Hindawi
Journal of Advanced Transportation
Volume 2021, Article ID 5569331, 18 pages
https://doi.org/10.1155/2021/5569331

Research Article

Shared Mobility for Transport and Its Environmental Impact VeSIPreS: A Vehicular Soft Integrity Preservation Scheme for Shared Mobility

Valaenthin Tratter,[1,2] **Mudassar Aslam**,[1,3] **and Shahid Raza**[1]

[1]*RISE Cybersecurity, Stockholm, Sweden*
[2]*Department of Electrical and Computer Engineering, Technical University of Munich (TUM), Munich, Germany*
[3]*Department of Cybersecurity, National University of Emerging Sciences (FAST NUCES), Islamabad, Pakistan*

Correspondence should be addressed to Valaenthin Tratter; valaenthin.tratter@tum.de

Received 19 February 2021; Accepted 31 May 2021; Published 23 June 2021

Academic Editor: Qi-zhou Hu

Car manufacturers are noticing and encouraging a trend away from individual mobility, where a vehicle is owned and driven by one or only a few other persons, and towards shared-mobility concepts. That means that many different people use and have access to the same vehicle. An attacker disguised as a regular short-time user can use the additional attack vectors (s)he gets by having physical access to tamper the vehicle's software. The software takes a continuously more crucial role in cars for autonomous driving, and manipulations can have catastrophic consequences for the persons on board. Currently, there is no mechanism available to the vehicle owner to detect such manipulations in the vehicle done by the attacker (short-time user). In this work, a novel vehicle attestation scheme called Vehicular Soft Integrity Preservation Scheme (VeSIPreS) is proposed to detect tampering in the software stack of a vehicle and guarantee the upcoming driver that the previous user has not changed the software of the vehicle. The solution consists of a software module in the vehicle and a mobile-based user application for the vehicle owner to monitor the vehicle's soft integrity. Inside the vehicle, the software module is implemented in the central gateway, which acts as the primary security component. VeSIPreS uses Trusted Platform Module (TPM) in the central gateway, which anchors trust in our proposed solution. This paper also provides a proof-of-concept implementation with a TPM, demonstrating its application and deployment feasibility and presentig a security analysis to show the security of VeSIPreS.

1. Introduction

Safety in vehicles has always been an essential aspect of automotive development; however, vehicles have changed a lot since their invention in the 19th century. The first introduction of an electronic control unit (ECU) in 1975 by Ford started the process of automating functions in the vehicle to make existing systems more efficient and add new features. The introduction of safety features such as antilock braking system (ABS) or electronic stability control (ESC) were essential milestones in car history and were only made possible by introducing ECUs. With the continuous introduction of electronic components in the vehicle, securing these components got increasingly more important and hence the vehicular security became an important requirement in the automotive development. Future trends in automotive industry include autonomous driving of the vehicles and shared use models for the cars. All of these trends will add further electrical components, interfaces, and software in the vehicle.

The trend towards shared mobility will significantly change travelling in the future [1]. A vehicle is thereby used by many persons who are not always trusted and can be potential attackers. They can use the extended access they gain for manipulations on the software that harm upcoming drivers. The same is true for increasingly connected and autonomous cars, which provide more interfaces that can be used as attack vectors. An increase in

attack vectors also increases the probability of malicious software manipulations. At the same time, it gets always more crucial to secure the software due to the ever-increasing tasks and corresponding functional safety responsibility. This paper proposes a scheme that allows detecting such scenarios (VeSIPreS).

1.1. Use-Case Scenario. This paper targets applications such as car-sharing, rental cars, or company fleets, where many people have physical access to a vehicle. A car-sharing (a list of some major commercial car-sharing providers includes http://www.zipcar.com, http://www.share-now.com, or http://www.sunfleet.com) or rental car company (a list of some well-established rental car companies includes http://www.europcar.com, http://www.sixt.com or http://www.hertz.com) has a high number of different users that use vehicles from it. Since they can never trust a client that he is not manipulating anything in the car and at the same time must guarantee that the vehicle is intact for the next user, they need a mechanism to check the software in a car for eventual changes quickly and reliably. The considered threat scenario is that an attacker rents a vehicle from the rental company. He then installs modified software on one of the controllers (ECU or Gateway) in the car intending to compromise the security of an upcoming driver or the vehicle itself. He returns the vehicle to the rental company who cannot detect the modification and hence rents it out again to the next user. The manipulation can then harm this user by either compromising the privacy or functional safety of the vehicle. A demonstration of the workflow sequence is shown in Figure 1.

1.2. Major Contribution. TPM chips are already well-established cryptographic coprocessors used in desktop computers, servers, and smartphones. These chips allow to attest a system and bring the concept of a trusted platform to it. Vehicles are evolving towards "smartphones with wheels" where it is essential always to assure the system's safety. There is not yet much research done on how the benefits of a TPM can be used for the special needs and circumstances in a vehicle. Most research focuses on securing the remote update capability of the vehicle.

