypted token. Then PL sends an acknowledgement of the 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} and the $\mathsf{query\_an}()$ function to anonymously retrieve $[C^{u_c}]$ and $[C^B]$ from PL, such that PL cannot identify u_c. Then, u_c uses the $\mathsf{open}()$ 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 $\mathsf{mac}()$ function with $K_2^{u_c}$ and M^B as inputs. In the 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, $\{AT^{car}, ID^{car}\}$.

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

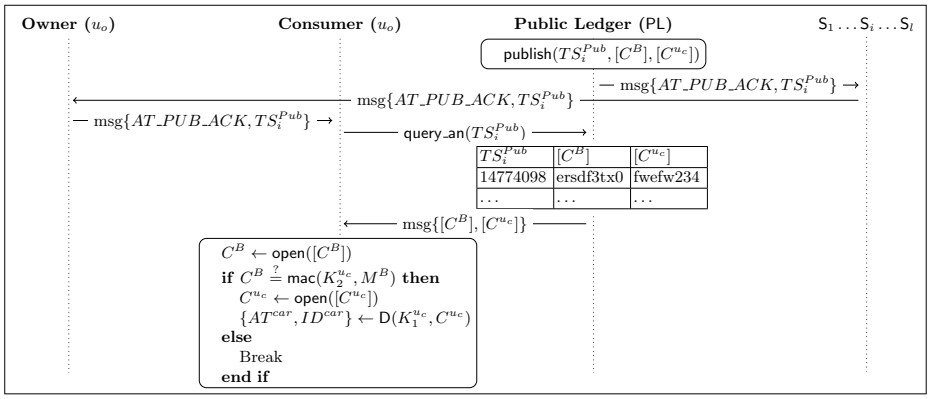

Fig. 6. Step 3: access token distribution and verification.

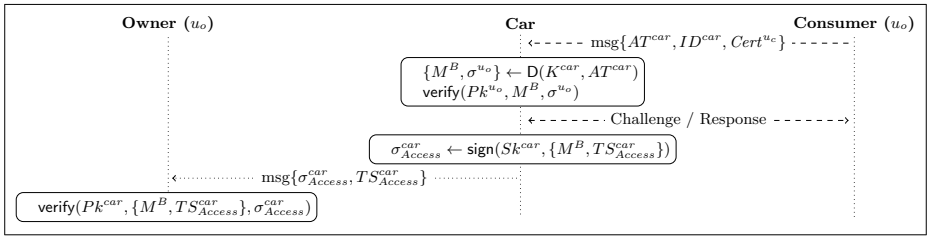

Fig. 7. Step 4: car access. Dashed lines represent close range communication.

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 hash($Cert^{u_c}$) in M^B), it can use any challenge-response protocol based on public/private key [19] and RFIDs [20]. 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 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 specifically, u_o generates session keys, requests an access token, queries the PL, and sends the token to the car using long range asynchronous communication channel such as LTE.

5 Security and Privacy Analysis

We prove that SePCAR satisfies the security and privacy requirements of Sect. 2, provided that its underlying cryptographic primitives are sufficiently secure. The theorem statement and the proof given below 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 [25],*
- *the pseudo-random function* prf *is multi-key secure [24],*
- *the public-key encryption scheme* enc *is multi-key semantically secure [5],*
- *the symmetric key encryption scheme* E *is multi-key chosen-plaintext secure [6],*
- *the MAC function* mac *is multi-key existentially unforgeable [25], and*
- *the hash function* hash *is collision resistant [45],*

then SePCAR fulfils the security and privacy requirements of Sect. 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. [5] 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

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

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

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 $\mathsf{verify}(Pk^{u_o}, M^B, \sigma^{u_o})$ 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 $\mathsf{hash}(Cert^{u_c}) = \mathsf{hash}(Cert^{u_c\prime})$, which is computationally infeasible 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 pseudo-random function prf that is secure, i.e., that is indistinguishable from a random function. This implies that all session keys are drawn independently and uniformly at random. In addition, the symmetric encryption scheme 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, $\mathsf{verify}(Pk^{u_o}, M^B, \sigma^{u_o})$.

