IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 31, 2020, accepted June 15, 2020, date of publication June 19, 2020, date of current version July 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3003574

PRESTvO: PRivacy Enabled Smartphone Based Access to Vehicle On-Board Units

BOGDAN GROZA, (Member, IEEE), TUDOR ANDREICA, ADRIANA BERDICH, PAL-STEFAN MURVAY, (Member, IEEE), AND EUGEN HORATIU GURBAN

Faculty of Automatics and Computers, Politehnica University of Timisoara, 300223 Timisoara, Romania

Corresponding author: Bogdan Groza (bogdan.groza@aut.upt.ro)

This work was supported in part by the Ministry of Research and Innovation, CNCS-UEFISCDI, under Project PN-III-P1-1.1-TE-2016-1317, within PNCDI III, and in part by the Romanian Ministry of Research and Innovation, under Project 10PFE/16.10.2018, PERFORM-TECH-UPT (The increasing of the institutional performance of the Polytechnic University of Timişoara by strengthening the research, development and technological transfer capacity in the field of Energy, Environment and Climate Change, within Program 1-Development of the National System of Research and Development, Subprogram 1.2-Institutional Performance-Institutional Development Projects-Excellence Funding Projects in RDI, PNCDI III).

ABSTRACT Smartphones are quickly moving toward complementing or even replacing traditional car keys. We advocate a role-based access control policy mixed with attributes that facilitates access to various functionalities of vehicular on-board units from smartphones. We use a rights-based access control policy for in-vehicle functionalities similar to the case of a file allocation table of a contemporary OS, in which read, write or execute operations can be performed over various vehicle functions. Further, to assure the appropriate security, we develop a protocol suite using identity-based cryptography and we rely on group signatures which preserve the anonymity of group members thus assuring privacy and traceability. To prove the feasibility of our approach, we develop a proof-of-concept implementation with modern smartphones, aftermarket Android head-units and test computational feasibility on a real-world in-vehicle controller. Our implementation relies on state-of-the-art cryptography, including traditional building blocks and more modern pairing-friendly curves, which facilitate the adoption of group signatures and identity-based cryptography in automotive-based scenarios.

INDEX TERMS Access control, authentication, automotive applications, cryptography, smart devices.

I. INTRODUCTION AND MOTIVATION

The generous interface of modern smartphones and their ubiquitousness opens road for adding access control to various car functionalities as well as for remote configuration and rights delegation. In contrast, classical radio-frequency (RF) and/or mechanical vehicle keys are rigid and lack in terms of flexibility and functionalities. Perhaps surprising, despite their simplicity, classical RF keys have shown numerous flaws that led to a plethora of reported attacks targeting weaknesses in regular RF keys [59], [63], open-source immobilizer specifications [53], or passive keyless entry systems [24], [26], [64]. So it seems that the security of traditional car keys is lacking in many respects. The causes are numerous, including poor selection of cryptographic algorithms or poor randomness, etc. This merely complements a landscape which became familiar to us in the recent years as cars are unsatisfactory prepared in terms of security, e.g., [15], [39], [45].

By using smartphones, specific applications can be targeted and the interface customized to gain access to virtually any device or component from the car. Moreover, rights delegation can address complex scenarios due to increased connectivity at a global scale. Consequently, replacing traditional keys with smartphones appears like a natural step in achieving increased usability and an improved user experience. This is in fact proved both by many research works (which we separately address in the related work section) but also by recent industry efforts such as the Car Connectivity Consortium which drives a global initiative of top players from the automotive domain for car-to-smartphone connectivity.[1] To place the current research into context, Figure 1 provides a depiction of the interface that we implemented in PRESTvO. Some car functionalities are outlined and four user roles are

The associate editor coordinating the review of this manuscript and approving it for publication was Junggab Son.

[1] https://carconnectivity.org/

FIGURE 1. PRESTvO user interface.

TABLE 1. An example of role access rights.

	Owner	Driver	Technician	Child Occuppant	Valet	Passenger
Start Engine	–––	––e	––e	–––	––e	–––
Open Trunk	––e	––e	––e	–––	–––	––e
Open Doors	––e	––e	––e	––e	––e	––e
Limit speed	rw–	rw–	–––	–––	–––	–––
Fuel Level	r––	r––	r––	–––	r––	–––
Diagnosis	––e	––e	––e	–––	–––	–––
SW Update	–––	–––	––e	–––	–––	–––
Park car	–––	––e	––e	–––	––e	–––
Home	––e	––e	––e	–––	–––	–––
Alarm	––e	––e	––e	–––	––e	–––
Start A/C	––e	––e	––e	–––	––e	––e
Defrost	––e	––e	––e	–––	––e	–––
Mirrors	––e	––e	––e	–––	––e	–––
Lights	––e	––e	––e	–––	––e	–––
Play music	––e	––e	––e	––e	––e	––e
Limit volume	rw–	rw–	rw–	–––	–––	–––
Trip Computer	rw–	rw–	r––	–––	–––	–––

displayed: car owner, driver, passenger and a technician role. Table 1 summarizes role rights which are marked in a similar fashion to access rights over files in a modern operating system.

However, replacing traditional keys with smartphones comes with additional security and privacy challenges. For example, smartphones will have to pair with the car over a wireless communication interface such as Wifi, Bluetooth or NFC. But all these interfaces have been commonly found vulnerable, e.g., key reinstallation attacks on the WPA2 have attracted much attention a few years ago [58], some vulnerabilities of NFC-based payments were shown by [28], and quite long list of potential exploits on Bluetooth can be found in [1], [33] with more recent results showing attacks on Bluetooth elliptical curve based pairings in [9]. Such vulnerabilities can be overcome only at the application layer by proper protocol designs based on specific cryptographic functionalities. This is precisely our intention here in PRESTvO, to design, implement and test a secure protocol for gaining access from a smartphone to a car via an existing wireless interface, e.g., WiFi or Bluetooth. The protection mechanism will not be limited to the communication channel, it will also have to address on-device adversaries, such as malicious manufacturers, that may want to retrieve information from the vehicle in order to track the users. Recent incidents showed that major companies can be involved in privacy leakages either by their own consent or due to data breaches [35], [65]. Consequently, we focus both on the security of the communication channel and on the privacy of the users which is assured by the more demanding cryptographic group signatures. In this way, as the identity remains hidden behind the group key, the car should be unable to log any specific information about the person which gained access to the car. In what follows we briefly discuss the cryptographic toolset that enables us to reach the desired security and privacy goals.

Cryptographic Building Blocks: In principle, cryptography offers a comprehensive set of functions, i.e., cryptographic primitives, that can assure various security objectives , e.g., authenticity, confidentiality, etc. Besides regular symmetric and asymmetric primitives, i.e., public-key encryptions and digital signatures, more recent advances set room for more exotic cryptographic functionalities. These include identity-based cryptography where the identity of a user can be used to derive his public key or group signatures where the identity of a user can be preserved anonymously under the public-key of a group (still allowing the group manager to trace the user if a dispute arises). There is no question that these cryptographic functionalities will be sooner or later adopted by the industry. The AUTOSAR standard already defines interfaces for regular cryptographic building blocks in the automotive domain [4], [5]. Identity-based signatures are part of an ISO standard [36] published long ago and it targets embedded devices such as smartcards. There are numerous research works that advocate for the use of these cryptographic building blocks for car access-control, these are summarized in Table 2 and will be discussed in the related work section. However, the heterogeneity of the addressed environment, where smartphones interact with in-vehicle units, raises performance concerns. Our work builds upon various cryptographic building blocks (identity-based signatures, group signatures, etc.) and besides designing a car access protocol we try to bring answers regarding the feasibility of deploying this solution both on mobile devices and on in-vehicle components.

TABLE 2. Summary of some existing proposals for car keys with enhanced capabilities (in chronological order by year of publication).

Paper	Main Concept	Platform for Key Deployment	Comm. intrf.	Other cryptographic rq.	Car AC	Rights deleg.
[20]	Access control rights delegation with trusted execution environment on Android	Nexus S (Android 2.3.3 patched with TrustDroid security extensions)	NFC	–	–	✓
[12]	Car access and rights delegation from an Android smartphone, security reinforced by smart-card	Samsung Galaxy S3, Arduino Uno (8-bit microcontroller for car immobilizer)	NFC	–	✓	✓
[54]	Secure vehicle-to-cloud communications based (uses OAuth & HSM)	–	NFC	–	✓	–
[37]	Rights Management with NFC Smartphones and Electronic ID Cards	Blackberry Bold 990	NFC	–	–	–
[32]	Pairing cars and smartphones by OOB (out-of-band) channels: light and sound (augmented by the Encrypted Key Exchange protocol, Diffie-Hellman version)	Motorola Droid 1, Android 2.2.3 (Froyo)	BT	–	✓	–
[52]	Car sharing functionality with secure multi-party computation	Intel Core i7, 2.6 Ghz CPU, 8GBof RAM	NFC, BT	Secure multi-party computation	✓	✓
[19]	Car sharing by short-range wireless communication (RFID, NFC, BLE) with two-factor authentication	Samsung S4 (Android 5.0), Google Nexus 5 Android 5.1 (Lollipop), + Giesecke & Devrient Mobile Security Card (MSC) with DESFire applet, contactless Mifare DESFire EV1 smartcard, Samsung Galaxy Gear SM-V700 smartwatch (Android 4.2)	NFC, BLE	–	✓	–
[62]	Car sharing in a hierarchical system between KGC (key generation center), owner/rental company, user	Android Nexus 5	NFC	Identity-based encryption and signatures	✓	✓
[29]	Rights delegation to a car from a low-cost embedded platform by using only symmetric crypto-primitives	MSP430 (16-bit microcontroller from Texas Instruments for key), Freescale S12 (as car immobilizer)	RF	–	✓	✓
Our Work	Car access and rights delegation, preserving anonymity by using group signatures, eliminating PKI by using identity-based cryptography	LG, Samsung J5, ERISIN & PNI car head-units, Infienon TC297 (for car immobilizer)	NFC, BT, WiFi	Identity-based signatures, group signatures	✓	✓

Garnering Advantages From Group and Identity-Based Signatures: While some of the cryptographic building blocks that we use are more demanding, e.g., group signatures, there are clear benefits behind using them in this car access scenario. By using group signatures, the car will be aware which role is accessing the car, i.e., owner, driver, passenger or technician, but will have no information on the entity that instantiated the specific role. That is, there may be multiple drivers, passenger, technicians and even car ownership may be shared, while the car (and implicitly the car manufacturer) will be unable to log information regarding the exact user (it is only the role which stays visible). With specific functionalities of group signatures, which are later discussed in the protocol design section, the exact user can still be traced by the group manager in case when a dispute arises. Our design emphasizes on the right of ownership and thus we let the car owner to be in possession of the group manager secret key (other deployments may choose to attribute this functionality to a trusted authority). While traditional signatures may preserve the anonymity of the users by using pseudonyms in the certificates, it is still possible to separate between users based on their distinct public keys and additional information, e.g., driving time and location leakages, may be corroborated for the de-anonymization of the user behind the pseudonym. Group signatures provide better privacy guarantees in this respect.

The use of identity-based signatures will make public-keys far easier and more intuitively to manage. For example, we show in Figure 2 a user attempting to collect the identity of a car from its VIN number that is located on the driver's side where the windshield meets the dashboard (this is not the only location but it is the most common for cars). To recognize the text from the VIN number, we used the text detection package from the Mobile Vision API[2] which provides a framework for objects identification in photos and video. The text of the VIN number from the dashboard was collected with almost 100% accuracy showing that there are no technological shortcomings in this respect (provided that the dashboard is clean and the phone camera properly pointed to the VIN). Clearly, other elements, such as the license plate, can be used for the same purpose. The VIN however is immutable and will remain forever associated with a car, while the license plate can change. For example, in many countries new cars have temporary license plates until they are sold to the first customer. Thus the VIN number provides a more reliable identity for the car. This functionality may prove particularly useful in car-rental scenarios as well as for companies which share several cars between their employees.