In this paper, we study potential application fields of TPM 2.0 security modules in an automotive environment. Based on the attestation capabilities offered by TPM chips, we propose a protocol which allows the user to attest a vehicle for possible software manipulations. We believe to the best of our knowledge that the proposed idea of VeSIPreS is not presented before. Main contributions in this paper are the following:

(i) Present a novel approach using TPM capabilities to preserve the soft integrity of the modern vehicles

(ii) Provide a proof-of-concept implementation to present application potential of TPM in state-of-the-art and future vehicles

(iii) Allow shared vehicle user to detect any software manipulations in the vehicle (VeSIPreS) by the previous user

(iv) Implement a proof-of-concept prototype of the VeSIPreS scheme which includes a vehicle simulator representing the vehicle internals and a mobile user application

(v) Perform a detailed security analysis of the proposed scheme (VeSIPreS) to show its security

The rest of the paper is organised as follows: Section 2 gives an overview of the state of the art in the field of vehicle tampering, system health attestation, and remote firmware update, so-called over-the-air (OTA) update. The addressed problem is presented in Section 3 by describing a typical use-case scenario (Section 5). A detailed description of the proposed solution, the VeSIPreS protocol, is given in Section 6. We evaluate the feasibility of the proposed solution, followed by a proof of concept presented in Section 7. Finally, the solution is analysed from security perspective in Section 8; and we conclude the paper with an outlook to possible future extensions in Section 9.

2. Related Work

The automotive industry faces the trend towards continuously more connected and automated vehicles and self-driving cars. With the introduction of more complex computers, interfaces, and other in-vehicle systems, the potential attack surface also widens up for vehicles. In [2–4], classifications and examples of possible attack scenarios are given and illustrate the broad variety of such attacks. The following sections give an overview of famous attacks on vehicles from the past and illustrate some modern automotive technology's vulnerabilities and describe state-of-the-art prevention attempts for such attacks. An important countermeasure for such attacks is the capability to modify the software and fix potential weaknesses quickly. Therefore, over-the-air updating is needed. Furthermore, this paper will give an overview of the field of remote attestation, a crucial component of VeSIPreS.

2.1. Vehicle Tampering. Koscher et al. [5] showed experimentally what hackers could do after they infiltrated the vehicle at a particular entry point. This marked a transition from theoretical research contributions towards a practical demonstration of the vulnerabilities of the cars. According to their experiment, the insertion of messages on the Controller Area Network (CAN) bus can perform physical changes to a car and affect the functional safety of it. Original equipment manufacturers (OEMs) such as Toyota appease the problem, arguing that physical access is needed to perform such attacks and an attacker would have in such a case easier ways of tampering the car, such as manipulating the break wires [6]. OEMs focused on the past more on fixing remote attack threats on vehicles electronic systems rather than on physical attacks. Checkoway et al. extended their research and released in [7] an exploitation of vulnerabilities in the infotainment system. This exploit allowed them by using the remote interfaces (Bluetooth and telematics unit) to inject arbitrary commands (CAN frames) into the vehicle network. That means that the attacks which were only possible through physical access to the car were now also made executable remotely.

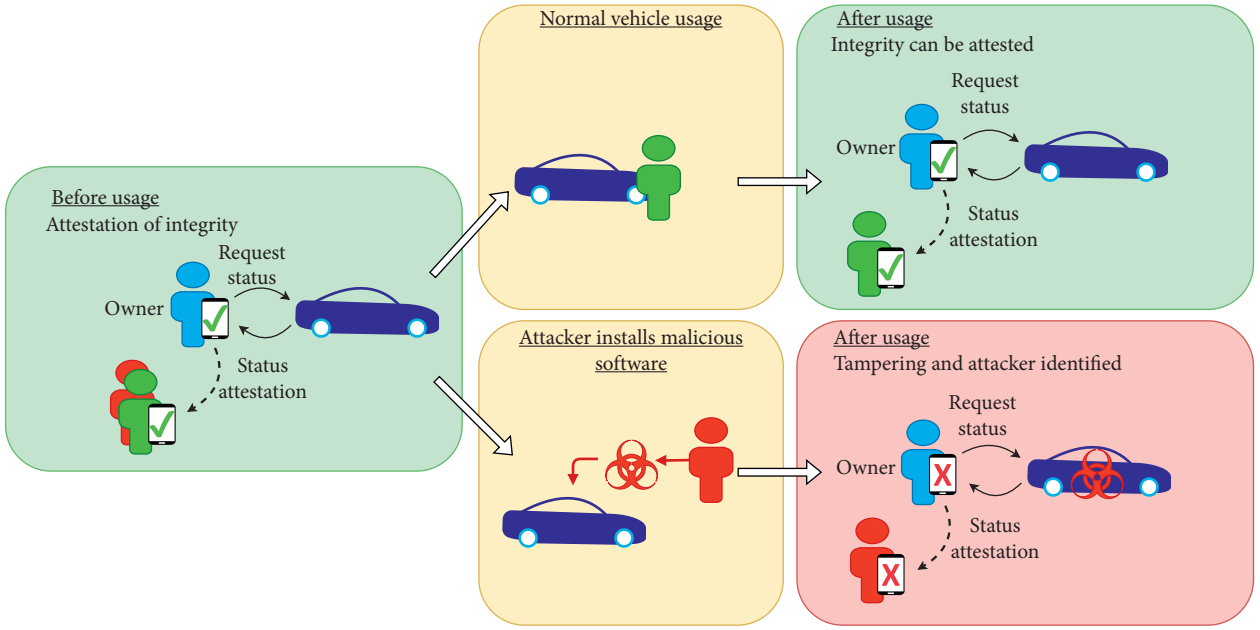

Figure 1: Considered scenario with normal usage on the car and an attacker which installs malicious software. The software manipulation can be detected afterwards by using VeSIPreS.

