HERMES: Scalable, Secure, and Privacy-Enhancing Vehicle Access System

Iraklis Symeonidis*, Dragos Rotaru†‡, Mustafa A. Mustafa§‡, Bart Mennink¶, Panos Papadimitratos*
*NSS, KTH, Sweden, †Cape Privacy, ‡imec-COSIC, KU Leuven, Belgium, §Department of Computer Science, The University of Manchester, UK, ¶Digital Security Group, Radboud University, Netherlands
Email: *irakliss@kth.se, †dragos.rotaru@esat.kuleuven.be, ‡mustafa.mustafa@manchester.ac.uk,
¶b.mennink@cs.ru.nl, *papadim@kth.se

***Abstract*—We propose HERMES a scalable, secure, and privacy-enhancing system, which allows users to share and access vehicles. HERMES outsources the vehicle access token generation to a set of untrusted servers, utilizing several cryptographic primitives with secure multi-party computation efficiently. It conceals the vehicle secret keys and transaction details from the servers such as vehicle booking details, access token information, and user-vehicle identities. It also provides user accountability in case of disputes. We prove that HERMES meets its security and privacy requirements. Moreover, we demonstrate that HERMES scales for a large number of users and vehicles, making it practical for real-world deployments. To achieve high-performance computations, we evaluate HERMES over two different multi-party computation protocols for Boolean and arithmetic circuits. We provide a detailed comparison of their performance, together with other state-of-the-art access provision protocols. Through a proof-of-concept implementation, our performance analysis demonstrates that HERMES requires only $\approx$ 61ms for a single-vehicle access provision. At the same time, it handles 546 and 84 access token generations per second from a single-vehicle owner and large branches of rental companies with over a thousand vehicles, respectively.**

I. Introduction

Vehicle-sharing is an emerging smart mobility service leveraging modern technology to enable users to share their vehicles with others or access shared vehicles. The service has been gaining popularity in recent years. Statistics demonstrate an increase of the worldwide number of users of vehicle-sharing services from 2012 to 2014 by 170% (total 5 million) [1], with a tendency to reach a total of 26 million by 2021[1] [2]. Moreover, several companies, including Volvo [3], BMW [4], Toyota [5], and Apple [6], have been already investing in vehicle-sharing services. For instance, Apple announced the "CarKey" API in the first quarter of 2020. The API allows users to (un)lock and start a vehicle using an iPhone or Apple Watch. "CarKey" can also be shared with other people, such as family members, enabling vehicle-sharing [7], [8]. With the use of in-vehicle telematics and omnipresent portable devices, such as smartphones, vehicle owners can distribute temporary digital keys, a.k.a. Access Tokens (ATs), to other users [9]. Vehicle Sharing Systems (VSSs) enable dynamic and occasional use of multiple types of vehicles (e.g., car, motorbike, scooter), catering to consumers' diverse needs and preferences [10], [11], [12]. Beyond user convenience, VSS provides better utilization of available vehicles, thus contributing towards sustainable smart cities. This, in turn, leads to positive effects such as a reduction of emissions [13], a decrease of city congestion [14], and more economical use of parking space [15].

Despite these advantages, a major concern is that information collected in VSS can jeopardize the system's security. An adversary may eavesdrop and attempt to tamper with the vehicle sharing details, extract the key of a vehicle stored in untrusted devices, generate a rogue AT to access or deny having accessed a vehicle maliciously. These are significant concerns that require VSS to deploy security mechanisms to ensure that vehicle-sharing details cannot be tampered with by unauthorized entities, digital vehicle-keys are stored securely and attempts to use rogue ATs are blocked. In addition, VSSs also introduce various other concerns ranging from connectivity issues [16], [17], key revocation when a user's device is stolen [18], and dispute resolution. Considering disputes resolution, the VSS needs to support user accountability while keeping their private information protected. Existing efforts [16] on these security issues rely on a centralized Trusted Provider (TP) which has access to the master key of vehicles, and collects and stores all the information exchanged between the vehicle provider and their users for every vehicle access provision.

However, the privacy of VSS is equally important to security. A major concern with VSS is collecting personal and potentially sensitive data regarding users and their vehicles. An adversary may eavesdrop on data exchanges to infer sensitive information about its users. For example, Enev et al. [19] demonstrated that it is possible to reach high identification rates of drivers, with 87% to 99% accuracy, based on data collected by the sensors of a vehicle over 15 minutes of open-road driving. An adversary can try linking two vehicle-sharing requests of the same user or vehicle, identify vehicle usage patterns, and deduce users' sharing preferences. These preferences can be established by collecting data about sharing patterns, such as time of use, duration, pickup location, when, where, and with whom someone is sharing a vehicle [19]. The adversary can also infer sensitive information about users' race and religious beliefs [20] or their health status by identifying vehicles for special-need passengers. Such user profiling is a direct violation of the General Data Protection Regulation (GDPR) [21]. Thus, a

[1]Note that these predictions were made in pre-COVID-19 times.

VSS system needs to preserve the unlinkability of any two requests of a consumer, keep the consumer's identity and the vehicle anonymous, and indistinguishable between sharing operations as the generation, update, or revocation. Towards addressing these privacy challenges, a state-of-the-art VSS solution, named SePCAR [22], proposes to deploy Multiparty Computation (MPC). It focuses on privacy-preserving access token provision, deploying multiple non-colluding servers for the generation and distribution of vehicle ATs.

Considering scalability in a real-world setting, VSS needs to support a large number of users with multiple vehicles per user. A variance in the number of vehicles could range from a few vehicles for private individuals to thousands of vehicles for large branches of companies [1], such as in car-rental scenarios [23]. However, achieving scalability in VSSs, in addition to providing security and privacy safeguards, is not straightforward, as these safeguards can significantly affect the performance of a VSS with a large number of users and vehicles. For example, SePCAR [22], although efficient (1.55 seconds for a car access provision), is limited to a single evaluation of the protocol. It has not been tested on how it scales in a real-world setting with a large number of users and vehicles per user. Hence, to the best of our knowledge, there is no VSS solution in the literature that provides security and privacy guarantees while at the same time being scalable. This work aims to fill this gap.

We present HERMES, an efficient, secure, and privacy-enhancing system for vehicle access provision that supports dispute resolution while protecting users' privacy. HERMES is based on SePCAR [22], but fundamentally differs in certain design choices to make it scalable and more efficient. Specifically, the contributions of this work are:

1) HERMES deploys multiparty computation primitives to ensure that ATs are generated without the VSS learning vehicle-sharing details and vehicle secret keys. With the use of a public ledger, it also ensures the unlinkability of any consumer's two requests, keeps the consumer's identity and the vehicle anonymous, and indistinguishable between sharing operations while generation, update, or revocation. Besides that, HERMES also supports dispute resolution without compromising users' private information while keeping users accountable. We also prove that the HERMES is secure and meets its appropriate security and privacy requirement.
2) We demonstrate that HERMES is highly efficient under real-world settings of VSSs. Our design and implementation is tailored to supporting a VSS with users ranging from private individuals with only a few vehicles to large branches in car-rental scenarios with over a thousand vehicles per branch. Our implementation and performance evaluation demonstrates that HERMES requires only $\approx$ 61ms for a single-vehicle access provision while it can handle multiple access token generations per second for individuals and branches of rental companies. We also tested and evaluated our results using two MPC instantiations for Boolean and arithmetic circuits, and we provide a comparison. We also analyze the complexity of these two protocols independently, provide a comparison, and motivate the two choices.

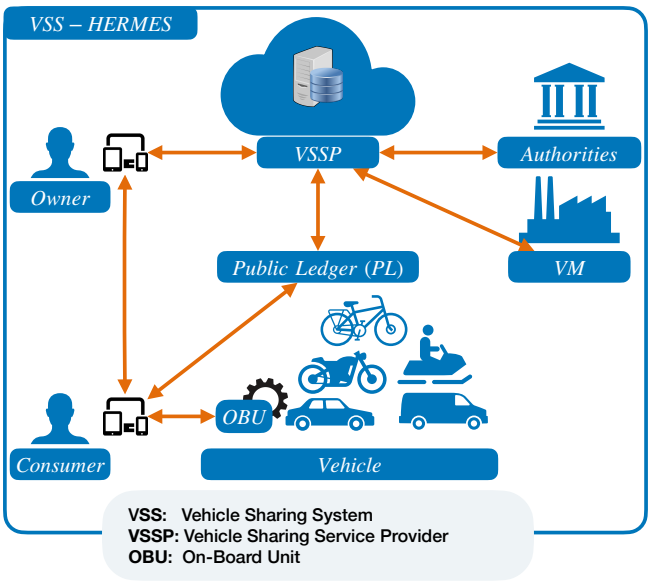

Fig. 1: Vehicle Sharing System (VSS) model with its entities and interactions.

The rest of the paper is organized as follows: Section II provides the system model and preliminary information of HERMES. Section III describes the cryptographic building blocks used in the design of HERMES, and Section IV describes the system in detail. Section V provides the security and privacy analysis of HERMES, and Section VI evaluates its theoretical complexity and practical efficiency. Section VII gives an overview of the state-of-the-art related work. Section VIII concludes our work.

II. System, Adversarial Models and Requirements

We describe the system model of VSS and specify the adversarial model and assumptions we use in our design, as well as, the functional, security, privacy and performance requirements which the system needs to satisfy.

1) System model: The VSS infrastructure, that comprises users, vehicles, their manufacturers, and authorities, is illustrated in Fig. 1. It considers two types of users: *owners* (u_o), individuals or vehicle rental companies willing to share (rent out) their vehicles, and *consumers* (u_c), individuals using vehicles available for sharing; both use portable Devices (PDs), such as smartphones to interact with VSS entities and each other. The On-Board Unit (OBU) is a hardware/software component that enables vehicle connectivity [24]. It has a wireless interface such as Bluetooth, NFC, or LTE and is part of the secure access management system of a *vehicle* [25]. The Vehicle Manufacturer (VM) is responsible for managing the digital keys that enable access into each vehicle. These keys are used for enabling vehicle sharing in VSS as well. The Vehicle Sharing Service Provider (VSSP) is a cloud infrastructure that facilitates the vehicle AT generation, distribution, update and revocation. It consists of *servers* that collaboratively generate ATs and publish them on the Public Ledger (PL). PL serves as a public bulletin board that guarantees the integrity of

the data [26]. The booking details are typically agreed upon by the owner and consumer prior to each vehicle-sharing session commences. We denote M^B, $AT^{vehicle_o}$, and $K^{vehicle_o}$ as Booking Details (BD), an Access Token (AT) for a vehicle, and for the vehicle secret key, respectively.

2) Adversarial Model: For our system design, we use the following adversary model. The communication channels are secure and authenticated among entities such as using SSL-TLS and NFC. The authorities are trusted entities. The VSSP, the VM and the PL are considered honest-but-curious entities. They execute the protocol correctly, but they may attempt to extract private information about users. Owners can be passive adversaries, as they hold information booking but they will not aim to alter the protocol. Consumers and outsiders can be active adversaries aiming to illegally access a vehicle, alter the booking information and hide on incidents. The vehicle, more specifically its OBU, is trusted and designed to resist deliberate or accidental physical destruction (i.e., it serves as black box). User PDs are untrusted as they can get stolen, lost, or broken.

3) Assumptions: For HERMES, we make the following assumptions. There is a Public Key Infrastructure (PKI) infrastructure [27] in place and each entity has its private/public-key pair with their corresponding digital certificates. The OBU is equipped with a Hardware Security Module (HSM) [27], [28] that supports secure key storage and cryptographic operations such as symmetric and public-key operations[2]. The VSSP servers are managed by non-colluding organizations, i.e., organizations with conflicting interests such as authorities, user unions, and VM. Before each evaluation, the Booking Details (BD) are agreed upon by the owner and consumer. Both keep the BD confidential against external parties. In BD, the identities of vehicle, owner and consumer, the location and time duration of the reservation can be specified. HERMES is agnostic to the specificities of BD drawing from the analogy of VSS.

4) System Design Requirements: The VSS should satisfy the following functional, security, privacy, and performance requirements [9], denoted as *FR*, *SR*, *PR*, and *PSR*, respectively.