System Design Goals: We now briefly discuss the design goals of our proposal. Figure 3 gives an overview of the

[2]https://developers.google.com/vision

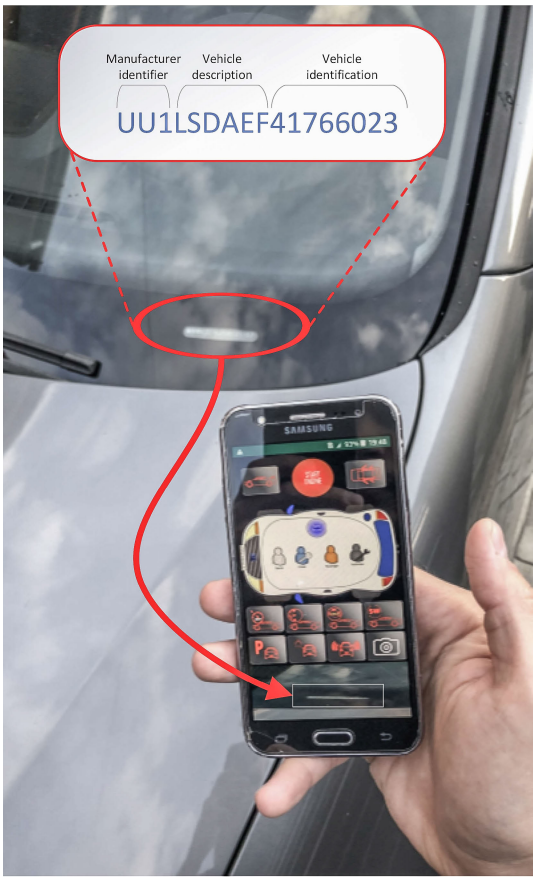


FIGURE 2. User handling the PRESTvO application to colect the car VIN number, i.e., the identity of the car.

addressed system. A user requests a particular functionality of the car which is to be executed by some in-vehicle electronic control unit (ECU). Access is mediated by PRESTvO. First an authentication service is called which verifies the identity of the user and the role he invoked. If the identity and roles are verified, the role along with the request is passed to the access control service which in turn verifies authorization for the particular request and returns the access decision. In case of a positive decision, the request is passed to the car which in turn responds according to the request, i.e., by executing the particular functionality and sending a response message. The user receives a response from PRESTvO which may be negative if his function request could not be approved or a confirmation otherwise. We design the protocol behind PRESTvO with both security and privacy in mind and also without forgetting that we address functionalities inside a car and target real-world automotive-grade embedded devices. The following summarizes the goals of our work:

1) *secure access control* to all vehicle functionalities mediated by the use of smartphones is the prime intention of our work,
2) a *flexible access control policy* determined by roles and attributes falling middle of the road between RBAC and attribute-based access control (ABAC) which seems the best option due to the variety of car-usage scenarios,
3) *rights delegation* and also rights revocation directly from the smartphone is a natural functionality,
4) *user privacy*, by which we keep the identity of the user anonymous to the car and manufacturers, is enforced by the use of group signatures which hide the identity of a specific user inside a group,
5) *user traceability* in case when a dispute arises is a mandatory procedure due to legal implications, e.g., the car may be involved in an accident and it becomes necessary to be able to trace a particular user,
6) *flexible use of wireless interfaces WiFi, Bluetooth and NFC* according to existing support on the hardware that we used (smartphones and vehicle head-units),
7) *comprehensive performance tests* on real-world automotive grade controllers are a must in order to prove that implementation is realistic with respect to state-of-the-art in automotive on-board units.

A. RELATED WORK

Related work on vehicle access control and rights delegation is extensive. While only a limited amount of research papers have been focusing on traditional car immobilizers, e.g., [42], there is quite a large number of recent works that address the use of smartphones for accessing vehicle functionalities. In what follows, we give a brief overview of the existing proposals and summarize the most relevant of them in Table 2. We also point out in the table whether the work uses any enhanced cryptographic capabilities, e.g., identity-based signatures, group signature, etc., besides the regular symmetric/asymmetric primitives which are present in all of the works.

The use of smartphones for access-control systems inside buildings and as replacement of traditional physical keys has been explored as early as the works of [6], [7].

In [12] a full platform for car access and rights delegation from an Android smartphone is presented. The security is reinforced by the use of a smart-card and the authors present both a proof-of-concept implementation and strong security arguments by model checking with ProVerif. A hierarchical car sharing architecture is proposed in [62]. The authors consider only a simplified hierarchy with 3 levels: a key generation center, the owner or the rental company and the end-user. A proof of concept implementation is presented on an Android Nexus 5 smartphone, the protocol relies on identity based encryptions and signatures. The authors of [20] propose a generic smartphone-based NFC access control system that allows access rights delegation. The access control system is based on a multi-level smartphone security architecture designed to provide trusted execution and storage environments. Formal security analysis as well as a proof of concept implementation on a simplified system model are provided. Several NFC-based use cases for the automotive environment are also described and implemented in [50].

Another secure access control system for car sharing is proposed in [19]. It employs two-factor authentication provided by an RFID token and a soft token to enable access to offline

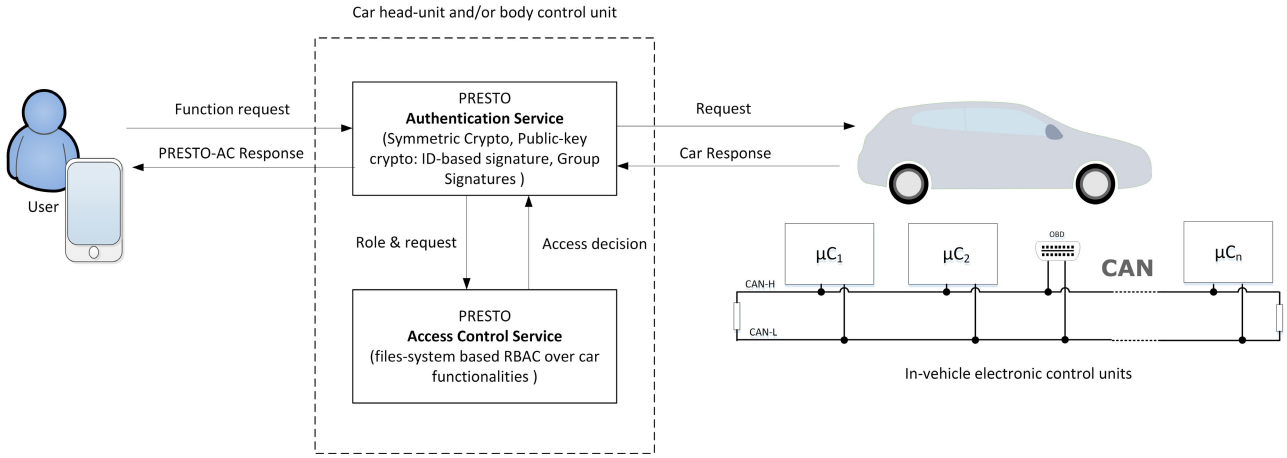


FIGURE 3. PRESTvO system design.

cars. The proposed instantiation uses a secure execution platform that can be implemented on devices such as smartcards or smartwatches. Another approach for car sharing is proposed in [52]. The paper presents a decentralised protocol that provides both security and privacy allowing owners to share their cars. Protocol analysis and a proof-of concept implementation are also considered.

Other works that use smartphones for gaining access to vehicles are [2] and [11]. The Green Move project described in [2] is a vehicle sharing system. In this project, the vehicles are equipped with Green e-box devices, which communicate with a smartphone via Bluetooth and with the cloud (Green Move Center) via HTTP. Using the smartphone application, the user retrieves the valid electronic key from the Green Move Center, which contains an encrypted ticket, the start time of reservation as well as all the information to identify the car. The encrypted ticket is used to lock/unlock the doors. In [11], the Terminal Mode technology is described. This technology integrates the smartphone into the car head unit. In this scenario, the input and output functions are the responsibilities of the car head unit, while the smartphone acts as the application platform. New functionalities can be added to the car head unit by easily upgrading the smartphone.

Some works are focused on cost-efficient solutions. The implementation of a dedicated device for car-rights delegation on low-cost MSP430 microcontrollers from Texas Instruments is discussed in [29]. A keyless car access system using RFID cards (e-driver licenses) is proposed in [34]. In this scenario, each driver license is assigned to the driver's identity based on an RFID card using a serial number stored in the cloud database. If the serial number exists in the database, the driver can use the car and the owner knows who is driving the car based on information provided by a smartphone application.

Pairing mobile devices with cars has also been targeted. A secure pairing between mobile devices and vehicles based on out-of-band (OOB) channels is proposed in [32]. The authors present several key agreement protocols using light and sound as OOB channels. Protocol analysis with AVISPA [3] as well as implementations are also provided.

Privacy concerns for smartphone applications in the automotive domain have been also addressed for pay-by-phone parking systems [27] or GPS tracking [43].

B. SELECTING SETUP COMPONENTS

Table 3 provides a summary of the devices that we used in our setup. In what follows, we discuss these in detail.

1) ANDROID HEAD UNITS AND SMARTPHONES

For our experimental setup we acquired two Android head-units with similar computational/communication capabilities. The first of them from ERISIN was designed to replace SEAT and VW head units. The head unit provides a 9-inch capacitive display with 1024 × 600 resolution running the Android 7.1 Nougat OS. The CPU it uses is an Allwinner Quad-Core T3 SoC (Quad-core Cortex A7, 1.63GHz and Mali400 MP2 GPU), 2GB RAM and 16GB internal memory. The storage can be increased by using microSD cards or USB connected memory devices. The T3 SoC integrates a high number of peripherals providing support for a multitude of standards: USB, SATA, UART, TWI, SPI, EMAC, GMAC, PS2. The unit offers wireless Bluetooth and Wi-FI 802.11b/g/n connectivity. In addition it integrates: GPS, AM/FM radio tuner, RDS, DAB/DAB+ and CAN communication. Also this unit includes an USB connected rear view camera. Diagnostics information can be retrieved by using the included Bluetooth OBD2 Module. The second aftermarket head unit, PNI A8020 HD, is a generic replacement head unit. This is a lower cost version but is essentially deployed on almost the same hardware. It comes with a 7-inch capacitive display and 1GB RAM but it uses the same Allwinner Quad-Core T3 SoC along with the same level of connectivity.

TABLE 3. Devices from our experiments: smartphones, head-units and in-vehicle control unit.

Device	Android	CPU	Memory	WiFi	BT
LG Optimus P700	4.0.3	1.0 GHz Cortex-A5	4 GB (2.4 GB user available) 512 MB RAM	Wi-Fi 802.11 b/g/n, Wi-Fi Direct, hotspot, DLNA	3.0, A2DP
Samsung J5	5.1.1	Quad-core 1.2 GHz Cortex-A53	8 GB, 1.5 GB RAM	Wi-Fi 802.11 b/g/n, Wi-Fi Direct, hotspot	4.1, A2DP
Samsung S5	6.0.1	Quad-core 2.5 GHz Krait 450	16 GB, 2 GB RAM	Wi-Fi 802.11 a/b/g/n/ac, dual-band, Wi-Fi Direct, hotspot	4.0, A2DP, EDR, LE, aptX
Samsung S7	6.0.1	Octa-core (4x2.3 GHz Mongoose & 4x1.6 GHz Cortex-A53)	32 GB, 4 GB RAM	Wi-Fi 802.11 a/b/g/n/ac, dual-band, Wi-Fi Direct, hotspot	4.2, A2DP, LE, aptX
Head-unit PNI/ERISIN	7.1.1	Quad-core 1.63 GHz Cortex A7	12 GB, 1/2 GB RAM	Wi-Fi 802.11 b/g/n, hotspot,	4.0, A2DP, BR/EDR
Infineon TC	N/A	Triple Core 300MHz TriCore	8 MB Flash, 728KB RAM	N/A	N/A

2) VEHICLE ON-BOARD UNITS

The on-board unit functionality can be built either as a stand-alone unit or as part of an ECU responsible for several other functionalities related to the body domain. We advocate for the latter since car access control is traditionally implemented as part of the body control module (BCM). The protocols implemented by the car access functionality are based on computationally intensive public key cryptography. Moreover, an embedded platform suitable to serve as the on-board unit as well as to implement other vehicle body functions should be able to perform all its designated functionalities in a timely manner. This calls for the use of a high performance automotive grade embedded platform capable of performing public key cryptography operations. We selected the TC297, an Infineon Aurix microcontroller, which can act in a real car as the BCM with on-board unit functionality. The multicore architecture of Aurix 32 bit microcontrollers is built to offer high performance. Covering automotive communication technologies such as CAN (and CAN-FD), FlexRay and Ethernet, the TC297 is suitable for a wide range of automotive applications. Additionally, the Aurix family introduces a hardware security module which provides random number generation, AES128 HW acceleration and a trusted execution environment for cryptographic algorithms. All these features make the TC297 a suitable candidate for the designated application.