In 2014, Miller and Valasek [8] surveyed different car models and their attack surface. They looked at the internal network architecture of different vehicles and discussed possible attack vectors and found several security threats. They identified several reasons that lead to the found attack surface: the increase of ECUs in a typical car from around 10 in 2010 to 100 in 2014, the implementation of more complex features with tools that are already known to attackers from desktop PCs, and a large variety of different architectural approaches. Such surveys were also performed by other groups and showed a wide variety of different attack vectors [2, 4]. Miller and Valasek in 2015 [9] used a series of exploits in the communication interface of a 2014 Jeep Cherokee's infotainment system to enter and take over the control of the car remotely by using a cellular network. The access to the infotainment system allowed them to change infotainment-related settings such as the volume or the view of the infotainment display, but also to get car information like the current location of the vehicle. There is a separated proxy chip in the infotainment system of the Jeep which accesses the two CAN buses of the car. This proxy can be compared with the gateway in the considered scenario of this paper. Since the proxy was not adequately protected, the researchers were able to install manipulated software on the proxy chip which allowed them to send arbitrary CAN messages to the vehicle. With access to the CAN bus, they could affect the functional safety of a car by sending commands which perform physical changes to essential car components (e.g., in the breaks, steering, engine control, and airbags). Chrysler reacted to this exploitation and consequently recalled 1.4 million vehicles [10].

A recent attack found by L. Wouters used a vulnerability in the key fob to remotely unlock and start a Tesla Model X. He was able to rewrite via Bluetooth the firmware on the key fob of the car and lift an unlock code that allowed him to access the car. In the car, he used other vulnerabilities to start the vehicle and drive away with it [11]. Tesla faced already some other attacks in the past, for example, a vulnerability in the infotainment system that allowed remote access to the car [12], but similar attacks can be found on many highly connected cars (e.g., BMW [13]). Other prominent attacks often target the central locking system to access the car [14, 15].

2.2. Vehicle Health Assurance. In [16], the implementation process and benefits of using TPM in a vehicle are discussed. The trusted computing group (TCG) released a particular TPM profile for vehicles, namely, the Automotive Thin Profile. Volkswagen is the first OEM that publicly announced the use of an automotive TPM from Infineon in their vehicles [17]. They use the TPM to protect the external interfaces and check the identity of the exchanged data. The TPM manages the cryptographic keys needed to ensure secure communication and can be used to propose features such as car-sharing, managing access rights, vehicle to vehicle communications, and over-the-air updating.

An approach to prevent attackers from accessing the diagnosis functionalities of the car is shown by Kleberger and Olovsson in [18]. Access to this functionality would allow the attacker to install malicious software on the vehicle. According to the paper, the diagnosis interface in modern cars is often only secured through authentication. However, when the equipment gets lost, an attacker can manipulate potentially any vehicle that uses this key. Kleberger and Olovsson secure the authentication by introducing a trusted third party which issues authorisation tickets. Authorisation tickets allow vehicles to validate

diagnostic requests. Together with the corresponding authorisation protocol and access control mechanisms, this can prevent such attacks. By implementing an authorisation method, they achieved authentication, message integrity of diagnostics sessions, filtering of specific diagnostics messages, and support encryption of diagnostics sessions. However, they have no possibility to check the integrity of the vehicle with this method. If an attacker is still able to tamper the software, he could potentially also disable this functionality. Besides the security of the individual ECUs, also network security is essential. For the widely used CAN bus, Kornaros et al. proposed in [19] a comprehensive security concept including encryption of the traffic on the bus to prevent attacks on the bus system such as man-in-the-middle attacks.

2.3. Over-the-Air Update. With vehicles that transform themselves into smartphones with wheels due to their highly complex software, it is impossible to guarantee at manufacturing time that the software is adequately protected against all present and future security threats. To react to software bugs and new security threads, the possibility to update the system is essential. The current approach of physical updating vehicles via a diagnosis device in a repair shop gets thereby quickly unpractical. It takes months to complete the update of a full fleet and is very costly. To avoid massive recalls for software update in the future, many OEMs want to implement OTA updates into their vehicles. Recalls are costly for car manufacturers and cause a security issue due to the delay between the discovery of the issue and the fixing of it. Beside enormous cost savings for the vehicle manufacturer, it also reduces the turnover time and enhances the security as bugs can be patched relatively quickly with updates [20]. An additional benefit is the possibility to add other software features after sale. In modern cars exist already, the possibility to update noncritical parts such as the navigation maps or infotainment system of vehicles via OTA updates. A drawback of full OTA update is that it is necessary to connect all ECUs among each other for updating via remote links, which opens new potential security threats. Attacks in the past were made possible by changing or modifying the software on crucial components (Tesla key fob [11]). Advantages are lower cost, improved safety, improved customer satisfaction, frequent updates, and increased value. Halder et al. compared in [20] different methods on how to implement secure OTA software updates in connected vehicles. The different possibilities vary mainly in the encryption and assurance technique of the update packet.

In [21], Idrees et al. proposed a protocol using trusted hardware for OTA updates. They propose an on-board security architecture to facilitate the update process by using security chips such as hardware security modules (HSMs) or TPMs, which also act as a root of trust. Depending on the type of host-platform, different security stages are considered. This motivates the implementation of a TPM in the car to assure the update process.

There are other protocols specified for firmware update [22, 23]; however, they only focus on a secure firmware reception without addressing remote attestation. All these solutions only protect though the access to install new malicious software. If an attacker is somehow still able to manipulate the software, they provide no mechanism to detect such changes. In [24], the authors developed a distributed remote update system for OTA updating vehicles. Howden et al. [23] analysed some elements which are to consider implementing automotive OTA updating.

2.4. Remote Attestation. The main goal of an attestation process is to decide whether the entity that is attested is trustworthy or not. Attesting the status is needed in many different applications, e.g., in the assessment of nodes in a network to assure that they are not compromising the network. The Internet Engineering Task Force (IETF) is working on Remote Attestation Procedures (RATS), a standardisation of remote attestation [25]. They propose different approaches for different applications, among others also automotive.