Functional requirements:

- *FR1 – Offline access provision.* Access provision should be provided for locations where vehicles have limited (or no) network connectivity.
- *FR2 – Access token update and revocation by u_o.* No one but u_o can initiate AT update and revocation.

Security requirements:

- *SR1 – Confidentiality of M^B.* No one but the shared vehicle, u_o and u_c should have access to M^B.
- *SR2 – Authenticity of M^B.* The shared vehicle should verify the origin and integrity of M^B from u_o.
- *SR3 – Confidentiality of $AT^{vehicle_o}$.* No one but the shared vehicle and u_c should have access to $AT^{vehicle_o}$.
- *SR4 – Confidentiality of $K^{vehicle_o}$.* No one but $vehicle_o$ and VM should hold a copy of $K^{vehicle_o}$.
- *SR5 – Backward and forward secrecy of $AT^{vehicle_o}$.* Compromise of a key used to encrypt any $AT^{vehicle_o}$ should not compromise other tokens (future and past) published on the PL for any honest u_c.
- *SR6 – Non-repudiation of origin of $AT^{vehicle_o}$.* u_o should not be able to deny it agreed to the terms of M^B, and initiated the generation of the corresponding $AT^{vehicle_o}$.
- *SR7 – Non-repudiation of delivery of $AT^{vehicle_o}$.* u_c should not be able to deny it has obtained and used $AT^{vehicle_o}$ to open the vehicle (once it has done so).
- *SR8 – Accountability of users (i.e., owners and consumers).* The VSSP should be able to provide authorities with the transaction details of an access provision to a vehicle at the request of law enforcement without violating the privacy of other users.

Privacy requirements:

- *PR1 – Unlinkability of any two requests of u_c for any $vehicle_o$(s).* No one but the shared $vehicle_o$, u_o and u_c should be able to link two booking requests of any u_c for any $vehicle_o$ linking their identities, i.e., ID^{u_c}, and $ID^{vehicle_o}$.
- *PR2 – Anonymity of u_c and the $vehicle_o$.* No one but the shared vehicle, u_o and u_c should learn the identity of u_c and the vehicle.
- *PR3 – Indistinguishability of $AT^{vehicle_o}$ operation.* No one but the shared vehicle, u_o and u_c (if necessary) should be able to distinguish between $AT^{vehicle_o}$ generation, update and revocation.

Performance requirement:

- *PSR1 – Scalability.* The VSS should maintain its efficiency even when it grows to support a large number of vehicles and users.

III. Cryptographic Building Blocks

1) Cryptographic Primitives: HERMES uses the following cryptographic building blocks. The suggested instantiations are 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 specification [30].
- $K \leftarrow \mathsf{kdf}(K, counter)$ is a key derivation function using a master key and a counter as inputs. It can be based on a Pseudo-Random Function (PRF) and implemented using CTR mode with AES [31].[3]
- $c \leftarrow \mathsf{enc}(Pk, m)$ 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 [33].
- $c \leftarrow \mathsf{E}(K, m)$ and $m \leftarrow \mathsf{D}(K, c)$ are symmetric key encryption and decryption functions. These can be implemented using CTR mode with AES.
- $z \leftarrow \mathsf{hash}(m)$ is a cryptographic hash function. This function can be implemented using SHA-2 or SHA-3.

[2]As specified in the secure vehicle specifications such as EVITA [29] and PRESERVE [27].

[3]In our case, the message input is small, i.e., $\ll 2^{64}$ blocks for AES in CTR, and the generation is performed with side channel attacks not to be a concern [32].

- $t \leftarrow \mathsf{mac}(k, m)$ is a cryptographic Message Authentication Code (MAC) which outputs an authentication tag t given a key k and a message m. These can be implemented using CBC-MAC-AES or HtMAC-MiMC (see below).

Furthermore, we use $z \leftarrow \mathsf{query}(x, y)$ to denote the retrieval of the xth value from the yth database DB (to be defined in Sect. IV), 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 [34], aiming to anonymously retrieve a published record submitted using the $\mathsf{publish}(y)$ function.

2) Multiparty Computation: MPC allows a set of parties to compute a function over their inputs without revealing them. Following the seminal papers of Yao for the two-party case [35] and by Goldreich, Micali and Wigderson in the multiple parties setting [36], secure MPC has gained much traction in the past years with many open-source frameworks [37].

To reason about the MPC protocol that fits our setting, one has to pinpoint various parameters such as network settings, security model, or a number of parties. For more details on adversary capabilities, the reader can refer to the survey by Lindell [38]. In order to keep the protocol simple, we consider a fast network (10 Gbps LAN) and a passive adversary for three-party replicated secret sharing scheme [39], [40]. However, if one wants to upgrade to other MPC protocols secure against malicious parties, this can be done fairly simple by changing the way external parties provide inputs or get outputs using known techniques [41]. To evaluate a function on secret inputs using MPC, one needs to unroll the function to a series of additions and multiplications in a field. A core contribution of the paper is the integration of two different modes of operations for the secrecy of AT generation considering honest-but-curious set of servers in VSSP: one mode that uses HtMAC which works over a large field [40], while the other mode uses CBC-MAC-AES over a binary field [39].

Our algorithms use building blocks whose instantiation depends on the protocol type. However, they can be treated generically. This is also called an arithmetic black-box functionality [42] .The functionality mainly in use consists of:

- $[x] \leftarrow \mathsf{share}(x)$ secret shares an input x to all parties. The underlying secret sharing scheme is described in detail in Araki et al. [39].
- $c \leftarrow \mathsf{E}([k], [m])$ An encryption function, i.e., E, takes as inputs a secret shared key $[K]$ and a vector of 128 bit blocks $[m]$. For the $\mathbb{F}_2$ case, E is implemented using AES in counter mode. Concretely, the AES circuit description is the one from SCALE-MAMBA [43], which has 6400 AND gates. For the $\mathbb{F}_p$ case, MiMC is used as a PRF in counter mode as presented in [44] to take advantage of PRF invocations done in parallel.
- $t \leftarrow \mathsf{mac}([k], [m])$ is a tag generation function for secret shared key $[k]$ and message $[m]$. For the case when inputs are in a large field, we will not compute the MAC as above, but rather as $\mathsf{mac}([k], \mathsf{E}([k'], [m]))$. The reason is that, according to [44], we can obtain a more efficient cryptographic MAC in MPC by first computing $\mathsf{E}([k'], [m])$ in parallel with a secret shared key $[k']$, opening the result, and evaluate the MAC function in the clear. Their optimizations hold only for arithmetic circuits although they could likely be extended to boolean circuits as well. In the Boolean case, the mac function is implemented as CBC-MAC-AES. Note that for the $\mathbb{F}_2$ case there are more efficient ways to do this, but we keep CBC-MAC as a comparison baseline to previous work such as SePCAR [22].
- $[z] \leftarrow ([x] \stackrel{?}{=} [y])$ outputs a secret bit $[z]$ where $z \in \{0, 1\}$. If x is equal to y then set $z \leftarrow 1$ otherwise set $z \leftarrow 0$. Note that for the large field case there is a statistical security parameter sec, whereas for the $\mathbb{F}_2$ case the comparison is done with perfect security (i.e. no sec parameter). The equality operator is implemented using the latest protocols of Escudero et al. [45].
- $x \leftarrow \mathsf{open}([x])$ which takes a secret shared value $[x]$ and opens it, making x known to all parties.

IV. HERMES

A. Overview of HERMES

We provide a brief overview of HERMES, which is also illustrated in Fig. 2. We consider a single owner, consumer, and a shared vehicle for simplicity and without loss of generality.

Before HERMES commences, there are *vehicle key distribution* and establishing the details for the *vehicle booking* prerequisite steps. Considering VM as a TP for VSS, it holds all the secret keys of vehicles that produce. By vehicle owners registering their vehicles, the VSSP using the owner's identity retrieves the vehicle's identity and the corresponding key. Both the identity and vehicle key are transferred from VM to VSSP in a secret shared form, and thus preserving the secrecy of both towards the VSSP. For each commences of HERMES, the BD is established between the owner and the consumer, specifying information for vehicle sharing, such as the duration of the vehicle's reservation and location.

HERMES consists of four steps: *session keys generation and data distribution*, *access token generation*, *access token distribution and verification*, and *vehicle access*. During the session key generation and data distribution, three-session keys are generated by the consumer. One of these keys will be used to encrypt the generated AT at the VSSP servers so that only the consumer has access to it, while the other two keys will be used to generate an authentication tag of the BD such that the consumer can identify and retrieve the AT from the PL as well as verify that the beforehand agreed BD are included in the AT. Since the consumer considers the owner and the VSSP as honest-but-curious entities, it will conceal and forward the session keys towards the VSSP in secret shared form. To protect its identity from being leaked to the VSSP, the consumer avoids direct contact with the VSSP and forwards the session keys' shares the owner. Together with the concealed session keys, the owner forwards to VSSP the BD and its signature in a shared form to each S_i server. Upon the VSSP receiving the session keys and the booking details, the *access token generation* step commences. The vehicle key is retrieved from the database in each S_i server using an equality test over MPC, preserving the key's secrecy. The AT is generated under the vehicle's encryption, such that

TABLE I: Notation.

Symbol	Description
S_i, PL, VM	Set of VSSP servers, the ith server for $i \in \{1 \dots l\}$, Public ledger, vehicle manufacturer
u_o, u_c, $vehicle_o$	The owner, consumer, and vehicle
$ID^B, ID^{u_o}, ID^{u_c}, ID^{vehicle_o}$	ID of booking, u_o, u_c, vehicle
CD^{u_c}/AC^{u_c}, $L^{vehicle_o}$	Set of conditions/access rights under which u_c is allowed to access a vehicle, vehicle's location
DB^{VM} / DB^{S_i}	Database that VM holds with $(ID^{u_o}, ID^{vehicle_{u_o}}, K^{vehicle_{u_o}})$ / that S_i holds with $(ID^{u_o}, [ID^{vehicle_{u_o}}], [K^{vehicle_{u_o}}])$ for all owners (u_o's) and their registered vehicles
$\vec{D}^{u_o}$	Vehicle records $(ID_x^{u_o}, [ID_y^{vehicle_{u_o}}], [K_y^{vehicle_{u_o}}])$ of the xth u_o for the yth vehicle extracted (query) from DB^{S_i}, where $\lvert\vec{D}^{u_o}\rvert = n$
$[\vec{D}]^{vehicle_o}$	Matched, equality output (i.e., $\stackrel{?}{=}$), yth vehicle key $([\overset{1}{0}] \cdots [0][\overset{y}{1}][0] \cdots [\overset{n}{0}])$, where $\lvert\vec{D}^{vehicle_o}\rvert = n$
Pk^x / Sk^x, $Cert^{u_c}$	Public/private key pair of the VSS entity x, certificate of u_c
M^B	Booking details, i.e, $\{\mathsf{hash}(Cert^{u_c}), ID^{vehicle_o}, L^{vehicle_o}, CD^{u_c}, AC^{u_c}, ID^B\}$
σ^{u_o}, $\sigma_{Access}^{vehicle_o}$	Signature (sign output) of M^B with Sk^{u_o}, and $\{M^B, TS_{Access}^{vehicle_o}\}$ with $Sk^{vehicle_o}$
$K^{vehicle_o}$	Symmetric key of the vehicle
$K_{master}^{u_c}$	u_c's master key
$K_{enc}^{u_c}$	u_c's session key used for encryption of the AT generated from K^{u_c} and $counter$ (kdf output)
$\vec{K}_{tag}^{u_c}$	u_c's session key $K_{tag_{enc}}^{u_c}$, $K_{tag_{mac}}^{u_c}$ used for generating of the $AuthTag$ generated from K^{u_c} and $counter + 1$/$counter + 2$ (kdf output)
$AuthTag^{M_B}$, $[AuthTag^{M_B}]$	The authentication tag of M^B with $\vec{K}_{tag}^{u_c}$, and $[M^B]$ with $[\vec{K}_{tag}^{u_c}]$
M^{u_c}, AT^{u_c}	Concatenation of M^B with σ^{u_o}, AT as the encryption (E output) of M^{u_c} with $K^{vehicle_o}$
C^{S_i}	Ciphertext (enc output) of session keys $\{[K_{enc}^{u_c}], [\vec{K}_{tag}^{u_c}]\}$ with Pk^{S_i}
$[C^{u_c}]$	Ciphertext (E output) of $\{[AT^{u_c}], [ID^{vehicle_o}]\}$ with $[K_{enc}^{u_c}]$
TS_i^{Pub}, $TS_{Access}^{vehicle_o}$	Time-stamp of u_c accessing the shared vehicle, a record published (publish) on the PL submitted by S_i

only the vehicle itself can retrieve the BD. Moreover, the consumer session keys are used to encrypt the AT and create an authentication tag, such that only the consumer can identify and access the AT. Each of the servers S_i then forwards the encrypted AT and its authentication tag to the PL. The PL serves as a bulletin board and notifies the VSSP upon publishing the information. At the *access token distribution and verification* step, the consumer can identify and retrieve the corresponding AT. Since the consumer considers the PL as honest-but-curious, it can hide its identity (i.e., IP address), querying the PL using an anonymous communication channel. It will also retrieve the AT, and the AT will be used by the vehicle to verify and allow access to the consumer for the predefined booking duration at the last step of *vehicle access*.