The consumer u_c verifies the origin through the verification of the MAC algorithm, $C^B \stackrel{?}{=} \mathsf{mac}(K_2^{u_c}, M^B)$. 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, $\mathsf{verify}(Pk^{car}, \{M^B, TS^{car}_{Access}\}, \sigma^{car}_{Access})$ 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 cannot 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^{u_o} stored). Sending the confirmation signature σ^{car}_{Access} can also be done offline. □

6 Performance Evaluation

Below we analyse the theoretical complexity and practical efficiency of SePCAR.

Theoretical Complexity. The complexity of MPC protocols is typically measured by the number of communication rounds produced by non-linear operations, as linear operations can usually be performed without any information exchange and are virtually free of charge. In one evaluation of SePCAR, the non-linear operations performed by the KSMS servers are (i) the retrieval of the car key through multiple calls of the eqz functionality using the ID^{car} and their counterparts in $\vec{D}^{car}$ as parameters, and (ii) have 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 [35]. 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 Sect. 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 number of communication rounds can thus be expressed as:

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

Efficiency. Our protocol is agnostic towards the underlying multiparty protocol. In our experiments we have incorporated the 3-party semi-honest protocol by Araki et al. [3], given its relative efficiency of AES calls compared to alternatives such as, [16, 32]. 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 Sect. 4. An allocation of the time on the different steps is provided 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 ms) and for sign we use RSA with SHA-2 with a 512-bit output ($\approx$ 50 ms). The prf is implemented using AES in CTR mode ($\approx 2\mu$s). For all these functions we

use OpenSSL [39]. The share function is implemented by the sharing primitive introduced by Araki et al. [3].

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: $\mathsf{hash}(Cert^{u_c})$ of 512-bits, ID^{car} of 32-bits, L^{car} of 64-bits, CD^{u_c} of 96-bits, AC^{u_c} of 8-bits, ID^{B} of 32-bits and σ^{u_o} of 512-bits. The booking details M^B are of size 768-bits (including padding) and the final access token AT^{u_c} is of size 1408-bits (including padding). For the dec function we use RSA with 2048-bit keys ($\approx$ 2 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 primitives proposed by Araki et al. [3], and we use the multiparty AES method of Damgård and Keller [14]. Using this method, 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, hence encryption requires $\nu = 100$ communication rounds. From (3) we obtain that in our simulation the total number of communication rounds is

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

Key expansion for different keys needs to be performed only 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 primitive of Araki et al. [3], and mac is implemented using AES in CBC mode ($\approx$ 13 ms).

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 our simulation for SePCAR in C++ and evaluated it using a machine equipped with an Intel $i7$, 2.6 Ghz CPU and 8GB of RAM.[5] The communication within the KSMS was simulated using socket calls and latency parameters. We used the setting from Araki et al. [3] to simulate the LAN latency ($\approx$ 0.13 ms) and from Ramamurthy et al. [41] for Wi-Fi ($\approx$ 0.50 ms). We did not assume any specific network configuration for our experimentation.

[5] The implementation can be obtained from https://bitbucket.org/Siemen11/sepcar.

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 Tor [50])
Total		1.551 ± 0.043 (+1 Tor)

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 designed a prototype as proof-of-concept. SePCAR provides a complementary solution to physical keys, aiming for those that hold portable devices and want a dynamic and efficient way to access to a car. As future work, we plan to extend SePCAR to support additional operations such as booking and payment. It would also be interesting to investigate potential modifications of the protocol, in order to provide security and privacy guarantees while KSMS, CM, and PL are active adversaries.

Acknowledgments. This work was supported in part by the Research Council KU Leuven: C16/15/058 and GOA TENSE (GOA/11/007). Bart Mennink is supported by a postdoctoral fellowship from the Netherlands Organisation for Scientific Research (NWO) under Veni grant 016.Veni.173.017.

References

1. ACEA: Carsharing: Evolution, Challenges and Opportunities. https://goo.gl/NTec4l, accessed April, 2017
2. 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. pp. 430–454. Springer (2015)
3. 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 CCS. pp. 805–817 (2016)
4. Balasch, J., Rial, A., Troncoso, C., Preneel, B., Verbauwhede, I., Geuens, C.: PrETP: Privacy-Preserving Electronic Toll Pricing. In: USENIX. pp. 63–78 (2010)
5. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Advances in Cryptology - EUROCRYPT. pp. 259–274 (2000)
6. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS. pp. 394–403 (1997)
7. Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. 61(3), 362–399 (2000), http://dx.doi.org/10.1006/jcss.1999.1694

8. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC. pp. 1–10. ACM (1988)
9. Bert, J., Collie, B., Gerrits, M., Xu, G.: What's ahead for car sharing?: The new mobility and its impact on vehicle sales. https://goo.gl/ZmPZ5t, accessed June, 2017
10. BMW: DriveNow Car Sharing. https://drive-now.com/, accessed Nov., 2016
11. Catrina, O., Hoogh, S.D.: Improved primitives for secure multiparty integer computation. In: SCN. pp. 182–199 (2010)
12. Council of the EU Final Compromised Resolution: General Data Protection Regulation. http://www.europarl.europa.eu, accessed Feb., 2015
13. 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. LNCS, vol. 3876, pp. 285–304. Springer (2006)
14. Damgård, I., Keller, M.: Secure multiparty AES. In: FC. pp. 367–374 (2010)
15. 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. LNCS, vol. 7485, pp. 241–263. Springer (2012)
16. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: CRYPTO '12. LNCS, vol. 7417, pp. 643–662. Springer (2012)
17. Daring Fireball: Regarding Ubers New Always Location Tracking. https://goo.gl/L1Elve, accessed April, 2017
18. Deloitte: Smart mobility: Reducing congestion and fostering faster, greener, and cheaper transportation options. https://goo.gl/NnOk5X, accessed April, 2017
19. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Cryptography 2(2), 107–125 (1992), http://dx.doi.org/10.1007/BF00124891
20. Dmitrienko, A., Plappert, C.: Secure free-floating car sharing for offline cars. In: ACM CODASPY. pp. 349–360 (2017)
21. Enev, M., Takakuwa, A., Koscher, K., Kohno, T.: Automobile driver fingerprinting. PoPETs 2016(1), 34–50 (2016)
22. EVITA: E-safety Vehicle Intrusion Protected Applications (EVITA). http://www.evita-project.org/, accessed Nov., 2016
23. 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. pp. 225–255 (2017)
24. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986)
25. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
26. GOV.UK: Reducing mobile phone theft and improving security. https://goo.gl/o2v99g, accessed April, 2017
27. Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: MPC-Friendly Symmetric Key Primitives. Cryptology ePrint Archive, Report 2016/542 (2016)
28. International Organization for Standardization: ISO/IEC 9797-1:2011. https://www.iso.org/standard/50375.html, accessed June, 2017
29. Internet Engineering Task Force: PKCS #1: RSA Cryptography Specifications Version 2.0. https://tools.ietf.org/html/rfc2437, accessed June, 2017

30. Internet Engineering Task Force: Use of the RSA-KEM Key Transport Algorithm in the Cryptographic Message Syntax (CMS). https://tools.ietf.org/html/rfc5990, accessed June, 2017
31. INVERS: Make Mobility Shareable. https://invers.com/, accessed April, 2017
32. Keller, M., Orsini, E., Scholl, P.: MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In: ACM SIGSAC. pp. 830–842 (2016)
33. Kerschbaum, F., Lim, H.W.: Privacy-preserving observation in public spaces. In: ESORICS. pp. 81–100 (2015)
34. Khodaei, M., Jin, H., Papadimitratos, P.: Towards deploying a scalable & robust vehicular identity and credential management infrastructure. CoRR abs/1601.00846 (2016)
35. Lipmaa, H., Toft, T.: Secure equality and greater-than tests with sublinear online complexity. In: ICALP (2). pp. 645–656 (2013)
36. Micali, S.: Algorand: The efficient and democratic ledger. arXiv:1607.01341 (2016)
37. Mustafa, M.A., Zhang, N., Kalogridis, G., Fan, Z.: Roaming electric vehicle charging and billing: An anonymous multi-user protocol. In: IEEE SmartGridComm. pp. 939–945 (2014)
38. Naphade, M.R., Banavar, G., Harrison, C., Paraszczak, J., Morris, R.: Smarter cities and their innovation challenges. IEEE Computer 44(6), 32–39 (2011)
39. OpenSSL: Cryptography and SSL/TLS Toolkit. https://www.openssl.org/, accessed April, 2017
40. PRESERVE: Preparing Secure Vehicle-to-X Communication Systems (PRESERVE). https://www.preserve-project.eu/, accessed Nov., 2016
41. Ramamurthy, H., Prabhu, B., Gadh, R., Madni, A.M.: Wireless industrial monitoring and control using a smart sensor platform. IEEE Sensors Journal 7(5), 611–618 (2007)
42. Raya, M., Papadimitratos, P., Hubaux, J.: Securing vehicular communications. IEEE Wireless Commun 13(5), 8–15 (2006)
43. reddit: Identifying Muslim cabbies from trip data and prayer times. https://goo.gl/vLrW1s, accessed April, 2017
44. Rogaway, P.: Formalizing human ignorance. In: Progressin Cryptology - VIETCRYPT 2006, First International Conferenceon Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers. pp. 211–228 (2006)
45. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: FSE. LNCS, vol. 3017, pp. 371–388. Springer (2004)
46. 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), 5–34 (2013)
47. Stinson, D.R.: Some observations on the theory of cryptographic hash functions. Des. Codes Cryptography 38(2), 259–277 (2006), http://dx.doi.org/10.1007/s10623-005-6344-y
48. Symeonidis, I., Mustafa, M.A., Preneel, B.: Keyless car sharing system: A security and privacy analysis. In: IEEE ISC2. pp. 1–7 (2016)
49. The Guardian: Hell of a ride: even a PR powerhouse couldn't get Uber on track. https://goo.gl/UcIihE, accessed April, 2017
50. Tor: TorMETRICS. https://metrics.torproject.org/torperf.html, accessed April, 2017
51. Tor Project: Protect your privacy. Defend yourself against network surveillance and traffic analysis. https://www.torproject.org/, accessed April, 2017