In principle, an attestation architecture works as follows: the attester (the entity that must be appraised) creates evidence which is sent to the verifier (the entity that appraises). This evidence is then processed by the verifier considering any endorsements and policies to get an attestation result. This result is then forwarded to the relying party which evaluates that result and decides how to interpret it. The verifier and relying party can be the same entity (see Figure 2 for a remote attestation schematic). The actual implementation of this principle scheme depends in detail on the real use case. The actual remote attestation protocol differs in the implementation from this model, depending on the specific design specifications, and can be further extended.

There are two main types of environments of an attester. The layered attestation procedure is a concept which can, for example, be found in a staged boot sequence. The first attesting environment delegates its tasks to the next attesting environment. This next one then measures once again the upcoming measurement of its upcoming environment. It is necessary to assure that the first environment is trusted and cannot alter, and all other environments cannot alter their own measurements.

A composite device is, on the other hand, an entity consisting of several subentities. To determine the trustworthiness of the entire device, the trustworthiness of every single component must be tested. The evidence of the single subentities is then collected by a lead attester that generates the evidence for the composite device.

RATS further proposes different topologies, namely, the Passport Model and the Background-Check model. In the Passport Model, the attester sends the evidence to the verifier who then creates the attestation result and sends that back to the attester. A relying party can then request the attestation result and decide whether it is trustworthy or not. In the Background-Check Model, the evidence is sent from the attester to the relying party, which forwards it to the verifier. The verifier creates the attestation result that is then sent to the relying party. There are also combinations of the two models possible, depending on the use case. A requirement for both models is that the verifier must be trusted.

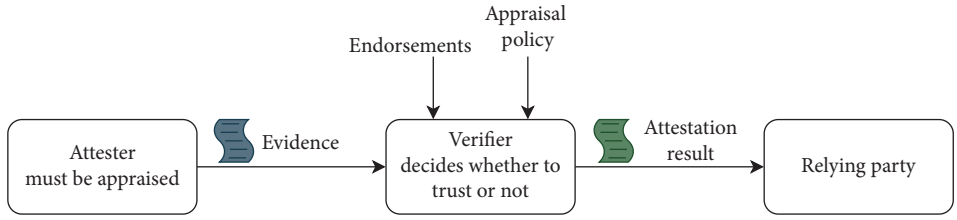

Figure 2: General remote attestation procedure as defined in [25].

The originality of the data/evidence is assured using two methods. First, a random number, usually referred to as nonce, is generated by the remote entity. The nonce is included in the signed response from the attester. The remote entity can then verify if the right nonce is included in the signature. Since the nonce is always different, there is no possibility to replay or alter a message. The second approach uses synchronised clocks in both the remote entity and the attester. The timestamp can then be included in the response. This, however, needs accurate clocks which must be trusted and are not suited for this work.

Table 1 gives an overview of the most relevant papers in that field. As we see in the research and the attacks, there is a big strive to secure the software of the cars against unwanted changes. There is however no possibility for the user to check at any moment the status of the software in the car. The upcoming sections will then explain the approach and evaluate VeSIPreS in detail.

3. Use-Case Scenario

VeSIPreS considers the scenarios where a vehicle is used by multiple users who have full access to a vehicle. Such a scenario defines our attacker who could be an insider, as an employee for a closed vehicle pool, or an outsider, like a customer of a rental car company. Such an attacker can install malicious software on the vehicle without other users noticing the change. Few possible scenarios include the following:

(i) An attacker can rent a car from a rental car or a leasing company. When the vehicle is returned, an agent will check the status of the vehicle but has only limited possibility to check the software of the car and software manipulations will be invisible to the agent.

(ii) The same is valid for car-sharing, which can be compared with the conventional rental car model. In a public, free-floating car-sharing system, a registered user can spontaneously rent a car and return it anywhere inside the business area. There is typically no agent who checks the car after return, so the probability of success is even higher in this scenario.

(iii) Transport companies with bus or truck fleets. An insider is the attacker and installs the malicious software while he has access to the vehicle.

(iv) Individual vehicle owners who rent their vehicles out to an untrustworthy party. A possible scenario can be a private person who gives its car to a repair shop or a valet parking service. Also, if a car is rented out to a friend, there is never certainty on whether he manipulated the software during his usage period.

In these scenarios, the vehicle operator needs a possibility to check if the software in the car is still as intended. Checking must be done before and after the vehicle user uses it, so it can be assured that the software is safe before he starts the use. An attestation protocol is optionally sent to the user such that he can trust the software running on the vehicle. After the usage, a software check attests that the user has not manipulated the vehicle during his usage. The software checks are performed by using a smartphone application which can take a measurement of the vehicle and aims at finding changes in the software stack of the vehicle. The measurement capture can be initiated either by the user or an agent of the vehicle company.

4. Attack Surfaces

A modern vehicle has a relatively large attack surface due to all its sensors and interfaces. The individual attack vectors can be divided into physical and remote attack vectors. We consider a strong attacker model by focusing on physical attack vectors where an attacker has full access to the vehicle. While an attacker with only access to remote attack vectors cannot use physical attack, an attacker with physical access can use both at the same time which increases the attacking chances. Physical attack vectors are in addition typically not as well protected as remote vectors are.

4.1. Remote Attack Vectors. Remote attack vectors are not only accessible by the user inside the vehicle but also from an attacker without physical access to the car. Such vectors include vehicle-to-everything (V2X) communications, Wi-Fi, and Bluetooth (if accessible from outside the car), cellular communication, satellite navigation system, radio key fobs, and remote sensors such as tire pressure measuring systems. Vehicle manufacturers are well-aware of these interfaces and protect them accordingly against attacks. However, studies such as [7] show that there is still attacking potential, which is demonstrated in attacks such as [9].