B. HERMES in detail

Below we discuss HERMES in detail, with a general overview picture given in Fig. 9. We first describe the *prerequisite* steps, and we complete the section with an overview of the possible additional operations after HERMES: *access token update and revocation* and *accountability of users*. Table I lists the notation used throughout the paper. A detailed overview of HERMES, along with additional material, can be found at the online full version [46].

1) Step 1 – Session key generation and data distribution: While u_o signs the booking details, i.e., M^B, u_c generates session keys for encryption and data authentication, i.e., $K_{enc}^{u_c}$ and $\vec{K}_{tag}^{u_c} = (K_{tag_{mac}}^{u_c}, K_{tag_{enc}}^{u_c})$, respectively. The generated material by u_c and u_o are sent via u_o to each S_i, such that the AT can be generated.

Prerequisite steps: Before HERMES commences, two prerequisite steps need to take place: *vehicle key distribution* and establishing the details for the *BD*.

a) Vehicle key distribution: It takes place immediately after the xth owner, $ID_x^{u_o}$, has registered her yth vehicle, $ID_y^{vehicle_{u_o}}$, with the VSSP. The VSSP retrieves from the VM database denoted by DB^{VM} the secret symmetric key of the vehicle and the corresponding identity of the owner in secret shared form, denoted by $[K_y^{vehicle_{u_o}}]$ and $[ID_y^{vehicle_{u_o}}]$, respectively. Then, it stores, $ID_x^{u_o}$, $[ID_y^{vehicle_{u_o}}]$ and $[K_y^{vehicle_{u_o}}]$ in its database denoted DB^{S_i}. For simplicity, in some parts of HERMES we will use ID^{u_o}, $ID^{vehicle_o}$ and $K^{vehicle_o}$ instead of $ID_x^{u_o}$, $ID_y^{vehicle_{u_o}}$ and $K_y^{vehicle_{u_o}}$.

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

2) Step 1 – Session key generation and data distribution: While u_o signs the booking details, i.e., M^B, u_c generates session keys for encryption and data authentication, i.e., $K_{enc}^{u_c}$ and $\vec{K}_{tag}^{u_c}$, respectively. The generated material by u_c and u_o are sent via u_o to each S_i, such that the AT can be generated.

In detail, as it is illustrated in Fig. 3, 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 the session keys, $K_{enc}^{u_c}$ and $\vec{K}_{tag}^{u_c}$. $K_{enc}^{u_c}$ will be used by the VSSP servers, S_i, to encrypt the AT, such that only u_c has access to it. Note that each S_i does encryption evaluations in a secret shared way. $\vec{K}_{tag}^{u_c}$ is used to generate an authentication tag which will

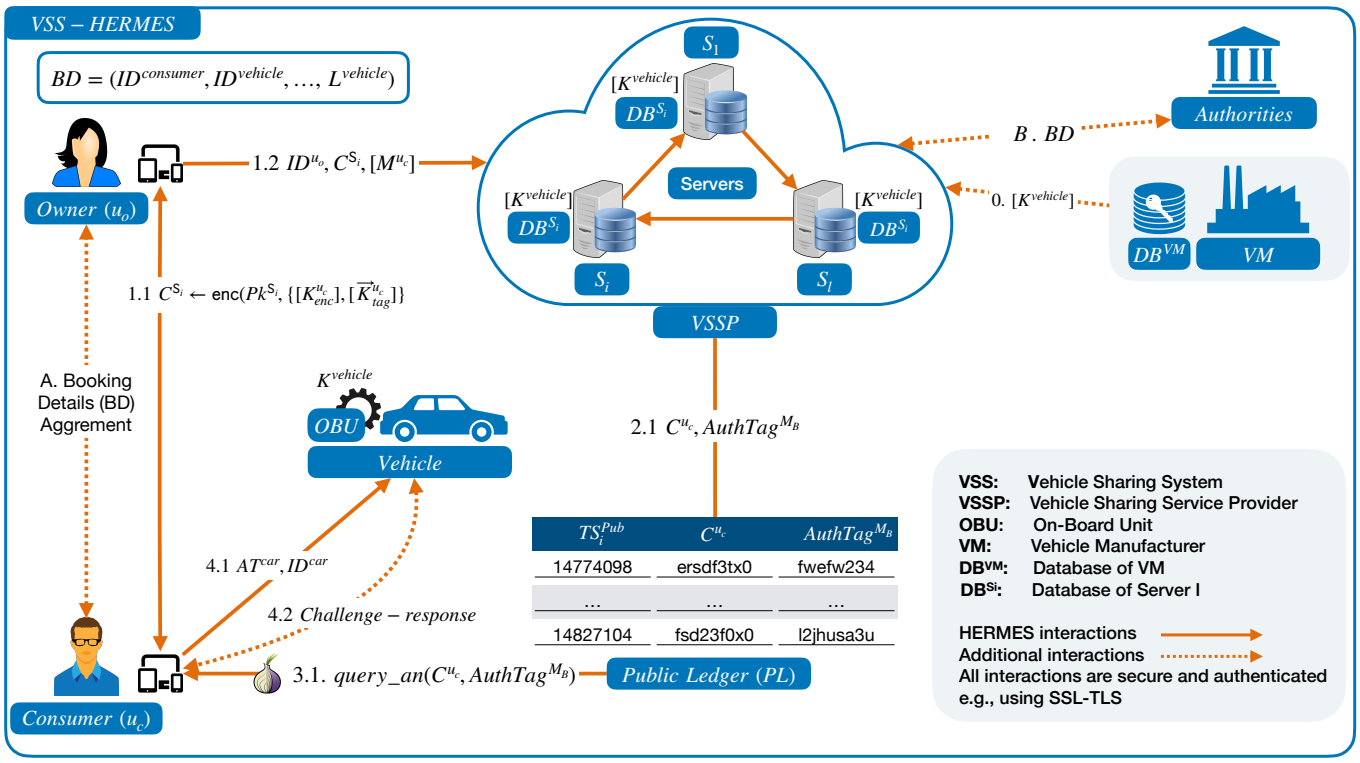

Fig. 2: HERMES high level overview. Numbers correspond to the steps outlined in the text of Section IV. Figures 3, 4, 5 and 6 describe steps 1, 2, 3 and 4 in more detail.

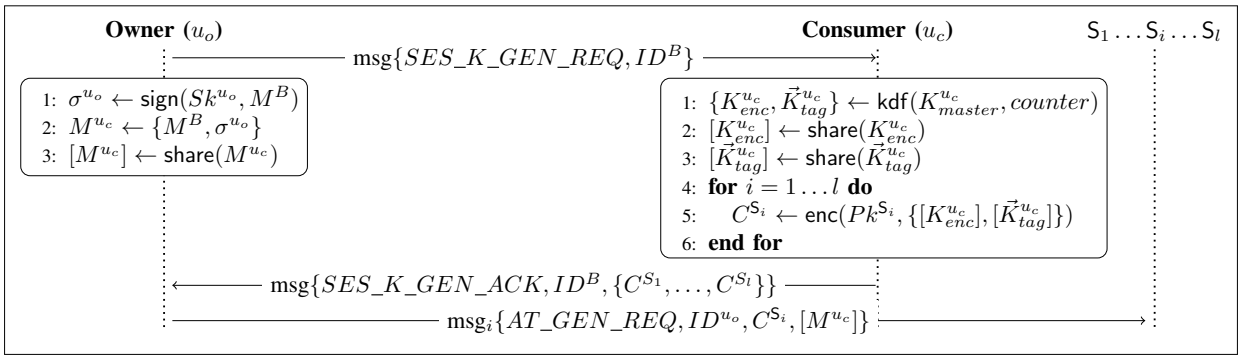

Fig. 3: Step 1: session keys generation and data distribution.

allow u_c to verify that the AT contains M^B which was agreed upon, during the *vehicle booking*. It utilizes a kdf() function instantiated by u_c's master key, i.e., $K^{u_c}_{master}$ and a *counter*. For $\vec{K}^{u_c}_{tag}$, two session keys are generated and stored: one for encryption, $K^{u_c}_{tag_{enc}}$ (i.e., $\vec{K}^{u_c}_{tag}[0] = K^{u_c}_{tag_{enc}}$), and one for authentication, $K^{u_c}_{tag_{mac}}$ (i.e., $\vec{K}^{u_c}_{tag}[1] = K^{u_c}_{tag_{mac}}$). Then, u_c transforms these keys into ℓ secret shares, $[K^{u_c}_{enc}]$ and $[\vec{K}^{u_c}_{tag}]$, 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 the shares of these keys securely.

The consumer encrypts $[K^{u_c}_{enc}]$ and $[\vec{K}^{u_c}_{tag}]$ with the public-key of each S_i, $C^{\mathsf{S}_i} = \mathsf{enc}(Pk^{\mathsf{S}_i}, \{[K^{u_c}_{enc}], [\vec{K}^{u_c}_{tag}]\})$, such that only the specific 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 $\{C^{\mathsf{S}_1}, \ldots, C^{\mathsf{S}_l}\}$. 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 vehicle 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 $[M^{u_c}]$.

3) Step 2 – Access Token Generation: The servers generate an AT and publish it on the Public Ledger (PL).

In detail, as shown in Fig. 4, upon receipt of *AT_GEN_REQ* from u_o, the servers obtain the session keys in shared form, $\{[K^{u_c}_{enc}], [\vec{K}^{u_c}_{tag}]\}$. Each S_i decrypts C^{S_i} using its private key. Session keys are for encrypting an AT used to access a vehicle by u_c and for generating an authentication tag used by u_c to verify the data authenticity of the booking details contained in the AT, respectively. To generate the AT, i.e., $[AT^{vehicle}]$,

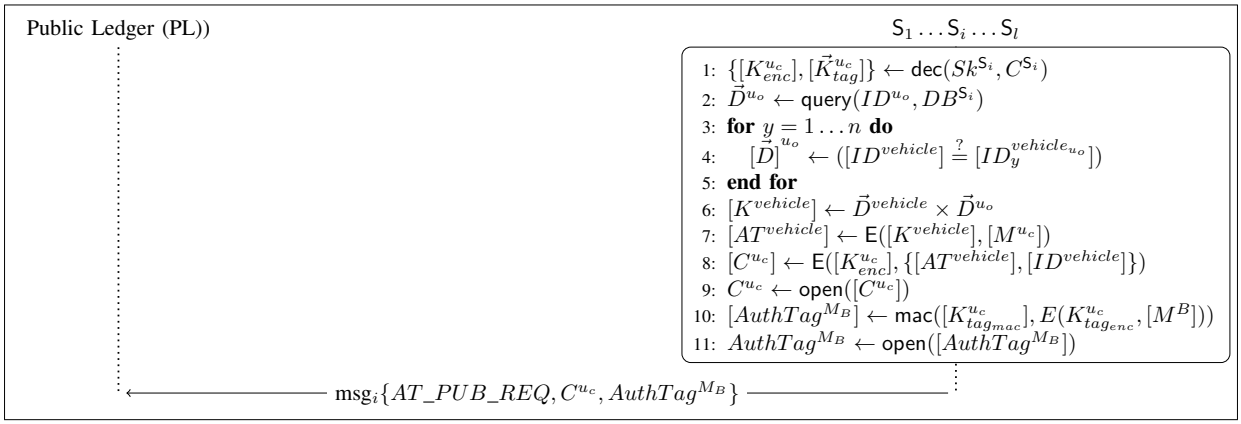

Fig. 4: Step 2: AT generation.

the key of the vehicle, i.e., $[K^{vehicle}]$, is retrieved from DB^{S_i} using query and equality check operations as proposed in [22].