C. SELECTING COMMUNICATION INTERFACES

A short discussion on the three communication interfaces, i.e., Bluetooth, WiFi and NFC, that we use in our deployment now follows. While each of them can be used for any of the protocol components, pros and cons exist. Also, some restrictions may occur due to the unavailability of some of them in existing components. For example, the head units that we use are not equipped with NFC readers while their Bluetooth connectivity allows only for media streaming. Next, we give a brief overview of these interfaces and how they are used in our practical implementation.

Bluetooth is a technology for wireless data transfer between devices for a short range with low power consumption. The maximum packet size that can be transferred on Bluetooth BR/EDR is 1021 bytes. In the last years, Bluetooth technology has been frequently used to communicate between the user and the car. The main use case is for the car infotainment system and in-vehicle wearables applications but also for car access control and maintenance tools. Bluetooth devices use profiles to specify the features that are supported and the type of data that can be transmitted/received. The infotainment units used in our work have four Bluetooth Profiles: Advanced Audio Distribution Profile (A2DP), Audio/Video Remote Control Profile (AVRCP), Hands-Free Profile (HFP), and Headset Profile (HSP). These profiles can be used only for multimedia functionalities. The smartphones that we used on this work have a richer set of Bluetooth profiles: Advanced Audio Distribution Profile (A2DP), Hands-Free Profile (HFP), Headset Profile (HSP), File Transfer Profile (FTP), Message Access Profile (MAP), Object Push Profile (OPP), and Phone Book Access Profile (PBAP). To transfer packets of bytes between two devices, both devices have to support at least one of the Bluetooth Profiles that uses the OBEX protocol (Object Exchange). The Bluetooth profiles that use OBEX are FTP, OPP, and MAP. Since our infotainment units do not have any Bluetooth profile with OBEX support, we used Bluetooth only between smartphones and relied on WiFi for communicating with the infotainment unit as we discuss next. However, Infotainment units with OBEX support for Bluetooth exist, so this is not a technical limitation for our protocol, it is just a small limitation of our setup.

Wireless networking (Wi-Fi) is a technology for communication with high speed data transfer. In automotive, Wi-Fi technology is sometimes used by infotainment systems and is an essential component for the connecting cars in vehicle-to-vehicle communication (V2V), vehicle-to-infrastructure (V2I) or vehicle to pedestrian communications, etc. A number of recent works have also focused on the use of WiFi for phone-to-phone communication inside vehicles, e.g., [51]. Consequently, we consider that deploying part of our protocol over WiFi is realistic and will benefit from a higher data rate. In particular, we find that Wi-Fi is specifically suitable for protocol components that rely on the larger group signatures.

Near-field communication (NFC) is a set of communication protocols which offers the possibility to establish a short-range communication between two electronic devices. NFC relies on RFID, having the operating frequency at 13.56 MHz and the bit rate between 106 kbit/s and 424 kbit/s. The communication range is up to 20 cm. A full NFC capable device can operate in one of the following three modes: card emulation, reader/writer and peer-to-peer. The first mode permits a smart NFC-enabled devices to act like smart cards, the second mode may be used for the reading and writing of NFC tags and the last mode, peer-to-peer, offers the possibility for two NFC-enabled smart devices to communicate in an ad-hoc manner. Each device that participates to the communication can be either the initiator or the target. A passive target can be powered by the RF field generated by the initiator. When compared to Bluetooth, which is also a short-range communication technology, NFC operates at a much shorter range and at slower speeds. On the other hand, the power consumption is an advantage of NFC as it consumes less power than Bluetooth. Another advantage of NFC is that it doesn't require pairing and, from the security point of view, a shorter communication range may preclude adversaries from intercepting the communication channel. Still, attacks have been reported on NFC as well, e.g., [25], [60]. We do use NFC for sharing access rights between smartphones but still rely on cryptographic building blocks to assure security.

Brief Discussion on Connectivity: The specific use-case that we address, i.e., car access mediated by smartphones, calls only for short distance connectivity. That is, the range of Bluetooth and WiFi is generally restricted to 10-100 meters [61] with possible extensions to around 200 meters for WiFi when the devices are outdoors. NFC targets different type of applications and is limited to a few dozen centimeters. This coverage is of course sufficient for a user that tries to gain access to the car. Other actions, such as blacklisting identities or certificate, i.e., certificate revocations, or removing certain user rights from the car, may be more efficiently performed remotely. In this case 4G/5G connectivity will be needed. This type of connectivity is within reach for modern vehicles and is in fact commonly required for vehicle-to-everything (V2X) communication. There is a large body of research works focusing on vehicular ad-hoc networks, routing and even security and privacy issues for such scenarios, e.g., [16], [41], [46]. Our work will focus on Bluetooth and WiFi communication which are more convenient for the scenario that we address.

II. DESIGN CONCEPT

In this section we discuss the design concept behind our proposal. Subsequently, we give precise details on each component of the proposed protocol suite.

A. ACCESS CONTROL CONCEPT

We now present the concept behind our access control policy. The access control procedure is based on role-based access control (RBAC) to which we add some attributes that are needed for the roles. Using RBAC seems natural in automotive environments since manufacturers can easily define specific roles for a car, e.g., driver, passenger, child occupant, etc., and each role may offer specific access rights to users. Using Discretionary Access Control (DAC) or alternatively Mandatory Access Control (MAC) are also viable alternatives but associating rights to specific roles seems the more natural approach for our car access control scenario. Roles will also enforce the anonymity of each actor carrying the role under group signatures. RBAC is well understood and standard specifications exist, e.g., [23]. Numerous extensions of it have also been discussed, e.g., using attributes [40], location-aware policies [17], public-key certificates [13], or trust [14], etc. This opens road for many future applications and while the basic RBAC may be somewhat rigid it can be easily augmented and made more flexible.

A graphical depiction of the proposed access control model is suggested in Figure 4. Following at least in part accepted/standardized definitions from [23], we briefly summarize core elements:

1) *Users* can either be individuals or entities instantiated by software agents, however, in the protocol descriptions that follow, we generally assume that users are persons requesting a particular action from the car,
2) *Role* represents the role played by an entity (an individual or some software agent). Roles include car owners, drivers, technicians, child occupants, etc., which are all played by users. Other roles such as the car rental company or the manufacturer may be played by software agents that delegate rights over the car or execute various tasks, e.g., a software update, etc.,
3) *Attributes* are characteristics associated to a user, they include: time, location, driver license, age, etc. For example, a technician may perform a particular update or access to a component only if he is in the range of a particular location (e.g., the authorized garage). Attributes are either numerical or boolean, each attribute can be set to $\perp$ when a particular attribute is not applicable to a particular user or in a particular scenario. For example, it may be irrelevant whether a technician has or not a driving license or sometimes location information may be unavailable while certain rights should be executed on the car.
4) *Objects* constitute the car functionalities intuitively viewed by us as files that may be classified into *macro-objects* intuitively viewed by us as folders. Macro-objects are the car functional domains related to engine, chassis, body and infotainment. Objects are the associated functionalities, e.g., adjusts seats or lights, use the infotainment unit, etc. To simplify our model in Figure 4 we have only considered macro-objects associated to four functional domains: engine, chassis, body and infotainment. This instantiation may be easily extended, we present what it seems sufficient for most scenarios that we could imagine.

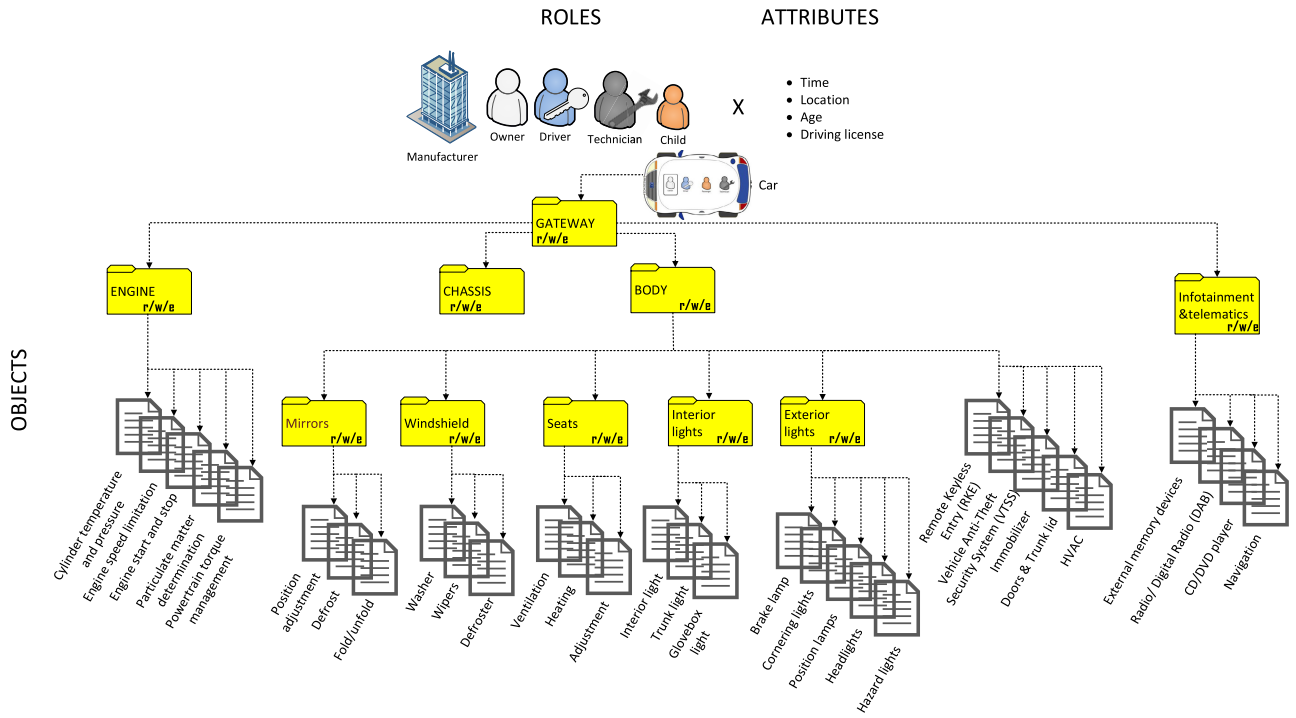


FIGURE 4. Overview of role-based access control in PRESTvO.

5) *Actions* are the activities that can be performed on an object. We view permissions over car functionalities similar to traditional Unix-like systems. Similarly to Unix files and folders we consider three types of actions: read simply lists the content that is as available, write modifies specific values and execute is the ability to run the particular functionality. That is $\mathsf{ACTIONS} = \{\mathsf{w}, \mathsf{r}, \mathsf{e}\}$ where each action is instantiated by a binary flag. The read permission allows a user read data, for example the user may read the fuel level, the mileage counter or the status of many other subsystems in the car. Write permissions are necessary to set specific values, for example setting the date and time, the cruise speed or resetting the trip computer. Execution rights, are required by specific programs, such as a movie player or by a software update. A technician may be entitled to make a software update, but not to play movies from Netflix, while for the passenger it's the reverse.
6) *Permissions* are the authorization given to a role over an object which makes $\mathsf{PERMISSIONS} = 2^{\mathsf{OBJECTS}\times\mathsf{ACTIONS}}$. For example, a potential car buyer may be authorized to list all functionalities in the car on his mobile phone, but without the possibility to run them. On the contrary, the manufacturer may be entitled to update the functionality. For simplicity, in the instantiation from Figure 4 we consider that permissions from a macro-object propagate identically over the objects below, but this can be changed according to practical needs.