4.2. Physical Attack Vectors. The physical attack surface includes all attack vectors that are accessible by persons inside the vehicle. The attacker has full access and can

Table 1: Overview of the relevant research literature in the field of trusted automotive, diagnosis, and over-the-air update.

Author	Title	Year	Automotive	TPM	Topic
Muhammad Sabir Idrees, Hendrik Schweppe, Yves Roudier et al.	Secure automotive on-board protocols: a case of over-the-air firmware updates [21]	2011	Yes	No	OTA update
Richard Petri, Markus Springer, Daniel Zelle et al.	Evaluation of lightweight TPMs for automotive software updates over the air [16]	2016	Yes	Yes	OTA update
Yunshui Zhou, Xinkai Wu, Pengcheng Wang	Secure software updates for intelligent connected vehicles [24]	2019	Yes	No	OTA update
Pierre Kleberger, Tomas Olovsson	Protecting vehicles against unauthorised diagnostics sessions using trusted third parties [18]	2013	Yes	No	Diagnosis
Pierre Kleberger, Tomas Olovsson	Securing vehicle diagnostics in repair shops [26]	2014	Yes	No	Diagnosis
Ahmad MK. Nasser	Securing safety-critical automotive systems [27]	2019	Yes	Yes	General
Geunhyoung Kim, Im Y. Jung	Integrity assurance of OTA software update in smart vehicles [28]	2019	Yes	No	OTA update
Andreas Fuchs, Christoph Krauß, Jürgen Repp	Advanced remote firmware upgrades using TPM 2.0 [22]	2016	No	Yes	OTA update
George Kornaros, Dimitris Bakoyiannis, Othon Tomoutzoglou et al.	TrustNet: ensuring normal-world and trusted-world CAN-bus networking [19]	2019	Yes	No	General
James Howden, Leandros Maglaras, Mohamed Amine Ferrag	The security aspects of automotive over-the-air updates [23]	2020	Yes	Yes	OTA update

connect to existing ports in the vehicle or strip down control units or networks to tamper the components. Inside the vehicle, he can access the on-board diagnostics (OBD-II) port that is intended for maintenance purposes. With a simple aftermarket OBD-II-dongle, he can access to this interface.

The infotainment is a crucial component in the interaction between the driver of the car and the vehicle. Besides a screen and interaction interfaces (keypad, buttons, and touchscreen) to communicate with the driver, it offers interfaces to connect user devices, such as Bluetooth or Wi-Fi, and has other interfaces such as a USB or CD drive that can be used for inserting malicious software to the system. Besides interfaces that are foreseen to be used by the user, an attacker can also strip down the internal networks or use the debugging ports in the ECUs itself. All these attack vectors can be accessed without manipulating the vehicle. For those attack vectors, it may be needed to remove some cover, but no modification of the essential components is required. The fact that no modification on the hardware is necessary distinguishes the attack surface from physical tampering, which is not in the scope of this paper.

5. Attack Model

Vehicles are equipped in each model iteration with more complex and sophisticated computers and interfaces. These include remote interfaces for a remote key fob or the remote tire pressure sensors and interfaces for the infotainment system such as Bluetooth or Wi-Fi. Furthermore, they can be equipped with interfaces to communicate with other vehicles (V2V) or infrastructures such as road signs (V2I), summarised in V2X. This process is mainly driven by current trends in the automotive industry which continuously require more computing power and interfaces with the outer world. Trends include the electrification of the drive train, sophisticated driver aiding systems, more comfort features, communication with other vehicles and infrastructure, monitoring of engine parameters, and more. These new developments often come with more ECUs which control specific functions. Since these driving computers control safety-related functions such as the brakes or steering, it is inevitable to be sure that these computers that are needed for the functional safety of the vehicle are not manipulated. According to Petri et al. in [16], potential motivations for an attacker can be as follows:

(i) The manipulation of the firmware to compromise the operational safety

(ii) Theft of intellectual property, such as source code or parameters (encoded within firmware)

(iii) Theft of privacy-related data (e.g., navigation destinations, driving behaviour, and address books)

(iv) Manipulation of odometers

(v) Illegal feature activation (i.e., activating software and hardware features without payment)

(vi) Chip tuning, firmware manipulation (with safety-critical impact), and firmware downgrade (as a possible intermediate step for an attack)

(vii) Product counterfeiting and used part refurbishment

5.1. Vehicle Rental Companies Attack Scenario. This section explains the proposed attack based on the scenario of a rental car company. It can, however, also be ported on other scenarios where different persons use a shared vehicle.

In the proposed attack, an agent from a rental car company rents out the vehicle to an attacker. The attacker has now full access to the vehicle and installs malicious software on at least one ECU in the vehicle which is intended to harm the upcoming user of that car. Afterwards he returns the vehicle again to the rental company. The agent cannot identify the manipulation of the software, and so the vehicle gets rented out to the next user (see Figure 3). The

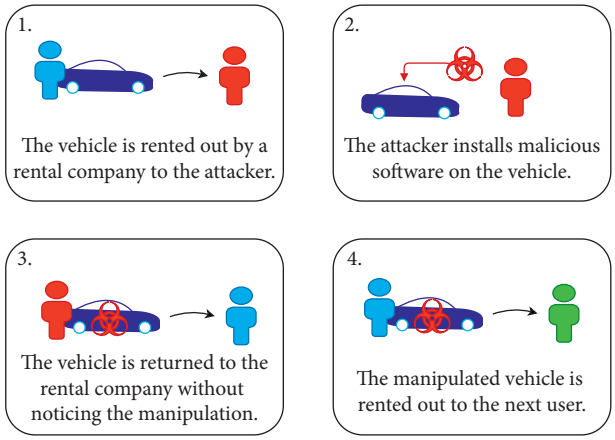

Figure 3: Possible attack scenario to manipulate a vehicle to harm an upcoming user of the vehicle.