The VSSP servers S_i then collaboratively encrypt $[M^{u_c}]$ using the retrieved $[K^{vehicle}]$ to generate an AT for the vehicle in shared form, $[AT^{vehicle}]$. As $AT^{vehicle_o}$ and $ID^{vehicle_o}$ need to be available only to u_c, a second layer of encryption is performed using $K^{u_c}_{enc}$. Then, the servers encrypt $[AT^{vehicle}]$ and $[ID^{vehicle}]$ with $[K^{u_c}_{enc}]$ to generate and retrieve C^{u_c} using $\mathsf{open}([C^{u_c}])$, i.e.,

$$\begin{aligned}[AT^{vehicle}] &\leftarrow \mathsf{E}([K^{vehicle}], [M^{u_c}]),\\ [C^{u_c}] &\leftarrow \mathsf{E}([K^{u_c}_{enc}], \{[AT^{vehicle}], [ID^{vehicle}]\}),\\ C^{u_c} &\leftarrow \mathsf{open}([C^{u_c}]).\end{aligned}$$

In addition, each S_i generates an authentication tag, i.e., $[AuthTag^{M_B}]$, that can be later used to retrieve the associated $AT^{vehicle_o}$ from the PL by u_c. Using $\mathsf{mac}()$ with $[\vec{K}^{u_c}_{tag}]$ and $[M^B]$ as inputs, each S_i creates an authentication tag $[AuthTag^{M_B}]$. Prior to posting on the PL, we use $\mathsf{open}([AuthTag^{M_B}])$ to obtain $AuthTag^{M_B}$, i.e.,

$$\begin{aligned}[AuthTag^{M_B}] &\leftarrow \mathsf{mac}([K^{u_c}_{tag_{mac}}], E(K^{u_c}_{tag_{enc}}, [M^B])),\\ AuthTag^{M_B} &\leftarrow \mathsf{open}([AuthTag^{M_B}]),\end{aligned}$$

where we recall that $\vec{K}^{u_c}_{tag} = (K^{u_c}_{tag_{mac}}, K^{u_c}_{tag_{enc}})$. Here, we note that for the efficiency over MPC, we perform Enc-then-Hash-then-MAC. The reason is that, following [44] encryption, i.e., E, can be done in parallel and separately (so efficient), and then the hash does not need to be done in MPC and the MPC parties can apply the hash function locally. Essentially, we trade "parallel MPC encryption" for "having to evaluate a hash function on large input in MPC".[4]

Finally, each S_i sends to PL an access-token-publication request, *AT_PUB_REQ*, along with C^{u_c} and $AuthTag^{M_B}$.

4) Step 3 – Access Token Distribution and Verification: The PL publishes the encrypted AT which is then retrieved by u_c. Once retrieved, u_c can obtain the AT and use it to access the vehicle.

In detail, as shown in Fig. 5, upon receipt of *AT_PUB_REQ*, PL publishes C^{u_c}, $AuthTag^{M_B}$ and TS^{Pub}, which is the time-stamp of the publication of the encrypted token. User u_c monitors PL for concurrent and following TS^{Pub} to identify the corresponding C^{u_c} using $AuthTag^{M_B}$. Upon identification, C^{u_c} queries and anonymously retrieves C^{u_c} from PL using $\mathsf{query\_an}()$, such that PL cannot identify u_c. Then, u_c decrypts C^{u_c} using $K^{u_c}_{enc}$ to obtain the AT and the vehicle identity, $\{AT^{vehicle}, ID^{vehicle}\}$.

Note that, in a parallel manner and for synchronization purposes, PL sends an acknowledgment of the publication, *AT_PUB_ACK*, along with TS^{Pub}_i to at least one S_i which it forwards it to u_o who, in turn, forwards it to u_c. Upon receipt of *AT_PUB_ACK*, u_c uses TS^{Pub}_i to query PL. In the same manner, it uses $\mathsf{query\_an}()$ to anonymously retrieve C^{u_c} and $AuthTag^{M_B}$. u_c verifies the authentication tag C^B locally using the $\mathsf{mac}()$ function with $K^{u_c}_{tag_{mac}}$ 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 *vehicle booking*. Then, u_c decrypts C^{u_c} using $K^{u_c}_{tag_{enc}}$ to obtain the access token and the vehicle identity, $\{AT^{vehicle}, ID^{vehicle}\}$.

5) Step 4 – Car Access: The consumer uses the AT $AT^{vehicle_o}$, and $ID^{vehicle}, Cert^{u_c}$, to obtain access to the vehicle using any challenge-response protocol based on public key implementations [47], [16] (see Fig. 6). A detailed representation of the last step can be found at the online full version of HERMES [46] and in [22].

Functional requirements realization:

a) FR1 – Offline access provision: Note that steps 1-3 of the protocol require a network connection, but step 4, vehicle access, is performed using close-range communication and with no need of a network connection. The decryption and verification of the AT can be performed by the vehicle offline (it has its key $K^{vehicle_o}$ and the owner's public-key Pk^{u_o} stored). Sending the confirmation signature $\sigma^{vehicle}_{Access}$ can also be done offline.

b) FR2 – Access token update and revocation by u_o: Upon an agreement between u_o and u_c to update or revoke an AT, HERMES can perform as it is described in steps 1-3. An update request can alter the booking values according to a new booking request, $\hat{M}^B$. For revocation, each of the parameters in $\hat{M}^B$ can receive a predefined value indicating the revocation action. There are occasions when u_o may need

[4]In our implementation, we use CBC-MAC-AES and HtMAC-MiMC.

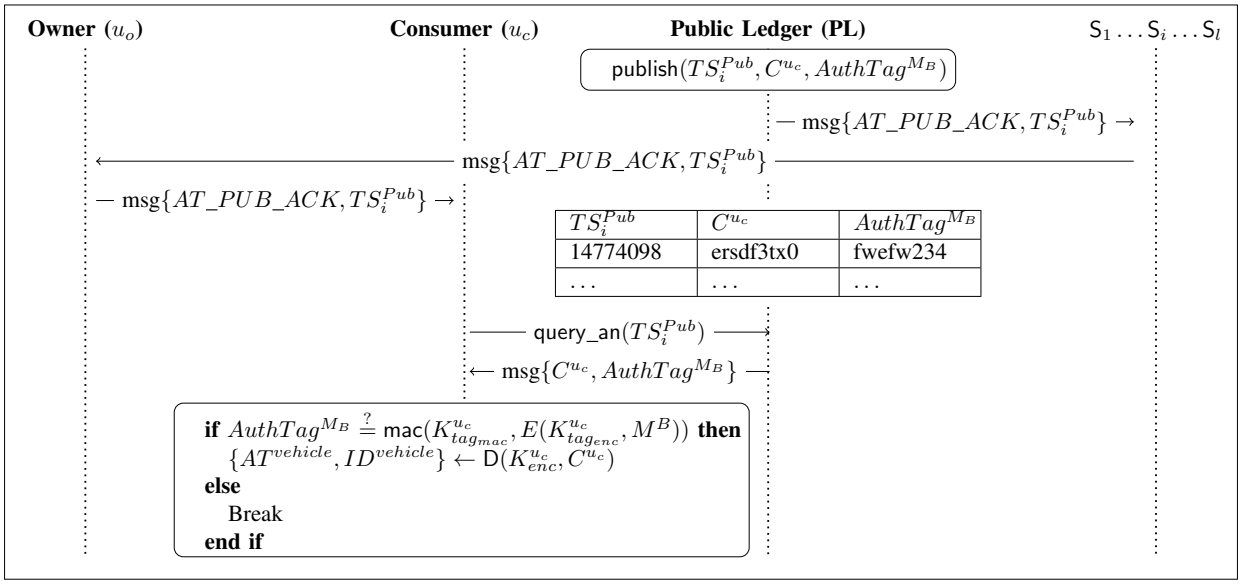

Fig. 5: Step 3: Access token distribution and verification.

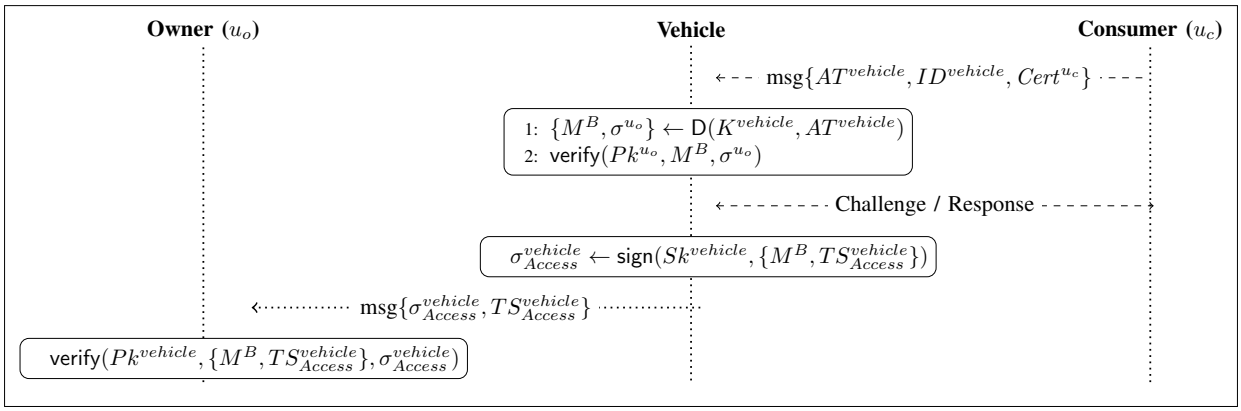

Fig. 6: Step 4: vehicle access. Dashed lines represent close range wireless communication.

to enforce an update or revocation of an AT. For u_c to prevent blocking such requests and operations, HERMES should be executed only by u_o, without the involvement of u_c. More specifically, u_o generates session keys, requests an AT, queries the PL, and sends the token to the vehicle using a long-range communication channel such as LTE.

V. Security and Privacy Analysis

We prove that HERMES satisfies the security and privacy requirements of Sect. II, provided that its underlying cryptographic primitives are sufficiently secure. The theorem statement and the proof given below are informal, in a similar sense by [22]. A complete formal description of the semantic security models and the stand-alone proofs are given in the online version of the system [46].

Theorem 1. *Assume that communication takes place over private channels. If:*

- *the MPC is statistically secure [44],*
- *the signature scheme* sign *is multi-key existentially unforgeable [48],*
- *the key derivation function* kdf *is multi-key secure [49],*
- *the public-key encryption scheme* enc *is multi-key semantically secure [50],*
- *the symmetric key encryption scheme* E *is multi-key chosen-plaintext secure [51],*
- *the hash function* hash *is collision resistant [52], and*
- *the MAC function* mac *is multi-key existentially unforgeable [48],*

then HERMES fulfills the security and privacy requirements of Sect. II.