52. Troncoso, C., Danezis, G., Kosta, E., Balasch, J., Preneel, B.: PriPAYD: Privacy-Friendly Pay-As-You-Drive Insurance. IEEE TDSC 8(5), 742–755 (2011)
53. Trusted Computing Group: TPM 2.0 Library Profile for Automotive-Thin. https://goo.gl/fy3DxD, accessed June, 2016
54. United States Patent and Trademark Office. Applicant: Apple Inc: Accessing a vehicle using portable devices. https://goo.gl/a9pyX7, accessed June, 2017
55. USA TODAY: Toyota will test keyless car sharing. https://goo.gl/C9iq34, accessed Nov., 2016
56. Volvo: Worth a Detour. https://www.sunfleet.com/, accessed Nov., 2016
57. Wielinski, G., Trépanier, M., Morency, C.: Electric and hybrid car use in a free-floating carsharing system. International Journal of Sustainable Transportation 11(3), 161–169 (2017)

A SePCAR Complete Representation.

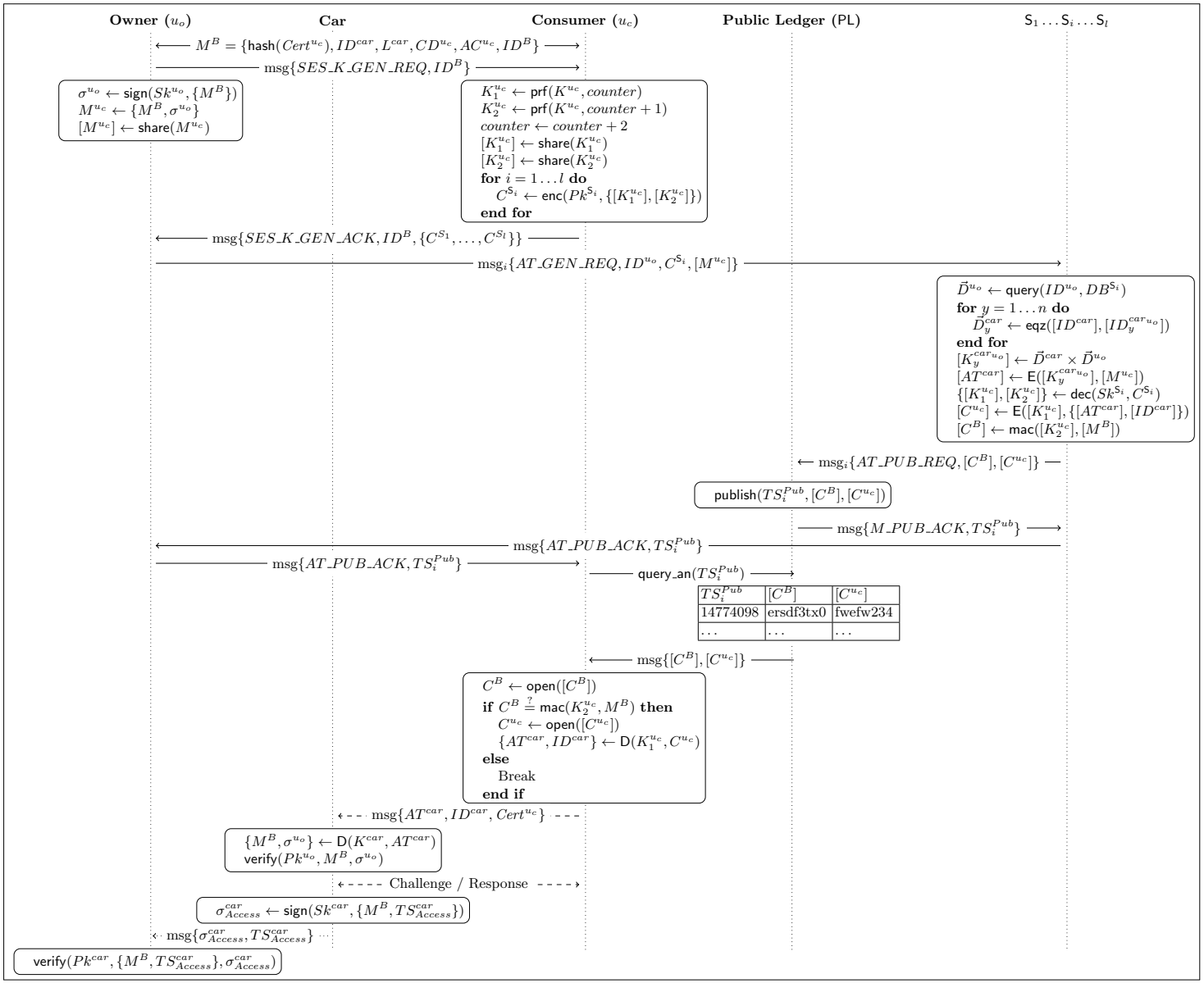

Fig. 8. SePCAR complete representation.

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. [25], for pseudorandom functions by Goldreich et al. [24], for public key encryption by Bellare et al. [5], and for symmetric key encryption by Bellare et al. [6].

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. [5] 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 $\mathrm{Func}(f)$ 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* $\mathrm{sign} = (\mathsf{keygen}, \mathsf{sign}, \mathsf{verify})$. *For any adversary $\mathcal{A}$, we define its advantage in breaking the μ-multikey existential unforgeability as*

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