Defining Roles, Persistent and Ephemeral Delegation: We consider that roles are predefined by the producer during the manufacturing process. Subsequently, the car owner is responsible for assigning the group public keys for each role. In this way the manufacturer cannot control a car owned by some individual but it does have control over the rights given to each role which is necessary to avoid misuse of a particular service. We consider that the owner of the car assumes a root role and that the root is the only role allowed to install public keys in the car. Rights delegation can be persistent or ephemeral and it can be performed by any actor in a specific role. As we later present in the experimental results section, we allow delegation from one smart-phone to another via NFC, but of course any other interface such as WiFi, Bluetooth or even 4G can be used for this purpose. We prefer NFC due to its short range which makes it harder for an adversary to eavesdrop on the channel (the delegation protocol is secure nonetheless, so any communication channel can be used). Ephemeral delegation is designed to be short lived, e.g., a car that is rent for weeks or days. Persistent delegation is designed to be long lived and makes role owners indistinguishable one from another, e.g., rights delegation to family of the car owner. It is obvious that the environment inside the car, e.g., mirror or seat position, does leak some information about the car occupant but addressing such issues is out of scope for us. We address privacy only from a protocol design perspective, i.e., the protocol run should not leak information about the role player in case of persistent users. Any other role can make ephemeral delegations of his access rights but persistent delegation can be done by the root only.

Rights revocation can be done by the root or by the delegating user. Both persistent and ephemeral users can be revoked. Revocation requires a certificate revocation list that is maintained in the cloud so the car must have Internet connectivity, e.g., via 4G. While this is not a complicated demand for modern cars, it may be the case that in certain situations the car does not have such connectivity. In this situation, rights are to be revoked as soon as the car connects to the Internet or as soon as they expire based on the attributes.

B. CRYPTOGRAPHIC TOOLS

1) SYMMETRIC PRIMITIVES

Our protocol makes use of standard Message Authentication Codes (MACs) and symmetric encryption which are instanced by SHA2 and AES in our proof-of-concept implementation. Besides these, we use more advanced cryptographic building blocks such as identity-based signatures and group signatures. We discuss these in more detail next. While symmetric primitives are present in all protocol actions along with asymmetric primitives, they play an exclusive role in the *on-the-fly* execution procedure which is designed for fast interaction with the car.

2) PUBLIC-KEY PRIMITIVES

Besides the more complex identity-based and group-based primitives that we discuss next, our protocol uses regular public-key cryptographic functions. These are generally used in practice to establish a secure communication channel between participants by facilitating the exchange of a secret session key (which is later used for symmetric encryption). To achieve this, we can rely on RSA [48] encryption or, for a more compact representation, on the elliptical curve version of the Diffie-Hellman key-exchange [18] , i.e., ECDH. Currently, 224-256 bit ECDH keys are viewed as the security equivalent of 2048-3072 bit RSA keys. While RSA leads to larger keys when compared to the elliptical-curve Diffie-Hellman, these are still easily manageable by modern smartphones. We later show in the experimental section that the computational overhead induced by RSA is of little concern while manipulating keys of a few thousand bits is even less of a problem for modern smartphones that have several giga-bytes of RAM.

Identity-based signatures (IBS) provide a more flexible framework which removes the need for exchanging digital certificates. The idea of identity-based signature originates from Shamir [49], for a more comprehensive introduction we refer the reader to [38]. In our proof-of-concept implementation, we consider both the original Shamir scheme as well as the Guillou-Quisquater scheme [30] which is part of the ISO/IEC 14888-2:2008 standard and is commonly proposed for use in embedded devices such as smart-cards [31]. While this scheme is a bit more computational intensive than the regular RSA, it can be easily handled by modern Android devices (as we discuss in the experimental section) and it removes the need for digital certificates. To clarify the functionalities behind an identity-based signature, we provide next a generic description for it. For brevity, the concrete description of the two identity-based schemes is moved to Appendix. An identity-based signature scheme consists of the following four algorithms:

1) $\mathsf{Setup}(k)$ is the key setup algorithm which outputs the master secret key msk and the global parameters pk.
2) $\mathsf{KeyDer}(\mathsf{msk}, I)$ is the key generation algorithm which uses the master secret key msk and the identity of the user I to output the private key of the user.
3) $\mathsf{Sign}(\mathsf{sk}, m)$ is the signature generation algorithm which uses the secret key sk on the message m to return the signature σ.
4) $\mathsf{Ver}(\mathsf{pk}, I, m, \sigma)$ takes as input the system global parameters pk, the identity of the user I, the message m and the signature σ and returns true or false accordingly.

Group signatures (GS) are used in order to provide the anonymity for group members. That is, the receiver of the signature can verify that it originates from a group member, but he cannot trace the particular group member. We use the scheme proposed by Boneh *et al.* in [10]. Technical details are dense, we stick to a brief formalism that help us clarify the operations required by the group signature. The group signature is a collection of three algorithms:

1) $\mathsf{Gen}(n)$ is the key generation algorithm that takes the number of group users n and returns the group secret master key gmsk, the group public key gpk and the vector containing the group secret keys $\mathsf{gsk} = \{\mathsf{gsk}_1, \mathsf{gsk}_2, \ldots, \mathsf{gsk}_n\}$ that will be distributed to the group users,
2) $\mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}_i, m)$ is the signature algorithm that takes as input the group public key gpk, the secret key of the signer gsk_i and a message m then returns the signature σ,
3) $\mathsf{Ver}(\mathsf{gpk}, m, \sigma)$ is the verification algorithm which takes as input the group public-key gpk, the message m and the signature σ and returns *true* if the signature is correct otherwise it returns $\perp$,
4) $\mathsf{Trace}(\mathsf{gpk}, \mathsf{gmsk}, m, \sigma)$ is the tracing algorithm that can determine the signer based on the group secret key gmsk, the group public-key gpk, the message m and the signature σ.

Additionally, the group signature has mechanisms for revoking the keys. This will require updating the public key of the group. For simplicity, we skip formalism for this procedure.

As a general rule we assume that all signatures are time-stamped, i.e, they contain a timestamp and a loose time synchronization exists between all devices in the scheme. Such a requirement is in fact ubiquitous in Internet security and should not raise additional concerns for our scenario. To avoid overloading our notations, the timestamp is not explicitly mentioned in the signatures.

C. PROTOCOL STEPS

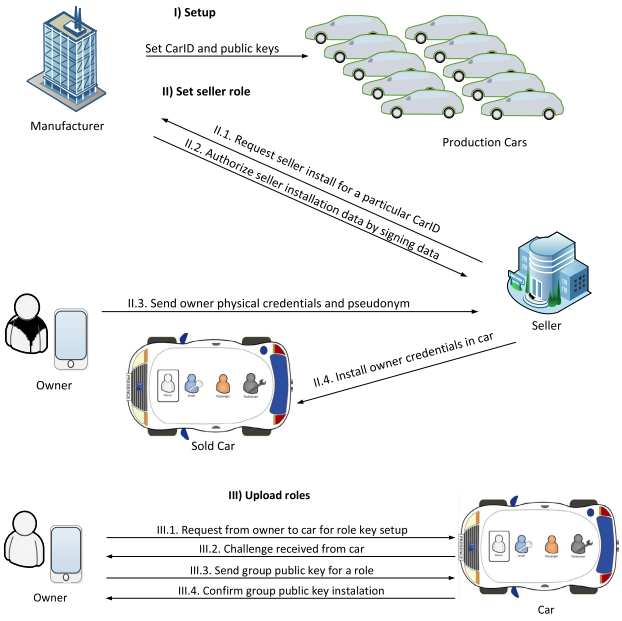
The procedures required by the protocol are discussed next.

1) CAR SETUP AT THE MANUFACTURER

The procedures for setting up the car and installing the root (which is the owner) are graphically depicted in Figure 5. We consider that these steps are done in a secure environment. Generally, this should be the case since if the production environment is insecure then the software on the ECUs may already be altered which may have more disastrous consequences. Still, if this is not the case, secure channels can be introduced during setup but this is out-of-scope for our work. We assume that during production, the manufacturer installs the secret key of the car, i.e., sk_{car}, inside each car. Each car also has a unique identifier CarID and the manufacturer is also responsible for installing the roles and rights which are expressed as a vector product $\overline{\mathsf{Role}} \times \overline{\mathsf{Rights}}$. Since we rely on identity-based signatures, all public keys will be derived by the car from the identities of the principals with which it interacts. Any other public parameters related to the identity-based cryptographic schemes will be installed in the car at this stage, i.e., step 1.

2) SETTING THE ROOT OWNER DURING CAR PURCHASE

The manufacturer is also responsible for giving rights to the seller as expressed in Protocol II from Figure 5. This happens by simply signing an installation message containing the identity of the seller and the identity of the car, i.e., $m_{mnf} = \{\mathsf{act{:}sel}, \mathsf{Sel}, \mathsf{CarID}\}$. Both the request of the seller and the confirmation from the manufacturer are signed using an identity-based scheme, i.e., $s_{sel} = \mathsf{IdSig}(\mathsf{sk}_{sel}, m_{sel})$, $s_{mnf} = \mathsf{IdSig}(\mathsf{sk}_{mnf}, m_{mnf})$. When the car is purchased, the owner first presents the owner data (these are physical credentials as a passport or identification card, etc.). Due to legal issues it does not seem viable to hide driver's personal information from the seller, thus the owner has to present his physical credentials in some way. We assume that the seller is trustworthy and keeps the confidentiality of the new owner. The owner choses and presents a pseudonym PsO and a public key PkO and fixes the start of the contract as T_{start}. The life-time of the purchasing contract is set to ∞ which seems a natural choice but can be changed according to practical needs. We use pseudonyms to assure driver's privacy in front of the car manufacturer. The seller Sel verifies the legal information and if all the criteria are met, it installs the owner data inside the car by sending the owner request message m_{own} with its signature s_{sel}. For simplicity we also included here the messages for setting the identity of the seller inside the car, i.e., m_{mnf}, s_{mnf}, but these can be set as well at some previous stage. We assume that the seller installs the data received from the manufacturer in a secure manner inside the car (the secure channel is suggested by the double arrow $\twoheadrightarrow$). If the setup needs to be done via an insecure port, such as OBD, we assume that this is done in a secure environment, e.g., an authorized garage.



I) Setup (by manufacturer)

1. Man $\twoheadrightarrow$ Car: $\mathsf{Man}, \mathsf{CarID}, \mathsf{sk}_{car}, \overline{\mathsf{Role}} \times \mathsf{Atr} \times \overline{\mathsf{Rights}}\}$

II) Set root (by seller & manufacturer)

1. Sel $\twoheadrightarrow$ Man: $m_{sel} = \{\mathsf{Sel}, \mathsf{Man}, \mathsf{CarID}\}$, $s_{sel} = \mathsf{IdSig}(\mathsf{sk}_{sel}, m_{sel})$
2. Man $\twoheadrightarrow$ Sel: $m_{mnf} = \{\mathsf{act{:}sel}, \mathsf{Sel}, \mathsf{CarID}\}$, $s_{mnf} = \mathsf{IdSig}(\mathsf{sk}_{mnf}, m_{mnf})$
3. Own $\twoheadrightarrow$ Sel: $owner\ data, m_{own} = \{\mathsf{PsO}, T_{start}, \infty\}$
4. Sel $\twoheadrightarrow$ Car: $m_{own}, s_{sel} = \mathsf{IdSig}(\mathsf{sk}_{sel}, m_{own}), m_{mnf}, s_{mnf}$

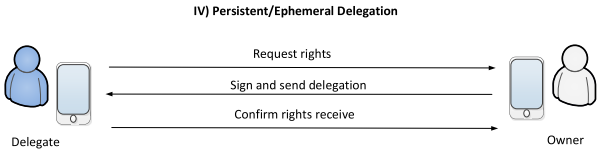
III) Upload role public keys (by root only)

1. Own $\rightarrow$ Car: $m'_{own} = \{\mathsf{N}_{own}, \mathsf{Own}, \mathsf{CarID}\}$, $s'_{own} = \mathsf{IdSig}(\mathsf{sk}_{own}, m'_{own})$
2. Car $\rightarrow$ Own: $m'_{car} = \{\mathsf{N}_{own}, \mathsf{N}_{car}\}, s'_{car} = \mathsf{IdSig}(\mathsf{sk}_{car}, m'_{car})$
3. Own $\rightarrow$ Car: $m''_{own} = \{\mathsf{N}_{own}, \mathsf{N}_{car}, \mathsf{Role}, \mathsf{gpk}, T_{start}, \infty\}$, $s''_{own} = \mathsf{IdSig}(\mathsf{sk}_{own}, m_{own})$
4. Car $\rightarrow$ Own: $m''_{car} = \{\mathsf{act{:}conf}, m''_{own}\}, s''_{car} = \mathsf{IdSig}(\mathsf{sk}_{car}, m''_{car})$

FIGURE 5. Protocol procedures for car setup and group key upload.