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 VSSP, and E and mac are used for independent keys (i.e., session keys) for every fresh evaluation of the protocol. We refer to Bellare et al. [50] 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 the consumer and owner have agreed upon the booking details prior to evaluating HERMES. Hence they know each other.

a) SR1 – Confidentiality of M^B: In one evaluation of the protocol, u_c, u_o, and the shared vehicle learn the booking details by default or design. The VSSP 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 $AuthTag^{M_B}$ and C^{u_c} satisfying

$$AuthTag^{M_B} = \mathsf{mac}(K^{u_c}_{tag_{mac}}, \mathsf{E}(K^{u_c}_{tag_{enc}}, M^B)), \quad (1)$$

$$C^{u_c} = \mathsf{E}(K^{u_c}_{enc}, \{\mathsf{E}(K_y^{vehicle_{u_o}}, \{M^B, \sigma^{u_o}\}), ID^{vehicle_o}\}), \quad (2)$$

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 $K^{u_c}_{enc}$ and $\vec{K}^{u_c}_{tag} = (K^{u_c}_{tag_{enc}}, K^{u_c}_{tag_{mac}})$. The nested encryption E does not influence the analysis due to the mutual independence of the keys $K^{u_c}_{enc}$ and $K_y^{vehicle_{u_o}}$.

b) SR2 – Authenticity of M^B: An owner who initiates the AT generation and distribution, first signs the booking details using its private key before sending those to the VSSP in shares. Therefore, once the vehicle receives the token and obtains the booking details, it can verify the owner's signature on the booking details. In other words, the vehicle 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 vehicle 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 the 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.

c) SR3 – Confidentiality of $AT^{vehicle_o}$: The AT is generated by the VSSP servers obliviously (as the MPC is statistically secure), and only revealed to the public in encrypted form, through C^{u_c} of (7). Due to the uniform drawing of the key $K^{u_c}_{enc}$ (and the security of the public-key encryption scheme used to transmit this key), only the legitimate user can decrypt and learn the AT. It shares it with the vehicle over a secure and private channel.

d) SR4 – Confidentiality of $K^{vehicle_o}$: Only the vehicle manufacturer and the vehicle itself hold copies of the vehicle key. Considering the VM as a Trusted Provider (TP), it holds all the secret keys of vehicles that produce. The VSSP servers learn these in shared form, hence learn nothing about it by virtue of the statistical security of the MPC. Retrieving a vehicle key from encryptions made under this key constitutes a key recovery attack, which in turn allows breaking the chosen-plaintext security of the symmetric key encryption scheme.

e) SR5 – Backward and forward secrecy of $AT^{vehicle_o}$: The AT is published on the Public Ledger (PL) as C^{u_c} of (7), encrypted under symmetric key $K^{u_c}_{enc}$. Every honest consumer generates a fresh key $K^{u_c}_{enc}$ for every new evaluation. It uses a key derivation function kdf utilizing a PRF for each key generation and every new evaluation of the protocol, and that is secure. 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 ATs for malicious users who may deviate from the protocol and reuse one-time keys.

f) SR6 – Non-repudiation of origin of $AT^{vehicle_o}$: The vehicle, which is a trusted entity, 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, i.e., $AuthTag^{M_B} \stackrel{?}{=} \mathsf{mac}(K^{u_c}_{tag_{mac}}, E(K^{u_c}_{tag_{enc}}, M^B))$. Note that u_c does not effectively verify $AT^{vehicle_o}$, but rather $AuthTag^{M_B}$, 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.

g) SR7 – Non-repudiation of delivery of $AT^{vehicle_o}$: The owner can verify the correct delivery of $AT^{vehicle_o}$ with the successful verification and message sent by the vehicle to the owner, $\mathsf{verify}(Pk^{vehicle_o}, \{M^B, TS^{vehicle_o}_{Access}\}, \sigma^{vehicle_o}_{Access})$ at the end of the protocol. Security breaks only if the signature scheme is forgeable.

h) SR8 – Accountability of users (i.e., owners, and consumers): In the case of disputes, the information related to a specific transaction (and only this information) may need to be reconstructed. Reconstruction can be done only if the VSSP servers collude and reveal their shares. In our setting, these servers have competing interests. Thus they would not collude unless law authorities' request. Due to threshold secret sharing properties, the private inputs can be reconstructed by a majority coalition. This is, if the VSSP consists of three parties, it suffices two of party-shares required to reconstruct the secret.

i) PR1 – Unlinkability of any two requests of u_c for any $vehicle_o$(s): The only consumer-identifiable data is in the consumer's certificate included in the booking details. Note that these are agreed upon and between the consumer and the owner, so the owner learns the consumer's identity by default. Beyond that, the consumer only communicates with the vehicle, which is supposed to learn the consumer's identity to perform proper access verification. The consumer consults the PL over an anonymous communication channel [34]. The booking details are transferred to and from the VSSP, but these are encrypted and do not leak by virtue of their confidentiality (security requirement SR1).

j) PR2 – Anonymity of u_c and the $vehicle_o$: The reasoning is identical to that of PR1.

k) PR3 – Indistinguishability of $AT^{vehicle_o}$ operations: Access token generation, update, or revocation is performed using the same steps and the same type of messages sent to

the VSSP and PL. Hence, outsiders and system entities cannot distinguish which operation has been requested. □

VI. Performance Evaluation

We take a different approach to [22] as we implement our protocols in a fully-fledged open-sourced MPC framework, i.e., MP-SPDZ [53]. The framework supports more than 30 MPC protocols carefully implemented in C++. In addition, a Python front-end compiler allows the expression of circuits in a relatively simple way. For MP-SPDZ, the compiler reduces the high-level program description to bytecode or set of instructions for which the parties then run an optimized virtual machine written in C++ to execute the protocols. In our case, two versions of HERMES were benchmarked: one with CBC-MAC tailored for binary circuits, while the other one uses HtMAC, which is tailored for arithmetic circuits. For our benchmark we use the following settings: $||M^{u_c}|| = ||AT^{vehicle}|| = 10 \cdot 128$ bits, whereas $ID^{vehicle} \leq 2^{32}$, which thus fits into one 128 bit-string, and $||M^B|| = 6 \cdot 128$ bits.

Environment Settings: We benchmarked our protocols of VSSP servers using three distinct computers connected on a LAN network equipped with Intel i7-7700K CPU and 32GB of RAM over a 10Gb/s network switch and 0.5ms Round Trip Time (RTT).[5]

1) Theoretical Complexity: Measuring the complexity of an MPC protocol usually boils down to counting the number of non-linear operations in the circuit and the circuit depth. We consider the case where a protocol is split into two phases. An input-independent (preprocessing) phase, where the goal is to produce correlated randomness. Additionally, an input-dependent (online) phase where parties provide their inputs and start exchanging data using the correlated randomness produced beforehand. One secret multiplication (or an AND gate for the $\mathbb{F}_2$ case) requires one random Beaver triple (correlated randomness) [54] from the preprocessing phase and two $\mathsf{open}()$ operations in the online phase.

Note that, in our case, the two versions of HERMES are benchmarked using the following two executables: *replicated-bin-party.x* ($\mathbb{F}_2$ case, CBC-MAC) and *replicated-field-party.x* ($\mathbb{F}_p$ case, HtMAC). The first executable is the implementation of Araki et al.'s binary based protocol [39], while the latter is for the field case. Next, we analyze the complexity of these two separately and motivate the two choices.

2) CBC-MAC-AES – case for binary circuits: This solution is implemented to have a baseline comparison with SePCAR [22] using MP-SPDZ [53]. The equality check is implemented using a binary tree of AND operations with a $\log n$ depth where n is the number of vehicles (see Fig. 4). Obtaining the corresponding vehicle key $[K^{vehicle}]$ assuming there are n vehicles per user has a cost of $159 \cdot n$ Beaver triples assuming 32 bit length vehicle IDs.

When evaluating the operations depicted in Fig. 4 in MPC, the most expensive part is computing $[AT^{vehicle}]$ since that requires encrypting $10 \cdot 128$ bits, calling AES 10 times, which has a cost of $6400 \cdot 10$ AND gates. In the next step, (in line 8 in Fig. 4), AES is called 11 times, while the operation computing CBC-MAC-AES (from line 10) takes only 6 AES calls. Given the above breakdown, the theoretical cost for generating an access token has a cost of $159 \cdot n + 6400 \cdot 28$ AND gates.

3) HtMAC-MiMC – case for arithmetic circuits: Recent results of Rotaru et al. [44] showed that, when considering MPC over arithmetic circuits, efficient modes of operation over encrypted data are possible if the underlying PRF is MPC-friendly. We integrate their approach into SePCAR, and results from Table II show that it is at least 16 times faster than using MPC over binary circuits with CBC-MAC-AES. This might come as a surprise since comparisons are more expensive to do in arithmetic circuits and although they weigh a good chunk of the overall cost. Recent improvements using edaBits [45] made comparisons much faster, which, in turn, improved the MPC protocols used by HERMES. To summarize, we breakdown the cost into the following:

- 10 calls to MiMC to encrypt M^{u_c} (excluding one call for computing the tweak according to [44]),
- 11 more calls to compute C^{u_c} - encrypting the concatenation of $AT^{vehicle_o}$ and $ID^{vehicle_o}$. Note that since we are using a different key than the first step we need to compute another tweak (one extra PRF call),
- 6 calls to compute $AuthTag^{M_B}$, one more PRF call for computing the tweak $N = E_{K^{u_c}_{tag}[0]}(1)$ and a final PRF call $E_{K^{u_c}_{tag}[1]}(\mathsf{hash}'(ct))$ where ct are the opened ciphertexts from encrypting M_B and $\mathsf{hash}'(\cdot))$ is a truncated version of a SHA-3 where we keep the first 128 bits hash [44].

If we include the PRF calls to compute the tweaks, there are 31 calls to a PRF, so one can think that the Boolean case is more efficient than the arithmetic case. In practice, we see that HtMAC construction is faster than CBC-MAC-AES, albeit with a factor of two communication overhead (see Table II). One reason for this is that HtMAC is fully parallelizable, resulting in an MPC protocol with fewer rounds than CBC-MAC-AES.

One of the main benefits of HtMAC construction is that it can be instantiated with the Legendre PRF, which can make the number of communication even lower. We chose the MiMC based PRF as that is demonstrated to be faster on a LAN [44] and to have a lower communication overhead – although a higher number of communication rounds than the Legendre based PRF.

4) Benchmark results: We vary the number of vehicle IDs (i.e., the numbers of vehicles registered per owner) and compute the communication rounds, data sent between the VSSP servers, and the total throughput meaning the total number of ATs generated per second from Fig. 4. In Table II, we report the performance for a low number of vehicle IDs (i.e., $1, 2, 4$), representing individuals, but also for a large number (i.e., $256, 512, 1024$), representing large branches of vehicle-rental companies.

[5]The implementation can be obtained from: https://github.com/rdragos/MP-SPDZ/tree/hermes

TABLE II: HERMES performance, i.e., access token generation, per number of vehicles utilizing: CBC-MAC-AES and HtMAC-MiMC. Throughput is evaluated for all servers and communication cost per server.

Type of Vehicle Owners	Protocol	Number of Vehicles per Owner	Communication Rounds	Communication Data (kB)	Throughput (ops/s)
Individuals	CBC-MAC-AES	1	568	64	33
	HtMAC-MiMC	1	167	108	546
	CBC-MAC-AES	2	568	64	32
	HtMAC-MiMC	2	167	108	546
	CBC-MAC-AES	4	568	107.7	32
	HtMAC-MiMC	4	167	117	544
Vehicle-rental company branches	CBC-MAC-AES	256	568	76	32
	HtMAC-MiMC	256	167	150	260
	CBC-MAC-AES	512	568	88	32
	HtMAC-MiMC	512	167	194	151
	CBC-MAC-AES	1024	568	112	32
	HtMAC-MiMC	1024	167	280	84

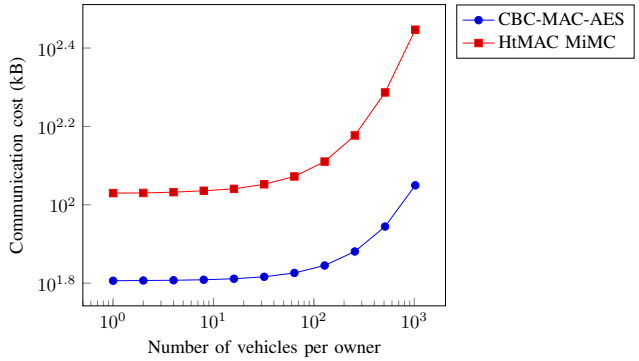

Fig. 7: Communication cost per server: access token generation cost (i.e., data sent-received) for a number of vehicles per owner.

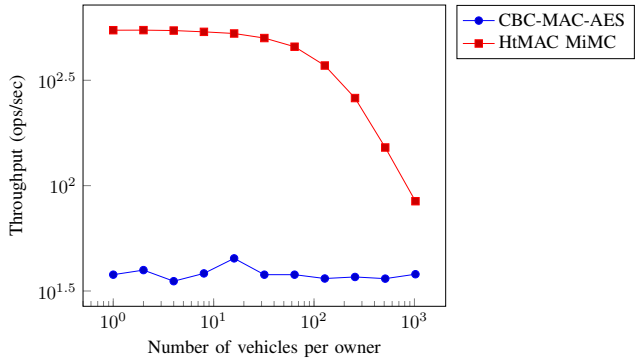

Fig. 8: Throughput for all servers: access token generation (i.e., ops) per second for a number of vehicles per owner.