where "forges" means that $\mathcal{A}$ outputs a tuple (i, M, σ) such that $\mathsf{verify}(Pk^i, M, \sigma) = 1$ *and M has never been queried to the i-th signing oracle. We define by* $\mathrm{Adv}^{\mu\text{-euf}}_{\mathrm{sign}}(q, t)$ *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* $\mathrm{prf} = (\mathsf{keygen}, \mathsf{prf})$. *For any adversary $\mathcal{A}$, we define its advantage in breaking the μ-multikey pseudorandom function security as*

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

We define by $\mathrm{Adv}^{\mu\text{-prf}}_{\mathrm{prf}}(q, t)$ *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* $\mathrm{enc} = (\mathsf{keygen}, \mathsf{enc}, \mathsf{dec})$. *For any adversary $\mathcal{A}$, we define its advantage in breaking the μ-multikey semantic security as*

$$\mathrm{Adv}^{\mu\text{-pke}}_{\mathrm{enc}}(\mathcal{A}) = \Big|\Pr\left((Pk^1, Sk^1), \ldots, (Pk^\mu, Sk^\mu) \xleftarrow{\$} \mathsf{keygen} \ : \ \mathcal{A}^{\mathcal{O}_0}(Pk^i) = 1\right) - \Pr\left((Pk^1, Sk^1), \ldots, (Pk^\mu, Sk^\mu) \xleftarrow{\$} \mathsf{keygen} \ : \ \mathcal{A}^{\mathcal{O}_1}(Pk^i) = 1\right)\Big| \ ,$$

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

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

$$\mathrm{Adv}^{\mu\text{-ske}}_{\mathrm{E}}(\mathcal{A}) = \Big|\Pr\left(K^1, \ldots, K^\mu \xleftarrow{\$} \mathsf{keygen} \ : \ \mathcal{A}^{\mathsf{E}(K^i, \cdot)} = 1\right) - \Pr\left(\$^1, \ldots, \$^\mu \xleftarrow{\$} \mathrm{Func}(\mathsf{E}) \ : \ \mathcal{A}^{\$^i} = 1\right)\Big| \ .$$