3) SETTING GROUP PUBLIC KEYS

This procedure is graphically suggested in Protocol III from Figure 5. The owner is the only entity entitled to add group public keys. He starts a challenge-response interaction with the car by sending a message with a nonce N_{own}, his identity Own and the car CarID. This message is signed with an identity-based scheme as $s'_{own} = \mathsf{IdSig}(\mathsf{sk}_{own}, m'_{own})$. The car replies with a message containing its own nonce N_{car} and this message is as well signed as $s'_{car} = \mathsf{IdSig}(\mathsf{sk}_{car}, m'_{car})$.



IV) Persistent or ephemeral delegation, i.e., $\mathsf{Usr} \in \{\mathsf{Del}, \widetilde{\mathsf{Del}}\}$

1. $\mathsf{Usr} \rightarrow \mathsf{Own}: m'_{\mathsf{usr}} = \{\mathsf{PsU}, \mathsf{Role}, \mathsf{Atr}, T_{start}, T_{stop}\},$
$m''_{\mathsf{usr}} = \{\mathsf{pk}_{\mathsf{usr}}, \mathsf{N}_{\mathsf{usr}}\}, s_{\mathsf{usr}} = \mathsf{IdSig}(\mathsf{sk}_{\mathsf{usr}}, m'_{\mathsf{usr}}, m''_{\mathsf{usr}})$
2. $\mathsf{Own} \rightarrow \mathsf{Usr}: m'_{own} = \{\mathsf{N}_{own}, \mathsf{N}_{\mathsf{usr}}, \{atk\}_{\mathrm{K}_{ses}}, \lfloor s_{\mathsf{usr}} \rfloor_{64}\},$
$s'_{own} = \mathsf{GrSig}(\mathsf{sk}_{own}, m'_{own})$
$atk = \begin{cases} \mathsf{sk[Role]} \ if \ \mathsf{Usr} = \mathsf{Del} \\ s''_{own} = \mathsf{GrSig}(\mathsf{sk}_{own}, m'_{\mathsf{usr}}) \ if \ \mathsf{Usr} = \widetilde{\mathsf{Del}} \end{cases}$
3. $\mathsf{Usr} \rightarrow \mathsf{Own}: m'''_{\mathsf{usr}} = \mathsf{IdSig}(\mathsf{sk}_{\mathsf{usr}}, \{m'_{\mathsf{usr}}, m'_{own}\})$
4. $\mathsf{Own} \rightarrow \mathsf{Usr}: m''_{own} = \{\mathrm{K}_{ses}\}_{\mathsf{pk}_{\mathsf{usr}}}$

FIGURE 6. Protocol procedures for persistent and ephemeral rights delegation.

The owner responds in m''_{own} by including both the nonce that he sent and the one received from the car, the message also contains the name of the role Role and the group public-key gpk along with start time T_{start} and validity period (we consider that validity is indefinite ∞ since the roles are persistent but this can be changed according to specific needs). The car confirms that the group public-key of the role was installed by signing message m''_{own} and a confirmation tag act:conf.

4) RIGHTS DELEGATION SCENARIO

Both the persistent and ephemeral rights delegation procedures are suggested in Figure 6, i.e., $\mathsf{Usr} \in \{\mathsf{Del}, \widetilde{\mathsf{Del}}\}$. The distinction between the two cases is in step 2 where the content of the authentication token *atk* is different between the two. A persistent user will receive a group public key while ephemeral users will receive a signed proof of their execution rights. For the later case, the delegation holds for a specified amount of time between T_{start} and T_{stop}. We now describe the rest of the protocol which is identical regardless of the case. The user Usr makes a request to the owner by sending his pseudonym PsU, an ephemeral public-key $\mathsf{pk}_{\mathsf{usr}}$, a nonce $\mathsf{N}_{\mathsf{usr}}$ and the role Role and attributes Atr for which he requests the rights. The message is signed by the user with an identity-based signature, i.e., $s_{\mathsf{usr}} = \mathsf{IdSig}(\mathsf{sk}_{\mathsf{usr}}, m_{\mathsf{usr}})$. The owner replies with a message containing a nonce N_{own}, the encrypted authentication token $\{atk\}_{\mathrm{K}_{ses}}$ and a truncated value from the signature of the previous message, i.e., $\lfloor s_{\mathsf{usr}} \rfloor_{64}$. The message also contains the signature of the owner which is a group signature for the group that he is part of, i.e., $s_{own} = \mathsf{GrSig}(\mathsf{sk}_{own}, m_{own})$. The reason for encrypting the authentication token and not yet disclosing the key is for the confirmation that the token was received in the next step. Otherwise, a delegated user may claim that he did not receive the token. In the next step the user confirms this with an identity-based signature on both previous messages, i.e., $\mathsf{IdSig}(\mathsf{sk}_{\mathsf{usr}}, \{m'_{\mathsf{usr}}, m'_{own}\})$. Now the owner discloses the session key K_{ses} in an encrypted manner such that only the user (which has the ephemeral public-key) can decrypt in the last message $\{\mathrm{K}_{ses}\}_{\mathsf{pk}_{\mathsf{usr}}}$. The authentication token will be used in the next procedure for executing functionalities on the car. The ephemeral public-key $\mathsf{pk}_{\mathsf{usr}}$ can be a regular RSA public-key or, if a more compact representation is desired, an elliptical curve Diffie-Hellman session key, i.e., the scalar multiplication of an elliptical curve point aP. For the later case, the owner will reply in the last message with the corresponding key share, i.e., bP and the session key K_{ses} is extracted from the common Diffie-Hellman key abP. In the experimental section we show the differences in terms of computational costs, but both the Diffie-Hellman and RSA operations are in the order of dozen milliseconds which is affordable for modern Android devices.

5) EXECUTE FUNCTIONALITY

Procedures for triggering the execution of a functionality inside the car are graphically suggested in Figure 7. The execute scenario calls for two distinct procedures that achieve the same goal. The first version relies on asymmetric primitives while the second (on-the-fly) relies on a session key and symmetric primitives alone. Obviously, the first procedure is more expensive and we assume that once a session key is established, only the second (faster) procedure is to be invoked. First, the user playing either a persistent or ephemeral user, i.e., $\mathsf{Usr} \in \{\mathsf{Del}, \widetilde{\mathsf{Del}}\}$, sends a message m'_{usr} presenting his role and attributes along with a nonce $\mathsf{N}_{\mathsf{usr}}$ to assure freshness. This message comes along with signature s'_{usr} which is either a group signature (in case of persistent users) or an identity-based signature (in case of ephemeral users). For the later case, the user will also present the credentials based on which he acquired his rights from the owner, i.e., $s''^{,iv}_{own}, m'^{,iv}_{\mathsf{usr}}$ (these were received during the delegation procedure). The car replies as challenge with an ephemeral public-key pk_{car}, a nonce N_{car}, a session identifier SID included in m_{car}. These are signed by the car with an identity-based signature and presented as s_{car}. The message m_{car} also includes a 64 bit truncation of the original signature from the user, i.e, $\lfloor s_{\mathsf{usr}} \rfloor_{64}$. Now the user generates a session key K_{ses} which is encrypted with the ephemeral public-key pk_{car}. The user presents the desired action on the car as an encrypted message, i.e., $\{\mathsf{act{:}exec}[i], \lfloor s_{car} \rfloor_{64}\}_{\mathrm{K}_{ses}}$ (note that this message also includes the last 64 bits of the car signature, i.e., $\lfloor s_{car} \rfloor_{64}$). The message is accompanied by a regular MAC, i.e, s''_{usr}, which is performed with the session key K_{ses}. Similar to the previous protocol fragment, the ephemeral public-key pk_{car} can be a regular RSA public-key or a more compact Diffie-Hellman session key. In the later case pk_{car} is replaced by aP and in the last message $\{\mathrm{K}_{ses}\}_{\mathsf{pk}_{car}}$ is replaced by bP from which the common session key abP is extracted.

For the on-the-fly version of the protocol, the signatures are replaced with symmetric key Message Authentication Codes (MACs). The session identifier SID links the

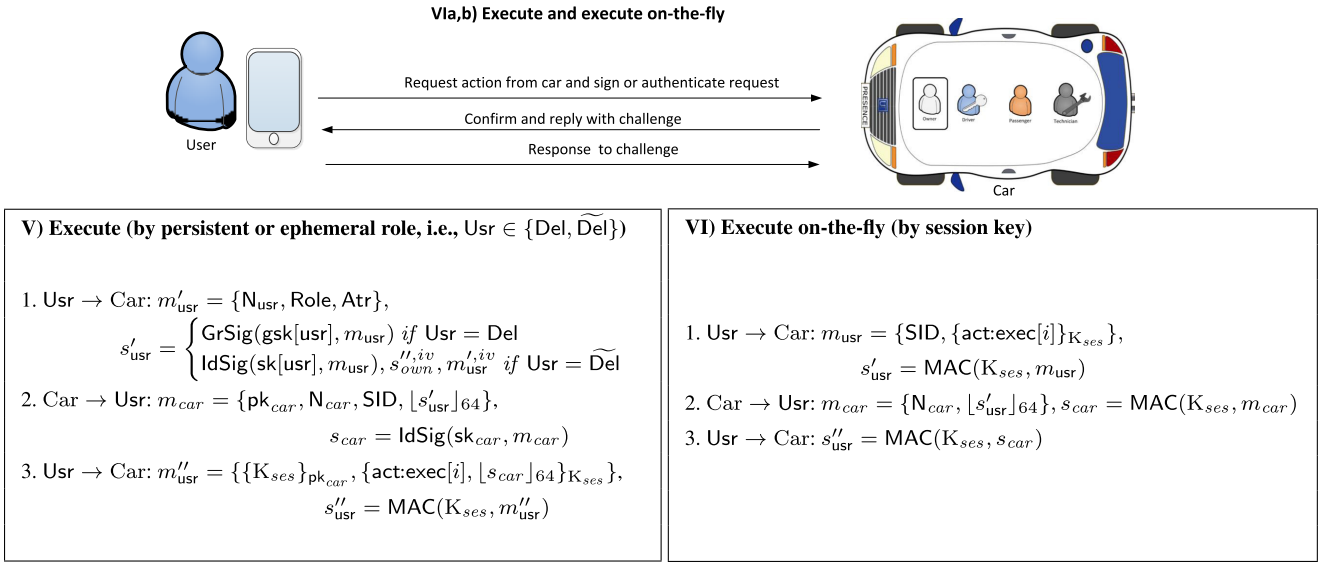


FIGURE 7. Protocol procedures for execution and on-the-fly execution.

on-the-fly execution with the session key K_{ses} from the previous procedure (the life-time of this session key can be hours, or days, depending on the practical circumstances). The user presents his request $\mathsf{act{:}exec}[i]$ encrypted with the session key K_{ses} and authenticated with a MAC, i.e., $s'_{\mathsf{usr}} = \mathsf{MAC}(\mathrm{K}_{ses}, m_{\mathsf{usr}})$. The car replies with a nonce N_{car} and this message along with the truncated value of s'_{usr} is also authenticated by a MAC in s_{car}. Finally, the user answers to this challenge with s''_{usr} which is a MAC computed on the previous message with the session key K_{ses}.

We do not present additional procedures for *rights revocation*. All of the included asymmetric primitives have well-known revocation mechanisms. This includes the group signature in [10] for which the procedure is less obvious. While deploying such revocation mechanisms is not straightforward, e.g., the car needs to keep a revocation list and update it accordingly, adding more details here is out of scope for our work. We also do not insist on other technicalities such as how to revoke owners or how to facilitate the resale of the car since adding protocol fragments is easy for any such action but will contribute little to the main concept from this work.

D. ADVERSARY MODEL AND SECURITY ARGUMENTS

As adversary model, we consider the general Dolev and Yao [21] intruder that has full control over the communication channel, i.e., he can eavesdrop, replay, inject or modify existing packets. Specific attacks related to the software implementation are out of scope for the current analysis. But as for future, more practical embodiments of our work, the use of specific Android security mechanisms or relying on hardware security, e.g., TPM 2.0 functions, may be projected.