We can see that the throughput of the AT generation when instantiated using CBC-MAC-AES remains constant, whereas, for HtMAC-MiMC, it is decreasing. The reason for this is that when scaling the number of vehicles, the number of comparisons is increasing as well. For the case of arithmetic circuits, the comparisons become costly operations, whereas, for Boolean circuits, comparisons can be made efficiently. However, the throughput for HtMAC-MiMC is always better than CBC-MAC-AES, and this is because MiMC-based PRF is more lightweight – requiring fewer multiplications – and has a smaller circuit depth. We report the full range of experiments for varying the number of vehicle IDs regarding the communication sent between the parties in Fig. 7 as well the throughput in Fig. 8.

5) Comparison with SePCAR [22]: SePCAR reports 1.2 seconds for generating the access token, see Table 2 in [22]. Note that SePCAR uses a single-vehicle ID when reporting [22]. When benchmarked on similar hardware we get a throughput of 33 access tokens per second, which demonstrates the benefits of integrating our solution in a fully-fledged MPC framework such as MP-SPDZ [53]. This makes HERMES with the CBC-MAC-AES construction roughly 40 times faster than SePCAR. Switching from CBC-MAC-AES to HtMAC offers a throughput of 546 ATs per second, which makes it $\approx$ 16.5 times better than CBC-MAC, making it around 600 times faster than original timings in SePCAR [22]. We stress that our implementation of SePCAR was faster due to writing CBC-MAC-AES using a mature MPC framework such as MP-SPDZ rather than using custom code as in [22].

6) Total time: SePCAR reports on Step 1 and Step 3 taking 0.22 seconds and 0.05 seconds, respectively, were the major bottleneck produces the RSA signatures (50ms per signing operation) [22]. However, when benchmarking the OpenSSL [55] RSA signatures on our computer, one RSA signature takes 0.5ms. Plugging in the RSA signature time to the estimations done in SePCAR, Step 1 should take $\approx$ 4ms. Step 3 could probably be improved by storing the data in memory instead of fetching it from a database. Using our RSA signature time, Step 1 and Step 3 from SePCAR should take $\approx$ 59ms, while Step 2 of our protocol takes 1.83ms in the batched setting (see Table II) as HtMAC-MiMC has a throughout of 546 ops/s. This might look surprising, but in our solution, the non-MPC part is the bottleneck, making the approximated total time of 61ms being around 25 times faster than [22].

VII. Related work

Our work is closely related to the SePCAR protocol proposed by [22]. They initially performed security and privacy analysis on vehicular-sharing systems [9], and designed a solution for privacy-preserving access provision protocol [22]. Their solution design considers untrusted servers for the generation and distribution of ATs. They utilize MPC in combination with several cryptographic primitives. With their work, they also consider malicious users, and SePCAR considers accountability revealing a user's identity in case of wrongdoings. However, the protocol is not tested on how it scales to multiple evaluations of equality checks – essentially to vehicular-fleet and multi-vehicle owners. Moreover, in terms of efficiency, HERMES is proven to run significantly faster than SePCAR due to optimized MPC constructions, making it efficient enough to support a fleet of vehicles.

State of the art on vehicle-sharing also focuses on either a trusted service provider, on complementary operations to access provision, or fulfills a subset of properties for a solution design. Initially, Dmitrienko and Plappert [16] were the first to design a secure free-floating vehicle sharing system. In contrast to HERMES, their protocol assumes a fully trusted vehicle sharing provider who has access to the master key of smart-vehicles and collects and stores all the information exchanged between the vehicle provider and their users for every vehicle access provision. Huang et al. [56] proposed a privacy-preserving identity management protocol focusing on authentication while verifying users' misbehavior. They utilize decentralized entities and a centralized vehicle sharing service provider. However, the service provider is trusted and can know who is sharing, which vehicle, with whom raising questions regarding their adversarial model. Madhusudan et al. [57] and De Troch [58] proposed privacy-preserving protocols for booking and payments operations on vehicle sharing systems. Their protocols utilize smart contracts on the Ethereum blockchain. Trust is placed on cryptographic primitives and blockchain instead of a centralized party. De Troch [58] also considers accountability in case of misbehavior, which malicious behavior is tracked and punished by the loss of privacy and deposit.

Considering secure vehicular communications, Papadimitratos [59] analyzed the security challenges in vehicular communication systems while Jin et al. [60] extend the analysis to the privacy landscape. Raya et al. [61] described the need for a vehicular PKI to secure vehicular communications, and Khodaei et al. [62] proposed a generic pseudonymization approach to preserve the unlinkability of messages exchanged between vehicles and PKI servers. Driven by [59], PRESERVE [27] and EVITA [29] were designated projects on the design and specification of the secure architecture of OBUs. Secure access controls to the OBU from smartphones were proposed by Groza et al. [63] to use smartphones as a vehicle key. Huayi et al. [64] proposed an enhanced scheme of [65], namely DUBI, a decentralized and privacy-preserving usage-based insurance scheme built on the blockchain technology to address privacy concerns for pay-as-you-drive insurances using zero-knowledge proofs and smart contracts.

VIII. Conclusion

In this paper, we proposed HERMES – an efficient, scalable, secure, and privacy-enhancing system for vehicle access provision. It allows users to dynamically instantiate, share, and access vehicles in a secure and privacy-enhancing fashion. To achieve its security and privacy-enhancing properties, HERMES deploys MPC-friendly primitives for access token generation and sharing while keeping the booking details confidential not only from malicious outsiders but also from untrusted servers. To ensure efficiency and scalability, HERMES uses two modes of operation, supporting various users and vehicles per users. This makes HERMES suitable for individuals with few vehicles and rental companies with potentially thousands of vehicles per branch. We presented a formal analysis of our system's security and privacy requirements and designed a prototype as a proof-of-concept. We benchmarked the MPC evaluations with HtMAC-MiMC for the case of arithmetic circuits and CBC-MAC-AES for the case of binary circuits. HERMES demonstrates a significant increase in performance of ≈ 61ms for a vehicle access provision, thus demonstrating its efficiency to SePCAR (i.e., 25 times faster).

In the future, we plan to extend HERMES to support additional operations such as booking and payment as well as to protect against MPC servers as active adversaries. Moreover, we aim to investigate whether CBC-MAC construction can be more efficient with more lightweight block ciphers such as Rasta.

References

[1] ACEA, "Carsharing: Evolution, Challenges and Opportunities," https://goo.gl/NTec4l, accessed April, 2017.

[2] J. Bert, B. Collie, M. Gerrits, and G. Xu, "What's ahead for car sharing?: The new mobility and its impact on vehicle sales," https://goo.gl/ZmPZ5t, accessed June, 2017.

[3] Volvo, "Worth a Detour," https://www.sunfleet.com/, accessed Nov., 2016.

[4] BMW, "DriveNow Car Sharing," https://drive-now.com/, accessed Nov., 2016.

[5] USA TODAY, "Toyota will test keyless car sharing," https://goo.gl/C9iq34, accessed Nov., 2016.

[6] United States Patent and Trademark Office. Applicant: Apple Inc, "Accessing a vehicle using portable devices." https://goo.gl/a9pyX7, accessed June, 2017.

[7] 9to5Mac. New "CarKey" feature in iOS 13.4 beta brings built-in support for unlocking, driving, and sharing NFC car keys. Accessed Feb., 2020. [Online]. Available: shorturl.at/ryAR8

[8] United States Patent and Trademark Office. Applicant: Apple Inc. Accessing a vehicle using portable devices. https://goo.gl/a9pyX7. Accessed Sept, 2017.

[9] I. Symeonidis, M. A. Mustafa, and B. Preneel, "Keyless car sharing system: A security and privacy analysis," in *IEEE ISC2*, 2016, pp. 1–7.

[10] A. Millard-Ball, *Car-sharing: Where and how it succeeds*. Transportation Research Board, 2005, vol. 60.

[11] S. Le Vine, A. Zolfaghari, and J. Polak, "Carsharing: evolution, challenges and opportunities," *22th ACEA Scientific Advisory Group Report, 20ff https://www. acea. be/uploads/publications/SAG_Report_-_Car_Sharing. pdf*, 2014.

[12] F. Ferrero, G. Perboli, M. Rosano, and A. Vesco, "Car-sharing services: An annotated review," *Sustainable Cities and Society*, vol. 37, pp. 501 – 518, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S221067071630573X

[13] E. W. Martin and S. A. Shaheen, "Greenhouse gas emission impacts of carsharing in north america," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1074–1086, 2011. [Online]. Available: https://doi.org/10.1109/TITS.2011.2158539

[14] S. A. Shaheen and A. P. Cohen, "Car sharing and personal vehicle services: worldwide market developments and emerging trends," *Int. Journal of Sustainable Transportation*, vol. 7, no. 1, pp. 5–34, 2013.

[15] M. R. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, "Smarter cities and their innovation challenges," *IEEE Computer*, vol. 44, no. 6, pp. 32–39, 2011.

[16] A. Dmitrienko and C. Plappert, "Secure free-floating car sharing for offline cars," in *ACM CODASPY*, 2017, pp. 349–360.

[17] The Guardian. My smart car rental was a breeze – until i got trapped in the woods. https://www.theguardian.com/technology/2020/feb/18/smart-car-gig-rental-app-trapped?CMP=Share_AndroidApp_WhatsApp. Accessed December, 2020.

[18] GOV.UK, "Reducing mobile phone theft and improving security," https://goo.gl/o2v99g, accessed April, 2017.

[19] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting," *PoPETs*, vol. 2016, no. 1, pp. 34–50, 2016.

[20] reddit, "Identifying Muslim cabbies from trip data and prayer times," https://goo.gl/vLrW1s, accessed April, 2017.

[21] Council of the EU Final Compromised Resolution, "General Data Protection Regulation," http://www.europarl.europa.eu, accessed Feb., 2015.

[22] I. Symeonidis, A. Aly, M. A. Mustafa, B. Mennink, S. Dhooghe, and B. Preneel, "Sepcar: A secure and privacy-enhancing protocol for car access provision," in *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, ser. Lecture Notes in Computer Science, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds., vol. 10493. Springer, 2017, pp. 475–493. [Online]. Available: https://doi.org/10.1007/978-3-319-66399-9_26

[23] K. Münzel, W. Boon, K. Frenken, J. Blomme, and D. van der Linden, "Explaining carsharing supply across western european cities," *International Journal of Sustainable Transportation*, vol. 14, no. 4, pp. 243–254, 2020.

[24] P. Papadimitratos, A. de La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Commun. Mag.*, vol. 47, no. 11, pp. 84–95, 2009. [Online]. Available: https://doi.org/10.1109/MCOM.2009.5307471

[25] INVERS, "Make Mobility Shareable," https://invers.com/, accessed April, 2017.

[26] S. Micali, "Algorand: The efficient and democratic ledger," *arXiv:1607.01341*, 2016.

[27] PRESERVE, "Preparing Secure Vehicle-to-X Communication Systems (PRESERVE)," https://www.preserve-project.eu/, accessed Nov., 2016.

[28] Trusted Computing Group, "TPM 2.0 Library Profile for Automotive-Thin," https://goo.gl/fy3DxD, accessed June, 2016.

[29] EVITA, "E-safety Vehicle Intrusion Protected Applications (EVITA)," http://www.evita-project.org/, accessed Nov., 2016.

[30] Internet Engineering Task Force, "PKCS #1: RSA Cryptography Specifications Version 2.0." https://tools.ietf.org/html/rfc2437, accessed June, 2017.

[31] E. Barker, J. Kelsey *et al.*, "Nist special publication 800-90a: Recommendation for random number generation using deterministic random bit generators," 2012.

[32] S. Cohney, A. Kwong, S. Paz, D. Genkin, N. Heninger, E. Ronen, and Y. Yarom, "Pseudorandom black swans: Cache attacks on ctr_drbg," in *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*. IEEE, 2020, pp. 1241–1258. [Online]. Available: https://doi.org/10.1109/SP40000.2020.00046

[33] 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.

[34] Tor Project, "Protect your privacy. Defend yourself against network surveillance and traffic analysis." https://www.torproject.org/, accessed April, 2017.

[35] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167.