We define by $\mathrm{Adv}^{\mu\text{-ske}}_{\mathrm{E}}(q, t)$ 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* $\mathrm{mac} = (\mathsf{keygen}, \mathsf{mac})$. *For any adversary $\mathcal{A}$, we define its advantage in breaking the μ-multikey existential unforgeability as*

$$\mathrm{Adv}^{\mu\text{-mac}}_{\mathrm{mac}}(\mathcal{A}) = \Pr\left(K^1, \ldots, K^\mu \xleftarrow{\$} \mathsf{keygen} \ : \ \mathcal{A}^{\mathsf{mac}(K^i, \cdot)} \textit{ forges}\right) \ ,$$

where "forges" means that $\mathcal{A}$ outputs a tuple (i, M, σ) such that $\mathsf{mac}(K^i, M) = \sigma$ and M has never been queried to the i-th MAC function. We define by $\mathrm{Adv}^{\mu\text{-mac}}_{\mathrm{mac}}(q, t)$ 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 $\mathrm{Adv}^{\mathrm{col}}_{\mathsf{hash}}(t)$. 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 [45, 47, 44] 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*

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

where μ_o denotes the maximum number of u_os, μ_c the maximum number of u_cs, 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_os and CM are honest-but-curious whereas u_cs and outsiders may be malicious and actively deviate from the protocol. Cars are trusted.

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

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

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

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

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, $\pi^{car_{u_o}}$ for the cars and unique π^{u_c}'s for every new encryption under u_c's session keys, at the cost of

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

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

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 $\$^{u_c}$ 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 $\pi^{car_{u_o}} \leftarrow \mathsf{query}(ID^{u_o}, DB^{\mathsf{KSMS}})$ 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

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

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

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 $\mathsf{verify}(Pk^{u_o}, M^B, \sigma^{u_o})$ 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

$$\mathsf{E}_1 \;:\; \mathcal{A} \text{ forges } \mathsf{sign}(Sk^{u_o}, \cdot) \text{ for some } Sk^{u_o}\,. \tag{9}$$

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

$$\mathsf{E}_2 \;:\; \mathcal{A} \text{ finds a collision for } \mathsf{hash}\,. \tag{10}$$

We thus obtain that a violation of SR2 implies $\mathsf{E}_1 \vee \mathsf{E}_2$.

SR3 - Confidentiality of AT^{car}. The access token is generated by the KSMS obliviously (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, $\mathsf{E}_{K^{car}}$ 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, $\mathsf{verify}(Pk^{u_o}, M^B, \sigma^{u_o})$. The consumer u_c verifies the origin through the verification of the MAC function, $C^B \stackrel{?}{=} \mathsf{mac}(K_2^{u_c}, M^B)$. 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

$$\mathsf{E}_3 \ : \ \mathcal{A} \text{ forges } \mathsf{mac}(K_2^{u_c}, \cdot) \text{ for some } K_2^{u_c} \,. \tag{11}$$

We thus obtain that a violation of SR6 implies $\mathsf{E}_1 \vee \mathsf{E}_3$.

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, $\mathsf{verify}(Pk^{car}, \{M^B, TS^{car}_{Access}\}, \sigma^{car}_{Access})$ at the end of the protocol. Security breaks only if the signature scheme is forgeable. Denote the event that this happens by

$$\mathsf{E}_4 \ : \ \mathcal{A} \text{ forges } \mathsf{sign}(Sk^{car}, \cdot) \text{ for some } Sk^{car} \,. \tag{12}$$

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 $\pi^{car_{u_0}}$ and u_o's public key Pk^{u_o} stored). Sending the confirmation signature σ^{car}_{Access} 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:

$$\mathrm{Adv}^{\mu_c\text{-prf}}_{\mathrm{prf}}(2q,t) + \mathrm{Adv}^{l\text{-pke}}_{\mathrm{enc}}(t) + \mathrm{Adv}^{q+\mu_{\mathrm{car}}\text{-ske}}_{\mathrm{E}}(2q,t) + \Pr\left(\mathsf{E}_1 \vee \mathsf{E}_2 \vee \mathsf{E}_3 \vee \mathsf{E}4\right) \ll 1\,.$$