Since we rely on existing cryptographic blocks that are assumed to be secure, rather than deriving more complicate cryptographic proofs, we consider that a proof by formal analysis (model-checking) offers better support for the security of our protocol. We choose to rely on the IF language for modelling which is the base language for the three model-checkers of the AVISPA platform [3]. In particular, we choose to rely on the CLAtse model-checker [55] from the AVISPA platform. Model-checkers assume the underlying cryptographic blocks to be perfect and model the intruder as a Dolev-Yao adversary. For brevity, we choose to model the last 2 protocol fragments V) and VI), i.e., execute (by persistent or ephemeral role) and execute on-the-fly (by session key) since these are the actual protocol components that grant access to the car. Modelling the entire protocol would require a large amount of work and would be out-of-scope for the technological readiness level that we target in the current work, i.e., proof-of-concept. By using the IF language of the AVISPA platform [3], we model each protocol step as a transition from the left-hand side (LHS) facts to the right-hand side (RHS) facts. The LHS and RHS are conjunctions of positive and negative facts and are not persistent (the RHS suppresses the LHS). The Dolev-Yao adversary is modelled by the *iknows* predicate which cumulates facts in a persistent manner, i.e., the intruder never forgets what he learns.

The first model that we analyze is the simple on-the-fly execution. We defined two actions, i.e., *open car* and *start car*, and ask the model checker if it can produce a trace that can trigger a *start* of the car given that the honest user is set to *open* the car. This covers the scenario in which the adversary can manipulate the commands of the genuine user. The model-checker answered that the protocol is safe. To test the correctness of our model we also added the session key K_{ses} to the intruder knowledge a case in which the model-checker immediately returned the attack (this proves that if the session key would have been leaked by any mean,

the intruder would have been able to start the car). We also checked the consistency of the model by verifying that the genuine user can set the car in the *open* open state which proved to be correct.

Figure 8 shows the trace output by the model-checker when we tested that the genuine user can open the car (in case of the adversary attack, there is no trace since an attack cannot be found and the model is reported as safe). The output trace shows that the intruder *i* mediates the communication channel by intercepting all messages. Compound messages are built with the *pair* operator and symmetric encryption is modeled by the *scrypt* predicate. Finally, the car reaches the state *state_car(2,usr,sid,kses,open,n4(NC))* which means that the user *usr* managed to *open* the car under session key *kses* and a random challenge nonce *n4(NC)* which is generated as a fresh symbolic term by the model checker.

We then proceed to the analysis of the execute procedure by persistent or ephemeral roles. The AVISPA toolset does not offer specific support for identity-based signatures or group signatures but formally speaking they do not differ in terms of the signing/verification procedures from regular signatures (the differences are in how the keys are derived and linked to an identity). In the IF language, a signature is modelled as encryption with the inverse of the public-key (similar to the RSA mechanism). For example, the response of the car in step 2 of protocol v) is symbolically expressed as *iknows(crypt(inv(PkCar), pair(PkCarE, pair(NC, pair(SID, crypt(inv(PkUsr), pair(NU, pair(Role, Atr))))))))*. Here, *crypt(inv(PkCar),_)* denotes the signature of the car on the message. We conducted similar tests as in the case of the on-the-fly procedure and the protocol proved to be safe.

Another type of attack at the protocol level that is worth considering is privilege escalation by which an attacker with certain privileges may try to perform an operation which he is not entitled to perform. This may include a manufacturer that tries to de-anonymize the users or a passenger that tries to achieve driver rights on the car. The only possibility to de-anonymize users is by using the traceability functionality of the signature designed by Boneh *et al.* [10] which would require access to the group manager secret key. Our implementation delegates this capability to the car owner, emphasizing on the ownership rights (other implementations may delegate this to a trusted authority). Currently, the scheme of Boneh *et al.* [10] is considered secure, so it would be out-of-scope for this work to bring a new proof that an adversary may not perform this attack. The same remark is available for the suggested privilege escalation by a passenger as he will not be able to generate a signature for a role with higher privileges as long as the group signature scheme is secure.

There are of course many other side-channels from which the identity of drivers may be inferred. For example, driving patterns such as driving time or specific driving behaviors that can be recorded from on-device sensors can be used to infer the driver's identity. Recent research has proved that accelerometer data can be used for this purpose,

```
i -> (car,4):  pair()
(car,4) -> i:  pair(n4(NC),
           pair(scrypt(kses,pair(sid,open)),
           scrypt(kses,pair(n4(NC),
           scrypt(kses,pair(sid,open))))))
           & Remove state_car(0,usr,sid,kses,open,nc);
           & Add state_car(1,usr,sid,kses,open,n4(NC));
           & Built from trans2

i -> (usr,3):  pair()
(usr,3) -> i:  pair(sid,pair(scrypt(kses,open),
           scrypt(kses,pair(sid,scrypt(kses,open)))))
           & Remove state_usr(0,usr,sid,kses,open,nc);
           & Add state_usr(1,usr,sid,kses,open,nc);
           & Built from trans1

i -> (usr,5):  pair(scrypt(kses,pair(n4(NC),
           scrypt(kses,pair(sid,open)))),
           pair(n4(NC),scrypt(kses,pair(sid,open))))
(usr,5) -> i:  scrypt(kses,scrypt(kses,pair(n4(NC),
           scrypt(kses,pair(sid,open)))))
           & Remove state_usr(1,usr,sid,kses,open,nc);
           & Add state_usr(2,usr,sid,kses,open,n4(NC));
           & Built from trans3

i -> (car,6):  scrypt(kses,scrypt(kses,pair(n4(NC),
           scrypt(kses,pair(sid,open)))))
(car,6) -> i:  pair()
         & Remove state_car(1,usr,sid,kses,open,n4(NC));
         & Add state_car(2,usr,sid,kses,open,n4(NC));
         & Built from trans4
```

FIGURE 8. Output trace for regular user connection to the car.

e.g., [22], [44], [57]. It is however out-of-scope for the current research to address all these possible leakages as our work is focused on protocol design and implementation only. For a fully secure solution, one will need Android devices to offer resilience to such leakages.

III. EXPERIMENTS

In this section we discuss experimental results on Android phones and Infotainment units as well as on automotive-grade controllers. We discuss both computational requirements for some of the cryptographic primitives that we use as well as the protocol running time for several of the procedures that we previously described.

A. ANDROID AND VEHICLE ON-BOARD UNIT IMPLEMENTATION

Figure 9 depicts the experimental setup with the after-market Android headunits on which we deployed our implementation.

We have implemented in Android Studio the last three procedures of our protocol: persistent and ephemeral delegation, execute and execute on-the-fly. We tried to keep our implementation as simple and scalable as possible. Therefore, the protocol implementation relies on a simple finite state machine. The state machine consists of three states for each of the execute and execute on-the-fly procedure, and of four states for the delegation procedure, adhering to the previous formal protocol description. The states correspond to the messages that are exchanged during the procedures. For the group signature scheme we have used the Pairings_in_C library[3] [56]. The library also contains an Android Demo

[3]https://github.com/IAIK/pairings_in_c

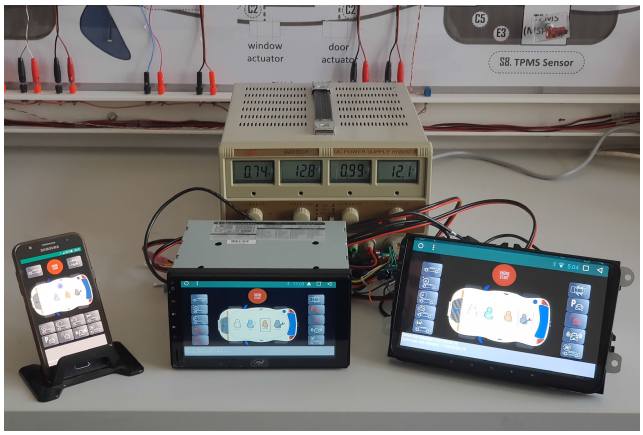


FIGURE 9. Android headunits from our experiments.

that implements the group signature by Boneh *et al.* [10], which was easily adapted and integrated in our protocol. The identity-based signature scheme used in our protocol was independently implemented by us, while the rest of the cryptographic functions are from standard Java libraries.

For the NFC communication, we have used the NFC card reader and NFC card emulation modes. As basis for our NFC implementation we used various samples from Android CardReader and Android CardEmulation Sample provided by Google.[4] The payload size of the NFC frames that were transmitted between the device running in card reader mode and the device running in card emulation mode was 254 bytes. Hence, the messages exchanged in the implemented procedures were divided in several NFC frames.

Wi-Fi communication is based on TCP IP and we used two sockets, a server socket that listens the incoming connections requests and a client socket that initializes the connection. In our application, the smartphone is configured as a client and the head unit is configured as a server. The headunit also plays the role of the access point and mobile-phones connect directly to it.

For our experimental evaluation the on-board unit is represented by the TC297 microcontroller clocked at 300MHz and equipped with 8MB of flash and 728KB of RAM. We evaluated the computational performance of the TC297 by measuring the execution time for the basic building blocks of our protocol. We base our implementations on the Miracl (Multiprecision Integer and Rational Arithmetic Cryptographic Library) library.[5]

B. RESULTS

The computational time of the required cryptographic primitives on several platforms is summarized in Table 4. The Shamir and GQ signature implementation was developed by us in C++ and Java for the Infineon controller and Android devices. The rest of the cryptographic functions come from the aforementioned libraries.

[4]https://github.com/googlesamples/
[5]https://github.com/miracl/MIRACL

On the Android devices, the computational time for the Group Signature (GS) is the highest along with the 2048-bit version of the GQ signature and tops between 250-500*ms*. For the GQ signature, it may be that our implementation can be further optimized but the speed difference is clearly in favour of using Shamir IBS. For setting the security parameters of the Shamir signature, we followed the recommendations in [8] which point to a 1024-bit modulus with a 160-bit hash functions, which we extend to a 2048-bit modulus with a 256-bit hash function that should be appropriate for current needs. For the Tricore in-vehicle unit the execution time becomes unacceptable with the 2048-bit version of GQ and since a 1024-bit modulus would lower the security level we find that using Shamir IBS is the only viable option. The cryptographic libraries that we use on Infineon had no platform-specific optimizations. The GS and the 2048-bit version of GQ have similar run-times on Android while there is a bigger computational gap between the two when executed on the Infineon controller. This suggests that the C++ code for the Infineon platform can be further optimized to obtain similar performances. As the IBS and GS are used less often, the protocol should cope for a real-world car access scenario. We assume that the on-the-fly execution, which relies only on symmetric-key cryptography is the regular way to access the car while the identity/group-based execution is only triggered once to establish a session key. The RSA has a shorter runtime than the GS and GQ, but of course these traditional building blocks do not offer the advantages of group or identity-based signatures. We also include results regarding ECDH, the time to generate the key-pair (a, aP) denoted as *GenKP* and the time to generate the secret key abP denoted as *GenSK*, to serve as a comparison to RSA. For this purpose we use regular Android cryptographic libraries from Spongy Castle, while for the Infineon controller we consider results from our previous work in [47]. The runtime is in general comparable with that of the RSA, though not surprising RSA encryption is still the fastest. On the embedded controller the results for RSA were somewhat poorer and given the larger key size it becomes somewhat clear that ECDH would be more suitable in this case.

In Table 5 we summarize the complete run-time for several protocol procedures run between smartphones and head-units. These protocol fragments are tested over the three interfaces NFC, Bluetooth and WiFi. Since our car head-units did not support regular data transfer over Bluetooth, in this case we tested the execution only between two smartphones. However, the performance should be close to the case when a head-unit is used. Sharing rights is done over NFC due to increased security as it works on a shorter range and is harder to spoof. The request for execution to the car head-unit is done over WiFi. The execution runtime is around 1 second (and generally less than 1 second with the more efficient Shamir IBS), which should be sufficiently fast, assuming that only the first execution is done with the slower group or identity-based signatures. The rest of the executions are carried by the on-the-fly procedure taking only a few hundred milliseconds

TABLE 4. Computational time for cryptographic primitives on selected platforms (ms).