[36] S. Micali, O. Goldreich, and A. Wigderson, "How to play any mental game," in *Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC*, 1987, pp. 218–229.

[37] M. Hastings, B. Hemenway, D. Noble, and S. Zdancewic, "Sok: General purpose compilers for secure multi-party computation," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1220–1237.

[38] Y. Lindell, "Secure multiparty computation (mpc)," Cryptology ePrint Archive, Report 2020/300, 2020, https://eprint.iacr.org/2020/300.

[39] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proceedings of the 2016 ACM SIGSAC CCS*, 2016, pp. 805–817.

[40] K. Chida, D. Genkin, K. Hamada, D. Ikarashi, R. Kikuchi, Y. Lindell, and A. Nof, "Fast large-scale honest-majority mpc for malicious adversaries," in *Annual International Cryptology Conference*. Springer, 2018, pp. 34–64.

[41] I. Damgård, K. Damgård, K. Nielsen, P. S. Nordholt, and T. Toft, "Confidential benchmarking based on multiparty computation," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 169–187.

[42] I. Damgård and J. B. Nielsen, "Universally composable efficient multiparty computation from threshold homomorphic encryption," in *Annual International Cryptology Conference*. Springer, 2003, pp. 247–264.

[43] A. Aly, K. Cong, D. Cozzo, M. Keller, E. Orsini, D. Rotaru, O. Scherer, P. Scholl, N. Smart, T. Tanguy *et al.*, "Scale–mamba v1. 10: Documentation," 2020.

[44] D. Rotaru, N. P. Smart, and M. Stam, "Modes of operation suitable for computing on encrypted data," *IACR Transactions on Symmetric Cryptology*, pp. 294–324, 2017.

[45] D. Escudero, S. Ghosh, M. Keller, R. Rachuri, and P. Scholl, "Improved primitives for MPC over mixed arithmetic-binary circuits," in *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, D. Micciancio and T. Ristenpart, Eds., vol. 12171. Springer, 2020, pp. 823–852. [Online]. Available: https://doi.org/10.1007/978-3-030-56880-1_29

[46] S. Iraklis, R. Dragos, M. A. Mustafa, M. Bart, and P. Panos, "Hermes: Scalable, secure, and privacy-enhancing vehicle access protocol (check)," *IACR Cryptology ePrint Archive*, vol. 2020, p. 00, 2020.

[47] W. Diffie, P. C. van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Des. Codes Cryptography*, vol. 2, no. 2, pp. 107–125, 1992. [Online]. Available: http://dx.doi.org/10.1007/BF00124891

[48] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, 1988.

[49] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *J. ACM*, vol. 33, no. 4, pp. 792–807, 1986.

[50] M. Bellare, A. Boldyreva, and S. Micali, "Public-key encryption in a multi-user setting: Security proofs and improvements," in *Advances in Cryptology - EUROCRYPT*, 2000, pp. 259–274.

[51] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption," in *FOCS*, 1997, pp. 394–403.

[52] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *FSE*, ser. LNCS, vol. 3017. Springer, 2004, pp. 371–388.

[53] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. ACM, 2020, pp. 1575–1590. [Online]. Available: https://doi.org/10.1145/3372297.3417872

[54] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, ser. Lecture Notes in Computer Science, J. Feigenbaum, Ed., vol. 576. Springer, 1991, pp. 420–432. [Online]. Available: https://doi.org/10.1007/3-540-46766-1_34

[55] OpenSSL, "Cryptography and SSL/TLS Toolkit," https://www.openssl.org/, accessed December, 2020.

[56] C. Huang, R. Lu, J. Ni, and X. Shen, "DAPA: A decentralized, accountable, and privacy-preserving architecture for car sharing services," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 4869–4882, 2020. [Online]. Available: https://doi.org/10.1109/TVT.2020.2980777

[57] A. Madhusudan, I. Symeonidis, M. A. Mustafa, R. Zhang, and B. Preneel, "Sc2share: Smart contract for secure car sharing," in *Proceedings of the 5th International Conference on Information Systems Security and Privacy, ICISSP 2019, Prague, Czech Republic, February 23-25, 2019*, P. Mori, S. Furnell, and O. Camp, Eds. SciTePress, 2019, pp. 163–171. [Online]. Available: https://doi.org/10.5220/0007703601630171

[58] D. De Troch, "dpace, a decentralized privacy-preserving, yet accountable car sharing environment," 2020.

[59] P. Papadimitratos, L. Buttyán, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J. Hubaux, "Secure vehicular communication systems: design and architecture," *IEEE Commun. Mag.*, vol. 46, no. 11, pp. 100–109, 2008. [Online]. Available: https://doi.org/10.1109/MCOM.2008.4689252

[60] H. Jin, M. Khodaei, and P. Papadimitratos, "Security and privacy in vehicular social networks," *CoRR*, vol. abs/2001.08014, 2020. [Online]. Available: https://arxiv.org/abs/2001.08014

[61] M. Raya, P. Papadimitratos, and J. Hubaux, "Securing vehicular communications," *IEEE Wireless Commun*, vol. 13, no. 5, pp. 8–15, 2006.

[62] M. Khodaei, H. Jin, and P. Papadimitratos, "Towards deploying a scalable & robust vehicular identity and credential management infrastructure," *CoRR*, 2016.

[63] B. Groza, T. Andreica, A. Berdich, P. Murvay, and E. H. Gurban, "Prestvo: Privacy enabled smartphone based access to vehicle on-board units," *IEEE Access*, vol. 8, pp. 119 105–119 122, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3003574

[64] H. Qi, Z. Wan, Z. Guan, and X. Cheng, "Scalable decentralized privacy-preserving usage-based insurance for vehicles," *IEEE Internet of Things Journal*, 2020.

[65] C. Troncoso, G. Danezis, E. Kosta, J. Balasch, and B. Preneel, "PriPAYD: Privacy-Friendly Pay-As-You-Drive Insurance," *IEEE TDSC*, vol. 8, no. 5, pp. 742–755, 2011.

[66] D. R. Stinson, "Some observations on the theory of cryptographic hash functions," *Des. Codes Cryptography*, vol. 38, no. 2, pp. 259–277, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10623-005-6344-y

[67] P. Rogaway, "Formalizing human ignorance," in *Progressin Cryptology - VIETCRYPT 2006, First International Conferenceon Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, 2006, pp. 211–228.

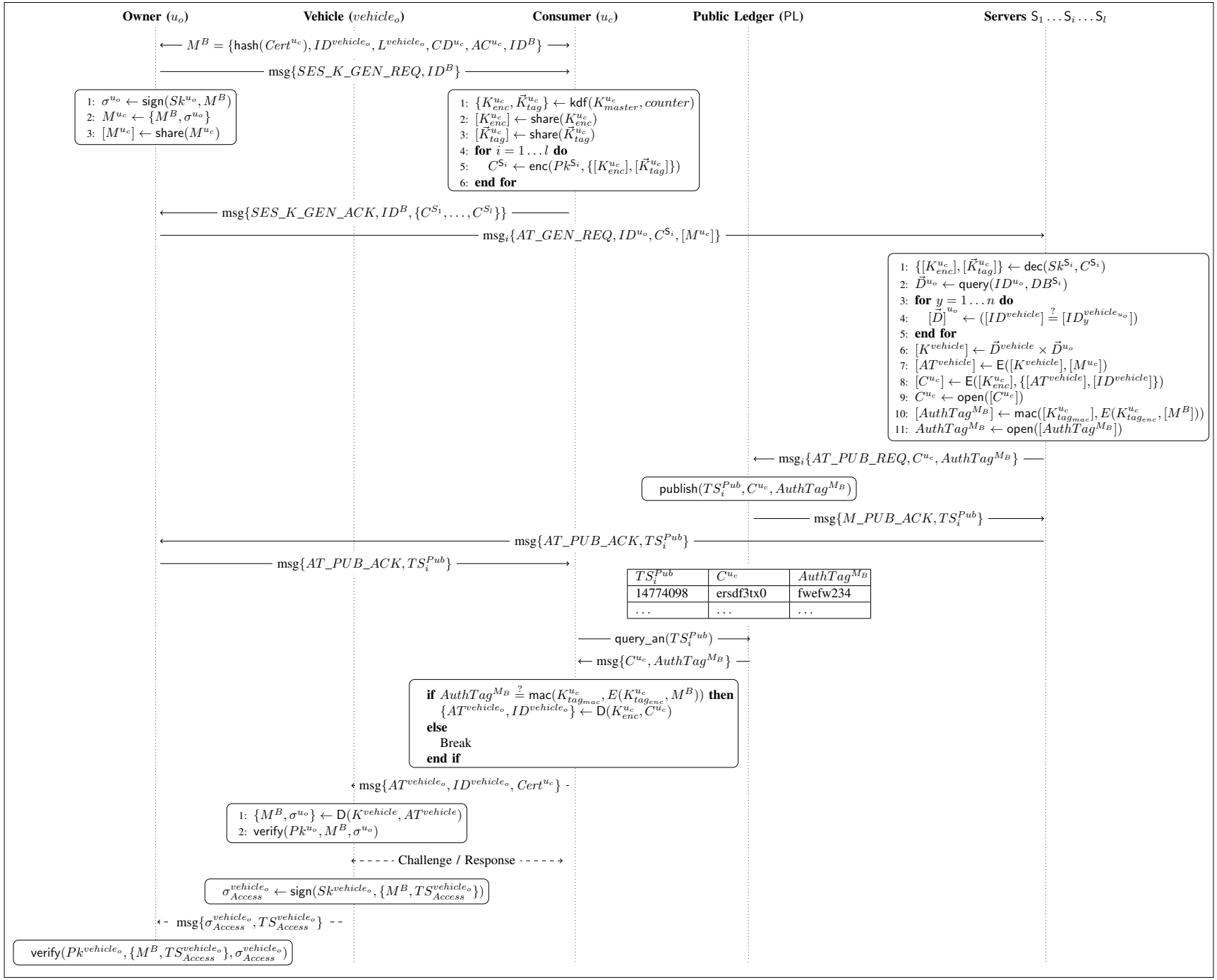

Fig. 9: HERMES complete representation.

APPENDIX A
EXTENDED SECURITY AND PRIVACY ANALYSIS

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

A. Cryptographic Primitives

The security definitions for signature schemes and MAC functions are inspired by Goldwasser et al. [48], for key derivation utilizing PRF by Goldreich et al. [49], for public key encryption by Bellare et al. [50], and for symmetric key encryption by Bellare et al. [51].

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 VSSP, and E and mac are used for independent keys for every fresh evaluation of the protocol. We refer to Bellare et al. [50] 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*

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

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 key derivation function using 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*

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

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 [52], [66], [67] for further discussion.

B. Analysis

We prove that HERMES satisfies the security and privacy requirements of Section II 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*

$$\mathrm{Adv}_{\mathrm{sign}}^{\mu_o+\mu_{\mathrm{vehicle}}\text{-euf}}(2q,t)+\mathrm{Adv}_{\mathrm{prf}}^{\mu_c\text{-prf}}(2q,t)+\mathrm{Adv}_{\mathrm{enc}}^{l\text{-pke}}(t)+\mathrm{Adv}_{\mathsf{E}}^{2q+\mu_{\mathrm{vehicle}}\text{-ske}}(3q,t)+\mathrm{Adv}_{\mathrm{mac}}^{q\text{-mac}}(q,t)+\mathrm{Adv}_{\mathsf{hash}}^{\mathrm{col}}(t)\ll 1\,,$$

where μ_o denotes the maximum number of u_os, μ_c the maximum number of u_cs, $\mu_{\mathrm{vehicle_o}}$ the maximum number of vehicles, l the number of servers in the VSSP, q the total times the system gets evaluated, and t the maximum time of any adversary.

Then, HERMES fulfills the security and privacy requirements of Section II.

Proof. Recall from Section II that u_os and CM are honest-but-curious whereas u_cs and outsiders may be malicious and actively deviate from the protocol. Vehicles are trusted.