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

$$\Pr\left(\mathsf{E}_1 \vee \mathsf{E}_2 \vee \mathsf{E}_3 \vee \mathsf{E}4\right) \leq \mathrm{Adv}^{\mu_o+\mu_{\mathrm{car}}\text{-euf}}_{\mathrm{sign}}(2q,t) + \mathrm{Adv}^{q\text{-mac}}_{\mathrm{mac}}(q,t) + \mathrm{Adv}^{\mathrm{col}}_{\mathsf{hash}}(t)\,,$$

which completes the proof. □

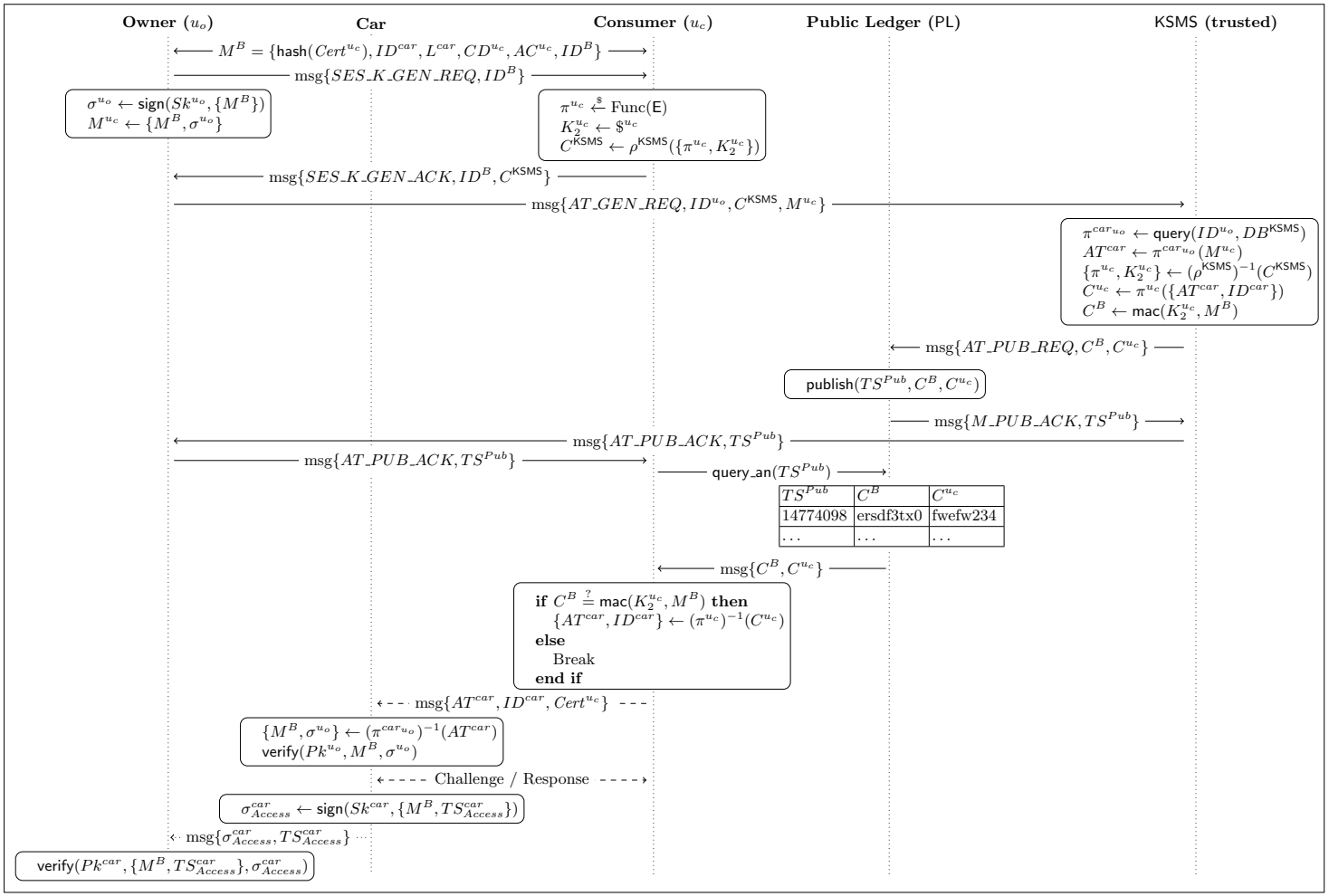

Fig. 9. Simplified representation of SePCAR for the proof of Theorem 2.