Device \ Primitive	GS (254 bit)		Shamir (2048 bit)		GQ (2048 bit)		ECDH (192 bit)		ECDH (256 bit)		RSA (2048 bit)	
	Sign	Ver	Sign	Ver	Sign	Ver	GenKP	GenSK	GenKP	GenSK	Enc	Dec
LG Optimus P700	259.21	364.35	49.26	46.49	218.50	424.80	49.66	26.39	83.96	56.88	2.15	52.59
Samsung S7	24.12	33.15	7.79	8.02	27.72	57.18	15.15	13.81	46.37	15.17	0.24	8.80
Samsung J5 SM-J500F	158.73	226.71	18.31	16.21	70.76	139.12	13.58	5.95	20.17	10.39	0.60	17.07
Samsung S5	68.09	96.54	23.26	22.59	102.98	203.25	17.65	8.57	27.86	15.08	0.78	24.91
Headunit 1(Erisin)	181.39	261.60	28.20	25.47	113.10	220.30	14.68	6.44	20.35	10.43	0.97	26.86
Headunit 2(PNI)	176.33	254.08	27.35	24.70	111.60	218.70	14.23	6.24	19.73	10.11	0.96	26.28
Tricore Tc297	511.60	745.60	448.00	448.00	2000.00	4120.00	36.40	37.40	59.80	69.10	26.00	462.00

TABLE 5. Computational/communication time for protocol procedures (ms).

Devices	Com.	Execute Persistent		Execute Ephemeral		Exec.	Delegate Persistent		Delegate Ephemeral	
		Shamir IBS	GQ IBS	Shamir IBS	GQ IBS	OTF	Shamir IBS	GQ IBS	Shamir IBS	GQ IBS
Phone2Phone(S5-S7)	NFC	–	–	–	–	–	998.64	1256.42	1165.64	1423.42
S7-Headunit 1(Erisin)	WiFi	758.48	892.55	923.87	1272.70	516.00	–	–	–	–
S7-Headunit 2(PNI)	WiFi	718.50	851.92	1127.15	1474.49	451.00	–	–	–	–
Phone2Phone (S5-S7)	BT	366.83	567.42	193.94	523.42	115.00	–	–	–	–
J5-Headunit 1(Erisin)	WiFi	771.85	1019.13	748.63	1203.72	381.80	–	–	–	–
J5-Headunit 2(PNI)	WiFi	758.65	965.81	668.01	1121.61	274.50	–	–	–	–

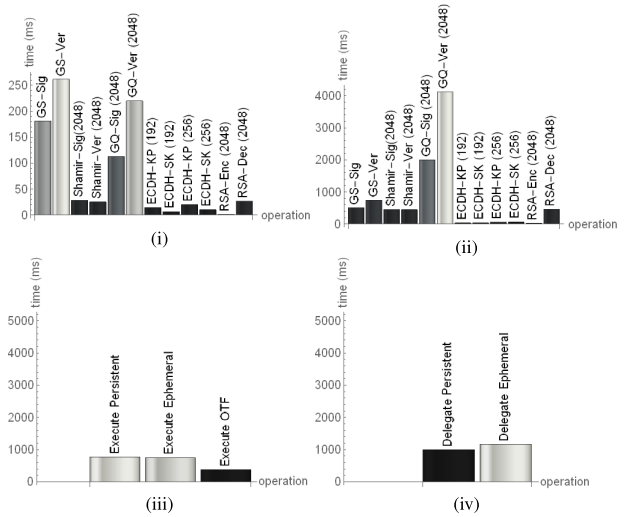


FIGURE 10. Execution time for: signatures on the ERISIN headunit (i) and Infineon Tricore in-vehicle board (ii), execute persistent/ephemeral (Shamir IBS) J5-Headunit 1(ERISIN) over WiFi (iii) and delegate (Shamir IBS) S5-S7 over NFC (iv).

(since a common secret shared key exists). Figure 10 summarizes in a graphic form on some of the computational times from Tables 4 and 5.

IV. CONCLUSION

The increased computational power of modern smartphones and their generous user-interface facilitates the implementation of various car access control functionalities and more exquisite protocols with advanced functionalities. These can benefit from state-of-the-art cryptographic building blocks such as identity-based cryptography or group signatures. While some of these require more computational power or build upon more expensive pairing-friendly elliptical curves, computational capabilities of modern smartphones and of high-end in-vehicle units are satisfactory for handling them. The provided experimental results prove that adoption is possible both on modern smartphones as well as on modern in-vehicle controllers, e.g., an Infineon TriCore car controller. At a minimum, the RBAC access control policy for car functionalities is within reach for most of the in-vehicle units on the market. With this research we hope to pave the way for addressing both security and privacy in car access control scenarios. Further improvements may consist in adding specialized hardware such as Trusted Platform Modules (TPM) or relying on trusted execution environment such as ARM TrustZone that already exists on some mobile phones. Nonetheless, porting functionalities to wearable devices such as smart-watches or smart-glasses may also increase the usability of the solution. We leave these as potential directions for future works.

APPENDIX
DESCRIPTION OF THE SHAMIR AND GUILLOU-QUISQUATER IDENTITY-BASED SIGNATURE SCHEMES

Since the native Android cryptographic libraries offer no support for the Shamir and Guillou-Quisquater identity-based signature schemes, we had to implement these separately (the source-code will be maintained on our project website). To clarify the algorithms we give their description in the syntax introduced in Section II.B.

The identity-based signature scheme proposed by Shamir [49] consists in the following four algorithms:

1) $\mathsf{Setup}(k)$ is the key setup algorithm which outputs the master secret key msk and the global parameters pk. For this, it generates two random primes p, q of k bits in length, computes $n = pq$, $\phi(n) = (p-1)(q-1)$, sets integer $e \in Z_{\phi(n)}$ s.t. $\gcd(e, \phi(n)) = 1$ then computes $d = e^{-1} \mod \phi(n)$. The master secret key is $\mathsf{msk} =$

$\{n, d\}$ and the global public key is $\mathsf{pk} = \{n, e, h\}$. Here h stands for a hash function that maps the user name to an element of $Z_{\phi(n)}$, i.e., $h : \{0, 1\}^* \rightarrow Z_{\phi(n)}$.

2) $\mathsf{KeyDer}(\mathsf{msk}, I)$ uses the master secret key msk and the identity of the user I to output the private key of the user. For this, it computes $h(I)^d \mod n$ and returns to each user the secret key $\mathsf{sk} = \{h(I)^d \mod n, n\}$ (the public key of each user is his identity, i.e., I).
3) $\mathsf{Sign}(\mathsf{sk}, m)$ is the signature generation algorithm which uses the secret key sk on the message m to return the signature σ. For this, it selects random $r \in Z_n$, computes $t = r^e \mod n$, then the hash of t concatenated with message m, i.e., $h = hash(t||m)$, then $s = h(I)^d r^h \mod n$. The signature is $\sigma = \{s, t\}$.
4) $\mathsf{Ver}(\mathsf{pk}, I, m, \sigma)$ takes as input the system global parameters pk, the identity of the user I, the message m and the signature σ. To verify that the signature is correct the algorithm computes s^e then checks if this is equal to $h(I)t^h \mod n$ and returns *true* if so or $\perp$ otherwise.

The Guillou and Quisquater [30] identity-based signature scheme is a collection of four algorithms:

1) $\mathsf{Setup}(k)$ is the key setup algorithm that generates the master secret key msk and the public key pk. In case of the GQ algorithm, the Setup algorithm, generates two random primes p, q, each having k bits in length, it selects random integer $v \in Z_n, n = pq$, computes $\phi(n) = (p-1)(q-1)$ and $v^{-1} \mod \phi(n)$. The master secret key is $\mathsf{msk} = \{n, v^{-1} \mod \phi(n)\}$ and the public key is $\mathsf{pk} = \{n, v\}$.
2) $\mathsf{KeyDer}(\mathsf{msk}, I)$ is the key derivation algorithm that uses the master secret key msk and the identity of the user I to generate his private key. In case of the GQ algorithm, the identity I of a principal is mapped (by a publicly known redundancy function) to a number $J \in Z_n$ then the algorithm computes $B = J^{-v^{-1}} \mod n$. The user secret key is $\mathsf{sk} = \{B, J, v, n\}$ (since this is an identity-based scheme, the public key to verify the signatures of this user is pk and the identity of the user I).
3) $\mathsf{Sign}(\mathsf{sk}, m)$ is the signature algorithm that takes as input the user's secret key sk and a message m then returns the signature σ. The GQ signing algorithm selects a random $r \in Z_n$, computes $T = r^v modn$, the hash of message m denoted as h, then $d = J^h T^{v^l} \mod n$ and $t = rB^d \mod n$ where l is an integer such that $v^l < m < v^{l+1}$. The signature is $\sigma = \{d, t\}$.
4) $\mathsf{Ver}(\mathsf{pk}, m, \sigma)$ is the verification algorithm which takes as input the public-key pk, the message m and the signature σ and returns *true* if the signature is correct otherwise it returns $\perp$. To verify that the signature is correct, the algorithm derives J from the identity I, computes $T' = J^d T^v \mod n$, computes the hash h of messages m and $d' = J^h T'^{v^l} \mod n$, then the verifier $d'' = J^{h+dv^l} t^{v^{l+1}} \mod n$ and checks if $d' = d''$ then returns *true* if so or $\perp$ otherwise.

REFERENCES

[1] M. A. Albahar, K. Haataja, and P. Toivanen, "Bluetooth mitm vulnerabilities: A literature review, novel attack scenarios, novel countermeasures, and lessons learned," *Int. J. Inf. Technol. Secur.*, vol. 8, no. 4, pp. 25–49, 2016.
[2] G. Alli, L. Baresi, A. Bianchessi, G. Cugola, A. Margara, A. Morzenti, C. Ongini, E. Panigati, M. Rossi, S. Rotondi, S. Savaresi, F. A. Schreiber, A. Sivieri, L. Tanca, and E. V. Depoli, "Green move: Towards next generation sustainable smartphone-based vehicle sharing," in *Proc. Sustain. Internet ICT Sustainability (SustainIT)*, Oct. 2012, pp. 1–5.
[3] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron "The AVISPA tool for the automated validation of Internet security protocols and applications," in *Proc. Int. Conf. Comput. Aided Verification*. Cham, Switzerland: Springer, 2005, pp. 281–285.
[4] *Specification of Crypto Abstraction Library, 4.2.2 Edition*, AUTOSAR, Munich, Germany.
[5] *Specification Crypto Service Manager, 4.2.2 Edition*, AUTOSAR, Munich, Germany, 2015.
[6] L. Bauer, L. F. Cranor, R. W. Reeder, M. K. Reiter, and K. Vaniea, "Comparing access-control technologies: A study of keys and smartphones," Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CYLAB-07-005, 2007.
[7] L. Bauer, L. F. Cranor, M. K. Reiter, and K. Vaniea, "Lessons learned from the deployment of a smartphone-based access-control system," in *Proc. 3rd Symp. Usable Privacy Secur. (SOUPS)*, 2007, pp. 64–75.
[8] M. Bellare and G. Neven, "Identity-based multi-signatures from RSA," in *Proc. Cryptographers' Track RSA Conf.* Cham, Switzerland: Springer, 2007, pp. 145–162.
[9] E. Biham and L. Neumann, "Breaking the Bluetooth pairing—The fixed coordinate invalid curve attack," in *Selected Areas in Cryptography—SAC* (Lecture Notes in Computer Science), vol. 11959, K. G. Paterson and D. Stebila, Eds. Waterloo, ON, Canada: Springer, Aug. 2019, pp. 250–273.
[10] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2004, pp. 41–55.
[11] R. Bose, J. Brakensiek, K.-Y. Park, and J. Lester, "Morphing smartphones into automotive application platforms," *Computer*, vol. 44, no. 5, pp. 53–61, May 2011.
[12] C. Busold, A. Taha, C. Wachsmann, A. Dmitrienko, H. Seudié, M. Sobhani, and A.-R. Sadeghi, "Smart keys for cyber-cars: Secure smartphone-based NFC-enabled car immobilizer," in *Proc. 3rd ACM Conf. Data Appl. Secur. Privacy (CODASPY)*, 2013, pp. 233–242.
[13] D. W. Chadwick, A. Otenko, and E. Ball, "Role-based access control with X.509 attribute certificates," *IEEE Internet Comput.*, vol. 7, no. 2, pp. 62–69, Mar. 2003.
[14] S. Chakraborty and I. Ray, "TrustBAC: Integrating trust relationships into the RBAC model for access control in open systems," in *Proc. 11th ACM Symp. Access Control Models Technol. (SACMAT)*, 2006, pp. 49–58.
[15] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. USENIX Secur. Symp.*, San Francisco, CA, USA, 2011, pp. 447–462.
[16] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang, R. Zhao, and L. Zhao, "Vehicle-to-everything (v2x) services supported by LTE-based systems and 5G," *IEEE Commun. Standards Mag.*, vol. 1, no. 2, pp. 70–76, Jul. 2017.
[17] M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca, "GEO-RBAC: A spatially aware RBAC," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 1, p. 2, 2007.
[18] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.
[19] A. Dmitrienko and C. Plappert, "Secure free-floating car sharing for offline cars," in *Proc. 7th ACM Conf. Data Appl. Secur. Privacy*, Mar. 2017, pp. 349–360.
[20] A. Dmitrienko, A.-R. Sadeghi, S. Tamrakar, and C. Wachsmann, "SmartTokens: Delegable access control with NFC-enabled smartphones," in *Proc. Int. Conf. Trust Trustworthy Comput.* Cham, Switzerland: Springer, 2012, pp. 219–238.
[21] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 2, pp. 198–208, Mar. 1983.