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

$$\mathrm{Adv}_{\mathrm{prf}}^{\mu_c\text{-prf}}(2q,t)\,, \tag{3}$$

as in every of the q evaluations of HERMES 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 VSSP 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

$$\mathrm{Adv}_{\mathrm{enc}}^{l\text{-pke}}(t)\,, \tag{4}$$

we can for simplicity replace it with a perfectly secure public-key encryption ρ^{VSSP} to the VSSP directly (an encryption does not reveal its origin and content, and only VSSP can magically decrypt), therewith eliminating the fact that VSSP 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, $\pi^{vehicle_{u_o}}$ for the vehicles, unique $\pi_{enc}^{u_c}$'s for every new encryption under u_c's session keys, and $\pi_{tag_{enc}}^{u_c}$'s for every new encryption in the tag computation under u_c's session keys, at the cost of

$$\mathrm{Adv}_{\mathsf{E}}^{2q+\mu_{\mathrm{vehicle}}\text{-ske}}(3q,t)\,, \tag{5}$$

as there are $2q+\mu_{\mathrm{vehicle_o}}$ different instances involved and at most $3q$ evaluations are made in total. Note that this means that, instead of randomly drawing $K_{enc}^{u_c}\leftarrow\$^{u_c}$, we now randomly draw $\pi_{enc}^{u_c}\xleftarrow{\$}\mathrm{Func}(\mathsf{E})$. Likewise, for $K_{tag_{enc}}^{u_c}\leftarrow\$^{u_c}$ we now randomly draw $\pi_{tag_{enc}}^{u_c}\xleftarrow{\$}\mathrm{Func}(\mathsf{E})$.

We are left with a simplified version of HERMES, namely one where the VSSP 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^{VSSP}, and symmetric-key encryptions are replaced by perfectly random invertible functions $\pi^{vehicle_{u_o}}$, $\pi_{enc}^{u_c}$, and $\pi_{tag_{enc}}^{u_c}$. The simplified system is given in Figure 10. Here, the derivation of the vehicle key (or, formally, the random function corresponding to the encryption) from the database is abbreviated to $\pi^{vehicle_{u_o}}\leftarrow\mathsf{query}(ID^{u_o},DB^{\mathsf{VSSP}})$ 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 HERMES, hence they know each other by design.

a) SR1 – Confidentiality of M^B: In one evaluation of the protocol, u_c, u_o, the *trusted* VSSP, and the shared vehicle learn the booking details by default or design. The booking details only become public through the values C^B and C^{u_c} satisfying

$$AuthTag^{M_B}=\mathsf{mac}(K_{tag_{mac}}^{u_c},\pi_{tag_{enc}}^{u_c}(M^B))\,, \tag{6}$$

$$C^{u_c}=\pi_{enc}^{u_c}(\{\pi^{vehicle_{u_o}}(\{M^B,\sigma^{u_o}\}),ID^{vehicle}\})\,. \tag{7}$$

The latter value reveals nothing about M^B as $\pi_{enc}^{u_c}$ is randomly generated for every evaluation, whereas the former value reveals nothing about M^B as $\pi_{tag_{enc}}^{u_c}$ and the key $K_{tag_{mac}}^{u_c}$ are randomly generated for every evaluation. The nested encryption $\pi_{enc}^{u_c}\circ\pi^{vehicle_{u_o}}$ does not influence the analysis due to the mutual independence of the two functions.

b) SR2 – Authenticity of M^B: An owner who initiates the AT generation and distribution, first signs the booking details using its private key before sending those to the VSSP in shares. Therefore, once the vehicle receives the token and obtains the booking details, it can verify u_o's signature on the booking details. In other words, the vehicle 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 vehicle 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

$$\mathsf{E}_1 \ : \ \mathcal{A} \text{ forges } \mathsf{sign}(Sk^{u_o}, \cdot) \text{ for some } Sk^{u_o} \, . \tag{8}$$

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

$$\mathsf{E}_2 \ : \ \mathcal{A} \text{ finds a collision for } \mathsf{hash} \, . \tag{9}$$

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

c) SR3 – Confidentiality of $AT^{vehicle_o}$: The AT is generated by the VSSP obliviously (as it is trusted), and only revealed to the public in encrypted form, through C^{u_c} of (7). Due to the uniform drawing of $\pi^{u_c}_{enc}$ (and the security of ρ^{VSSP} used to transmit this function), only the legitimate user can decrypt and learn the AT. It shares it with the vehicle over a secure and private channel.

d) SR4 – Confidentiality of $K^{vehicle_o}$: By virtue of our hybrid argument on the use of the symmetric-key encryption scheme, $\mathsf{E}_{K^{vehicle}}$ got replaced with $\pi^{vehicle_{u_o}}$, which itself is a keyless random encryption scheme. As the key is now absent, it cannot leak.

Moreover, considering the VM as a TP, it holds all the secret keys of vehicles that produce. By vehicle owners registering their vehicles, the VM forwards the list of each vehicle $ID^{vehicle_o}$ to VSSP. To retrieve the yth key from DB^{S_i}, i.e., $[K^{vehicle_{u_o}}_y]$, each S_i performs an equality check over MPC. The comparison outcomes 0 for mismatch and 1 for identifying the vehicle at position y, i.e.,

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

from which the share of the vehicle's secret key, $[K^{vehicle}]$, can be retrieved. Due to the properties of threshold secret sharing, the secret vehicle keys stay secret to each S_i. Thus, no one including the PD of the vehicle owner, but VM and the vehicle holds the vehicle key.

e) SR5 – Backward and forward secrecy of $AT^{vehicle_o}$: The AT is published on PL as C^{u_c} of (7), encrypted using $\pi^{u_c}_{enc}$. Every honest u_c generates a uniformly randomly drawn function $\pi^{u_c}_{enc}$ 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 ATs for malicious users who may deviate from the protocol and reuse one-time keys.)

f) SR6 – Non-repudiation of origin of $AT^{vehicle_o}$: The vehicle, 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, $AuthTag^{M_B} \stackrel{?}{=} \mathsf{mac}(K^{u_c}_{tag_{mac}}, \pi^{u_c}_{tag_{enc}}(M^B))$. Note that u_c does not effectively verify $AT^{vehicle_o}$, but rather $AuthTag^{M_B}$. 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 (8). For the latter, denote the event that this happens by

$$\mathsf{E}_3 \ : \ \mathcal{A} \text{ forges } \mathsf{mac}(K^{u_c}_{tag_{mac}}, \cdot) \text{ for some } K^{u_c}_{tag_{mac}} \, . \tag{10}$$

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

g) SR7 – Non-repudiation of delivery of $AT^{vehicle_o}$: u_o can verify correct delivery through the verification of the message sent by the vehicle to the him/her, $\mathsf{verify}(Pk^{vehicle}, \{M^B, TS^{vehicle}_{Access}\}, \sigma^{vehicle}_{Access})$ at the end of the protocol. Security breaks only if the signature scheme is forgeable. Denote the event that this happens by

$$\mathsf{E}_4 \ : \ \mathcal{A} \text{ forges } \mathsf{sign}(Sk^{vehicle}, \cdot) \text{ for some } Sk^{vehicle} \, . \tag{11}$$

We thus obtain that a violation of SR7 implies E_4.

h) SR8 – Accountability of users (i.e., owners, and consumers): 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 VSSP 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 VSSP consists of three parties, it suffices two of such parties to reconstruct the secrets (for semi-honest and malicious cases).

i) PR1 – Unlinkability of u_c and the vehicle: 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 vehicle, which is supposed to learn u_c's identity so that it can perform proper access control. u_c consults PL over an anonymous communication channel. The booking details are transferred to and from the VSSP, but these are encrypted and do not leak by virtue of their confidentiality (security requirement SR1).

j) PR2 – Anonymity of u_c and the vehicle: The reasoning is identical to that of PR1.

k) PR3 – Undetectability of $AT^{vehicle_o}$ operation: Access token generation, update, or revocation is performed using the same steps and the same type of messages sent to the VSSP and PL. Hence, outsiders and system entities can not distinguish which operation has been requested.

l) Conclusion: In conclusion, HERMES operates securely as long as the costs of (3-5), together with the probability that one of the events (8-11) occurs, are sufficiently small:

$$\mathrm{Adv}_{\mathrm{prf}}^{\mu_c\text{-prf}}(2q,t) + \mathrm{Adv}_{\mathrm{enc}}^{l\text{-pke}}(t) + \mathrm{Adv}_{\mathrm{E}}^{2q+\mu_{\mathrm{vehicle}}\text{-ske}}(3q,t) + \Pr\left(\mathsf{E}_1 \vee \mathsf{E}_2 \vee \mathsf{E}_3 \vee \mathsf{E}4\right) \ll 1\,.$$

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

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

which completes the proof. □

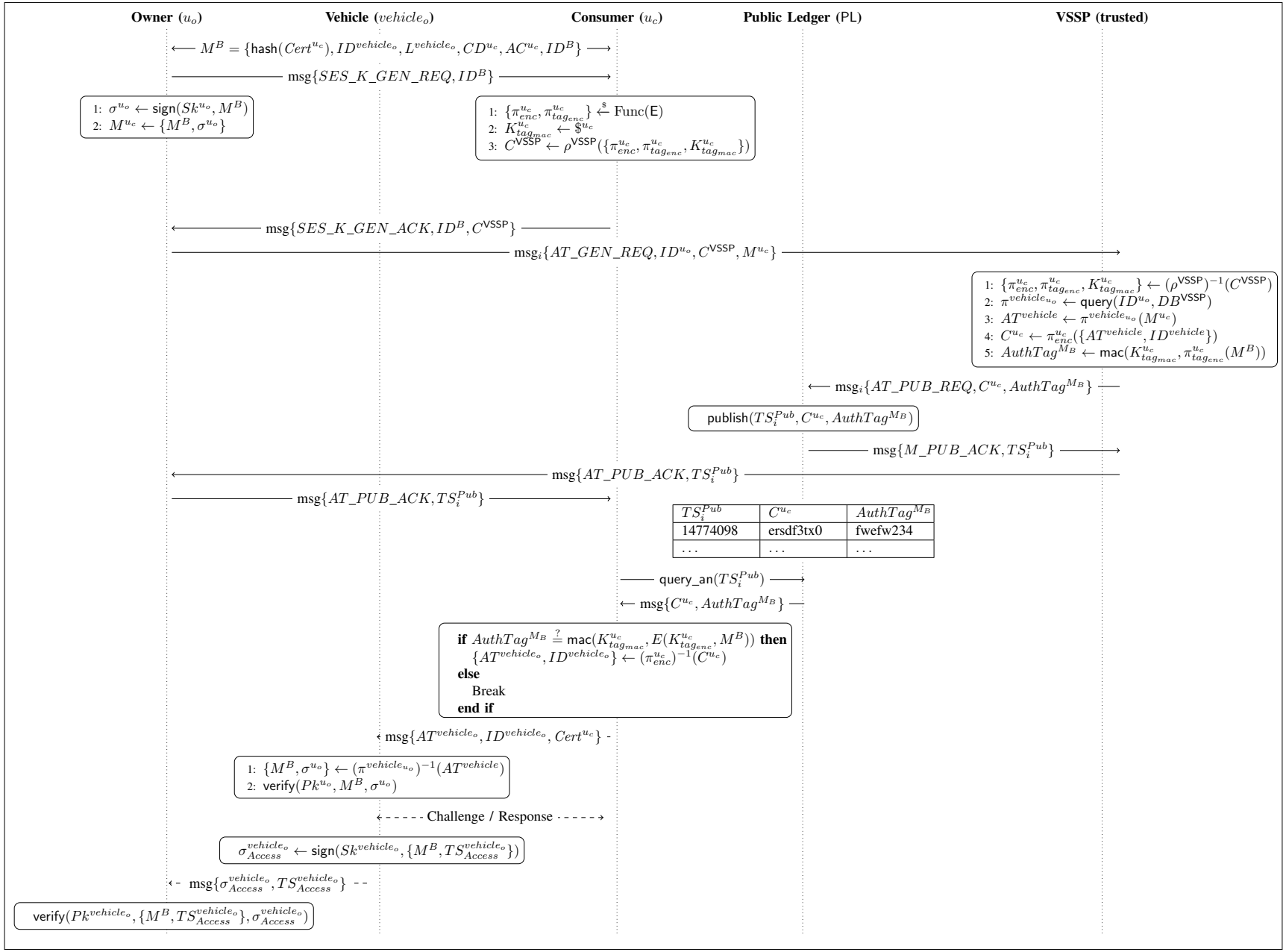

Fig. 10: HERMES complete representation.