manipulation can compromise the privacy or functional safety of the vehicle. Compromising the privacy of the driver includes attacks such as tracking of the vehicle, reading the address information of the connected phone, sending vehicle status updates, and more. Compromising the functional safety targets to harm the vehicle or driver. Unlike physical tampering, for example, cutting a brake wire, to harm a person, software manipulation is much more insidious and effective. There is no feasible possibility for a user to detect manipulation in the software stack with state-of-the-art methods and even more difficult to trace the attack back to identify the attacker. For example, instead of cutting a brake wire, the attacker can install in the break controller modified software with the condition that disables critical drive functions. These actions will lead to an attacker provoked accident and harm the driver.

The attacker uses for his attack the full physical access of the vehicle which he has during his rental period. Physical access is beneficial for the attacker since he can use the intended interface to install updates on the car, on most modern vehicles via the OBD-II port. The OBD-II port is a standardised interface that is mandatory for cars with a combustion engine to read engine parameters for diagnosis purposes on a test bench. Most car manufacturers use this interface for diagnosis and updating of vehicle's parameter.

Today's vehicles have no mechanism to detect or avoid such a scenario. ECUs and the gateway that links the different networks and interfaces are often not cryptographically protected against intrusions. That means that the stated attack would be successful. This shows how crucial it is to both secure the electronics against manipulation and introduce a mechanism to detect such attacks and manipulations.

6. Vehicular Soft Integrity Preservation Scheme (VeSIPreS)

To counter the presented attack, the ability of performing an integrity check on all software running in the car is required. The proposed solution is called VeSIPreS, an acronym for Vehicular Soft Integrity Preservation Scheme. VeSIPreS nominates a comprehensive system consisting of software in the central gateway and ECUs in the car and an application installed on the smartphone of the vehicle owner. With it, the owner can perform a software check by measuring the entire software stack of the vehicle before it gets rented out. After the usage by an untrusted person, or whenever he wants to check the status of the car, the owner takes a second measurement of the whole software stacks and compares that one with the stored first measurement. If both measurements are still the same, which corresponds to no changes in the software, it can be guaranteed that the previous user has not made any changes to the software. If, however, an attacker installs malicious software on one or more ECUs in the vehicle, the measurement that is taken after he returned the car will no longer match with the first measurement. In that case, the attack will be detected, and the attacker can be identified. Measurements can easily be performed by the owner using the VeSIPreS client app on his smartphone. The app connects to the vehicle and initiates a measurement, stores it, and compares two measurements with each other. Furthermore, the app can create a report which is shared with the user who rents the car.

The user app connects through a secured channel with the gateway in the car. The gateway is equipped with a TPM and processes all commands. As depicted in Figure 4, the attacker must follow this path through the gateway for an attack (otherwise, physical modification, e.g., of the wiring is needed, can be prevented by security stage SM and SH, see Section 6.3 for more details). The following sections will describe the individual components and steps of VeSIPreS in detail.

6.1. First Initialisation. The gateway and, depending on their security level, other ECUs are equipped with a TPM that needs to be initialised. The car manufacturer does this during the assembly of the car. All used notations in this paper are explained in detail in Table 2.

6.1.1. TPM Hierarchies. A TPM comes with four different hierarchies for cryptographic key management. The first hierarchy is used exclusively by the chip manufacturer and is not intended and suited for hosting the needed keys on it. The second hierarchy is reserved for the platform manufacturer, and the third and fourth hierarchy stages are reserved for the user. The main difference between the third and fourth hierarchy stage is that the third one is persistent, while the fourth one changes with every power cycle. Therefore, the fourth one is intended for temporary cryptographic operations and not suited for VeSIPreS. The keys needed for the proposed solution must either be created on the second or third hierarchy stages (platform or owner hierarchy). The actual choice of the hierarchy depends on how VeSIPreS is implemented and sold as a product. In the further course of this work, the second hierarchy (platform hierarchy) is used for the cryptographic keys of VeSIPreS. This second hierarchy stage is reserved for the platform manufacturer, which is in this case, the vehicle manufacturer. The vehicle manufacturer is the only one who can