[22] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting," *Proc. Privacy Enhancing Technol.*, vol. 2016, no. 1, pp. 34–50, Jan. 2016.
[23] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, Aug. 2001.
[24] A. Francillon, B. Danev, S. Capkun, S. Capkun, and S. Capkun, "Relay attacks on passive keyless entry and start systems in modern cars," in *Proc. NDSS*, 2011, pp. 1–15.
[25] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis, "Practical NFC peer-to-peer relay attack using mobile phones," in *Proc. 6th Int. Conf. Radio Freq. Identificat., Secur. Privacy Issues*. New York, NY, USA: Springer-Verlag, 2010, pp. 35–49.
[26] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlidès, "Lock it and still lose it—On the (in) security of automotive remote keyless entry systems," in *Proc. 25th USENIX Secur. Symp. Secur.*, 2016, pp. 929–944.
[27] R. Garra, S. Martinez, and F. Sebe, "A privacy-preserving pay-by-phone parking system," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 5697–5706, Jul. 2017.
[28] D. Giese, K. Liu, M. Sun, T. Syed, and L. Zhang, "Security analysis of near-field communication (NFC) payments," *CoRR*, abs/1904.10623, pp. 1–10, Apr. 2019. [Online]. Available: https://arxiv.org/abs/1904.10623
[29] B. Groza, T. Andreica, and P.-S. Murvay, "Designing wireless automotive keys with rights sharing capabilities on the MSP430 microcontroller," in *Proc. 3rd Int. Conf. Vehicle Technol. Intell. Transp. Syst.*, 2017, pp. 173–180.
[30] L. C. Guillou and J.-J. Quisquater, "A 'paradoxical' identity-based signature scheme resulting from zero-knowledge," in *Proc. Adv. Cryptol.* New York, NY, USA: Springer-Verlag, 1990, pp. 216–231.
[31] L. C. Guillou, M. Ugon, and J.-J. Quisquater, "Cryptographic authentication protocols for smart cards," *Comput. Netw.*, vol. 36, no. 4, pp. 437–451, Jul. 2001.
[32] J. Han, Y.-H. Lin, A. Perrig, and F. Bai, "MVSec: Secure and easy-to-use pairing of mobile devices with vehicles," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, New York, NY, USA. Oxford, U.K.: Association for Computing Machinery, 2014, pp. 51–56.
[33] S. S. Hassan, S. D. Bibon, M. S. Hossain, and M. Atiquzzaman, "Security threats in Bluetooth technology," *Comput. Secur.*, vol. 74, pp. 308–322, May 2018.
[34] Y.-S. Huang and C.-H. Lung, "Device with identity verification-apply in car driving as an example," in *Proc. IEEE Int. Conf. Appl. Syst. Invention (ICASI)*, Apr. 2018, pp. 243–246.
[35] J. Isaak and M. J. Hanna, "User data privacy: Facebook, cambridge analytica, and privacy protection," *Computer*, vol. 51, no. 8, pp. 56–59, Aug. 2018.
[36] *Information Technology, Security Techniques, Digital Signatures With Appendix*, 2nd ed., ISO/IEC Standard 14888-2:2008, 2008.
[37] T. Kasper, A. Kühn, D. Oswald, C. Zenger, and C. Paar, "Rights management with NFC smartphones and electronic ID cards: A proof of concept for modern car sharing," in *Radio Frequency Identification*. Berlin, Germany: Springer, 2013, pp. 34–53.
[38] E. Kiltz and G. Neven, "Identity-based signatures," in *Identity-Based Cryptography*, vol. 2. IOS Press, 2009, pp. 31–44.
[39] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 447–462.
[40] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding attributes to role-based access control," *Computer*, vol. 43, no. 6, pp. 79–81, Jun. 2010.
[41] C. Lai, R. Lu, D. Zheng, and X. S. Shen, "Security and privacy challenges in 5G-enabled vehicular networks," *IEEE Netw.*, vol. 34, no. 2, pp. 37–45, Mar. 2020.
[42] K. Lemke-Rust, A.-R. Sadeghi, and C. Stüble, "An open approach for designing secure electronic immobilizers," in *Proc. Inf. Secur. Pract. Exper.* Berlin, Germany: Springer, 2005, pp. 230–242.
[43] Z. Li, Q. Pei, I. Markwood, Y. Liu, M. Pan, and H. Li, "Location privacy violation via GPS-agnostic smart phone car tracking," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5042–5053, Jun. 2018.
[44] Z. Li, K. Zhang, B. Chen, Y. Dong, and L. Zhang, "Driver identification in intelligent vehicle systems using machine learning algorithms," *IET Intell. Transp. Syst.*, vol. 13, no. 1, pp. 40–47, Jan. 2019.
[45] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," in *Proc. Black Hat USA*, 2014, p. 94.
[46] M. Muhammad and G. A. Safdar, "Survey on existing authentication issues for cellular-assisted V2X communication," *Veh. Commun.*, vol. 12, pp. 50–65, Apr. 2018.
[47] L. Popa, B. Groza, and P.-S. Murvay, "Performance evaluation of elliptic curve libraries on automotive-grade microcontrollers," in *Proc. 14th Int. Conf. Availability, Rel. Secur. (ARES)*, 2019, pp. 1–7.
[48] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.
[49] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. Workshop Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 1984, pp. 47–53.
[50] R. Steffen, J. Preißinger, T. Schöllermann, A. Müller, and I. Schnabel, "Near field communication (NFC) in an automotive environment," in *Proc. 2nd Int. Workshop Near Field Commun.*, 2010, pp. 15–20.
[51] X. Sun, S. Hu, L. Su, T. F. Abdelzaher, P. Hui, W. Zheng, H. Liu, and J. A. Stankovic, "Participatory sensing meets opportunistic sharing: Automatic phone-to-phone communication in vehicles," *IEEE Trans. Mobile Comput.*, vol. 15, no. 10, pp. 2550–2563, Oct. 2016.
[52] 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 *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2017, pp. 475–493.
[53] S. Tillich and M. Wójcik, "Security analysis of an open car immobilizer protocol stack," in *Trusted Systems*. Cham, Switzerland: Springer, 2012, pp. 83–94.
[54] J. Timpner, D. Schürmann, and L. Wolf, "Secure smartphone-based registration and key deployment for vehicle-to-cloud communications," in *Proc. ACM Workshop Secur., Privacy Dependability Cyber Vehicles (CyCAR)*, 2013, pp. 31–36.
[55] M. Turuani, "The CL-Atse protocol analyser," in *Proc. Int. Conf. Rewriting Techn. Appl.* Cham, Switzerland: Springer, 2006, pp. 277–286.
[56] T. Unterluggauer and E. Wenger, "Efficient pairings and ECC for embedded systems," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Cham, Switzerland: Springer, 2014, pp. 298–315.
[57] M. Van Ly, S. Martin, and M. M. Trivedi, "Driver classification and driving style recognition using inertial sensors," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2013, pp. 1040–1045.
[58] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1313–1328.
[59] R. Verdult, F. D. Garcia, and J. Balasch, "Gone in 360 seconds: Hijacking with Hitag2," in *Proc. 21st USENIX Conf. Secur. Symp.* Berkeley, CA, USA: USENIX Association, 2012, p. 37.
[60] R. Verdult and F. Kooman, "Practical attacks on NFC enabled cell phones," in *Proc. 3rd Int. Workshop Near Field Commun.*, Feb. 2011, pp. 77–82.
[61] W. Webb, *Wireless Communications: The Future*. Hoboken, NJ, USA: Wiley, 2007.
[62] Z. Wei, Y. Yanjiang, Y. Wu, J. Weng, and R. H. Deng, "HIBS-KSharing: Hierarchical identity-based signature key sharing for automotive," *IEEE Access*, vol. 5, pp. 16314–16323, 2017.
[63] J. Wetzels, "Broken keys to the kingdom: Security and privacy aspects of RFID-based car keys," 2014, *arXiv:1405.7424*. [Online]. Available: http://arxiv.org/abs/1405.7424
[64] L. Wouters, E. Marin, T. Ashur, B. Gierlichs, and B. Preneel, "Fast, furious and insecure: Passive keyless entry and start systems in modern supercars," in *Proc. IACR Trans. Cryptogr. Hardw. Embedded Syst.*, 2019, pp. 66–85.
[65] Y. Zou, A. H. Mhaidli, A. McCall, and F. Schaub, "'I've got nothing to lose': Consumers' risk perceptions and protective actions after the Equifax data breach," in *Proc. 14th Symp. Usable Privacy Secur. (SOUPS)*, 2018, pp. 197–216.

BOGDAN GROZA (Member, IEEE) received the Dipl.Ing. and Ph.D. degrees from the Politehnica University of Timisoara (UPT), in 2004 and 2008, respectively. In 2016, he successfully defended his habilitation thesis having as core subject the design of cryptographic security for automotive embedded devices and networks. He has been actively involved inside UPT with the development of laboratories by Continental Automotive and Vector Informatik. Besides regular participation in national and international research projects in information security, he lead the CSEAMAN Project, from 2015 to 2017, and the PRESENCE Project, from 2018 to 2019, two research programs dedicated to automotive security funded by the Romanian National Authority for Scientific Research and Innovation. He is currently a Professor with UPT.

TUDOR ANDREICA received the B.Sc. and M.Sc. degrees from the Polithenica University of Timisoara, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree. Since 2015, he has been working as a Software Engineer with HELLA Romania, focusing on the security of various in-vehicle systems. He was a Research Student in the CSEAMAN Project and currently joined the PRESENCE Project as a Ph.D. Student. His research interest includes automotive cybersecurity.

ADRIANA BERDICH received the Dipl.Ing. and M.Sc. degrees from the Politehnica University of Timisoara (UPT), in 2017 and 2019, respectively, where she is currently pursuing the Ph.D. degree. From 2015 to 2018, she was a Software Developer in the automotive industry for Continental Corporation in Timisoara. She is also a Research Student in the Presence Project focusing on environment-based device association and also continues as a Software Developer in the automotive industry for Vitesco Technologies focusing on power-train applications.

PAL-STEFAN MURVAY (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from the Politehnica University of Timisoara (UPT), in 2008, 2010, and 2014, respectively. He is currently a Lecturer with UPT. He has a ten-year background as a Software Developer in the automotive industry. He has worked as a Postdoctoral Researcher in the CSEAMAN Project. He is currently a Senior Researcher in the PRESENCE Project. He also leads the SEVEN Project related to automotive and industrial systems security. His current research interest includes automotive security.

EUGEN HORATIU GURBAN received the Dipl.Ing. degree, the M.Sc. degree in automotive embedded software, and the Ph.D. degree from the Politehnica University of Timisoara (UPT), in 2007, 2009, and 2014, respectively. He is currently a Lecturer with UPT. He has worked for three years in the automotive industry as a Software Verification Engineer and the Team Leader of the Research and Development Software Engineering Department, Autoliv Romania, from 2007 to 2010. He currently works as a Postdoctoral Researcher in the PRESENCE Project.