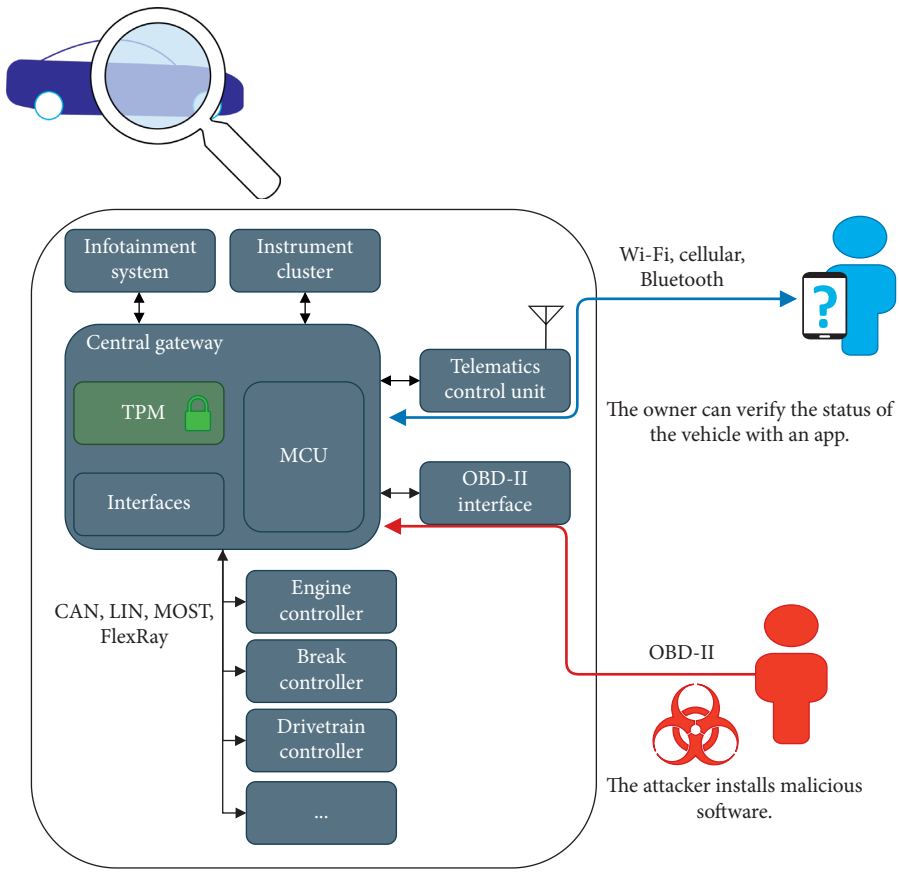

Figure 4: In-vehicle network architecture with the attack path of the attacker and the communication path for VeSIPreS.

Table 2: Notations used in VeSIPreS.

Notation	Description	Provider
S0, S1, S2	Security stages of the ECUs (see Section 6.3).	
N_{GW}	Nonce to quote gateway PCR.	Client app
N_{En}	Nonce to quote ECUn PCR.	Gateway
GWK (GWK_P, GWK_S)	Gateway signing key. Used to sign messages from gateway to user app. GWK_P is the public key, which is used to validate the signature, GWK_S is the private part of the key which is only accessible to the TPM in the gateway.	Gateway
E_nK (E_nK_P, E_nK_S)	ECUn signing key. Used to sign messages from ECUn to gateway. E_nK_P is the public key, which is used to validate the signature, E_nK_S is the private part of the key which is only accessible to the TPM or HSM in ECUn.	ECUn
Report	Measurement report containing the status of the system and the measurements of the boot stages (if measured boot is performed, only SH and gateway).	Gateway, ECUn
Platform configuration register (PCR)	Platform configuration registers. Special registers in TPM which represent the running software.	Gateway, ECUn

configure this hierarchy stage which guarantees the same cryptographic keys over the entire lifetime of the vehicle. The third hierarchy stage can then be used for user or owner specific tasks and can change with every transfer of ownership of the vehicle.

6.1.2. Gateway Initialisation. For the initialisation of the gateway, first, the authenticity of the TPM in the gateway must be checked. The TPM chip contains a certificate which is signed by the chip manufacturer who loaded it during manufacturing time persistently on the chip. This means that the certificate cannot be modified nor replaced by a potential attacker once loaded in the TPM. The vehicle manufacturer downloads the certificate from the TPM and checks the signature by using the public certificate of the chip manufacturer. After the vehicle manufacturer assured that the TPM is genuine, he can attest the vehicle using the TPM as the root of trust. The vehicle manufacturer, therefore, sets a key in the platform hierarchy stage and creates a certificate of the vehicle using that key and its private key. The certificate is then persistently loaded in the TPM. The public part of the certificate must be public such that the user app can later load the certificate from the TPM and verify the signature

using the certificate of the car manufacturer. If the signature is valid, the user can be sure that the car is genuine and was manufactured by the owner of the certificate (vehicle manufacturer).

6.1.3. ECU Initialisation. Depending on their security level, some ECUs are equipped with a cryptographic coprocessor. Depending on which type of coprocessor, they can be categorised in either stage SM or SH (HSM or TPM in general (see Section 6.3). Both stages allow secure communication among the ECUs when equipped. Stage SH enables furthermore to perform a measured boot. In the initialisation phase, a signing key is created, which is then later used for the communication between the ECU and the gateway. The same as for the gateway, the first step is to verify the authenticity of the chip by checking the certificate of it with the public certificate from the chip manufacturer. Then, the actual signing key E_nK for ECUn is created in the HSM or TPM of the ECU. The gateway needs an exact list of all ECUs in the vehicle and of the security level that they have. The list contains also the public keys E_nK_P of all ECUs stored in the gateway. The keys are needed so the gateway can request a report from all ECUs and verify them if they are signed.

6.2. Gateway. The gateway is a particular ECU in the car which acts as the central interface between the internal network of the vehicle and the connections to the outer world. It connects the different networks and domains in the car among each other. Typically, it is the gateway where the OBD-II interface is connected. This interface is used to update the software of the ECUs locally by using a diagnostic tool. The second method to update the vehicle is via over-the-air updates. In that case, the car is connected via a telematics unit to the Internet. The telematics unit is itself connected to the gateway to distribute the data in the vehicle and send relevant information in the other direction from the car to the telematics unit. Although the concrete architecture of the ECUs can vary from vehicle to vehicle, the proposed concepts are still valid. The tasks of the gateway can be taken by another ECU, depending on the concrete vehicle architecture. However, it is necessary to choose a central, dedicated ECU which bridges all different networks in the vehicle.

The gateway is equipped with a TPM that acts as a hardware root of trust for the vehicle. The TPM serves for the measured boot and afterwards the attestation of the car. A measured boot is always performed after a power cycle of the gateway. During that process, each boot stage of the gateway will measure the upcoming boot stage. The measurement is then noted in a boot log file and reported to a PCR in the TPM. After that, the execution is passed to the next stage in the boot sequence. The first executed stage in the processor, the core root of trust measurement (CRTM), has well-known software which is certified by the manufacturer. Measured boot assures that the TPM contains, after the measurement, a digest which represents the complete booted software of the gateway. If the software in the boot chain does not change, the resulting digest will also be same. For verification, this measurement is sent to the user application, which compares it with a reference value and assures that the gateway runs the intended trusted software.

After the operating system in the gateway is booted, the restricted RSA signing key is created. For a PCR quote, the processor sends a nonce (random number) to the TPM which signs the requested TPMs values together with the nonce in one answer. This is used to securely transmit the values of the PCR registers to the user application. When the gateway starts, the signing key is generated for the attestation, using a persistent seed and constant template. Every time this procedure is repeated, the key generation creates an identical key provided that the template and seed does not change. This key always needs to be same because the manufacturer certified this key in the initialisation step. TPM only allows to store the seeds needed to generate the key but not the key itself. After the measured boot and initialisation of the necessary keys, the gateway can execute its regular tasks the same as it would be without VeSIPreS implemented (see Figure 5 for a detailed message flow diagram of the boot sequence). The owner or any user can now send a measurement request via the user application to the gateway. The gateway receives this command and encodes it. When a full measurement of the vehicle is requested, the gateway initiates the status poll from all ECUs and sends them together with its own boot log and the PCR quote back to the user application.

6.3. ECUs. ECUs designate control units inside the vehicle that are connected to the gateway using internal networks. A typical car consists of dozens of different ECUs which perform various tasks in the car ranging from engine control and autopilot to comfort functions such as mirror adjustment. In VeSIPreS, they receive requests from the gateway, which then also receives their answers. Since they are located inside the car and are only accessible through the gateway from interfaces to the outer world, they are not as prone to attacks as the gateway is. Depending on their security relevance, it is however possible to add further protection to them. Three different security levels are therefore proposed for the ECUs:

(i) SL: the first security stage (security stage low) designates ECUs with no further protection by an additional cryptographic coprocessor such as an HSM or TPM. Since this stage offers the cheapest and easiest implementation, it can be used for all controllers where no special security requirements are needed. This includes most comfort functions in a vehicle.

(ii) SM: the second security stage (security stage medium) contains ECUs which are equipped with an HSM coprocessor. HSMs can hold cryptographic keys which are used for signing and encrypting messages. Many modern processors in ECUs have already embedded HSMs which can be used to cheaply add this security level to a substantial part of

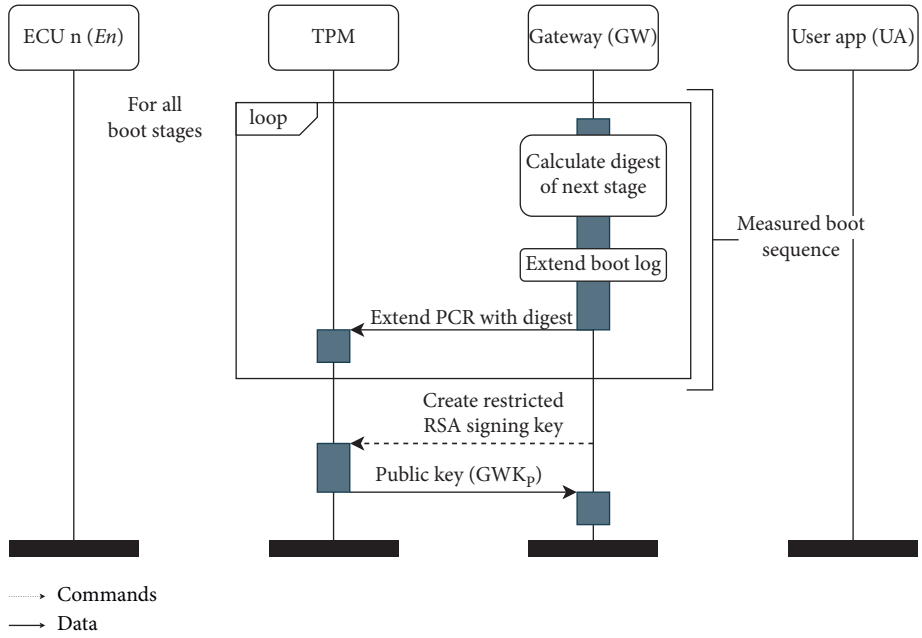

Figure 5: Message flow diagram of the starting sequence for the single components.

the ECUs in the vehicle. This allows them to sign the communication between the ECU and the gateway or another ECU. Signing the messages prevents man-in-the-middle attacks, however, also increases the computational effort and the transmitted data on the bus. To avoid such drawbacks, only sensitive messages such as status polls from the gateway are signed. This stage should be used for all ECUs which are relevant for the driving or other safety/security aspects of the car, such as the central locking system or power steering controller.

(iii) SH: the group which offers the highest protection (security stage high) contains ECUs that are, such as the gateway, equipped with a TPM. This allows them to perform a measured boot like the gateway does and get so trusted measured boot records which assure that they run the software which is intended to run on them. For this application, TPM offers the same functionalities as an HSM provides in security stage SM. That means that the measurements and other critical messages are sent signed between the gateway and ECU. A TPM additionally offers the measured boot capability to its hosting ECU. This stage should be used for highly critical ECUs such as an autonomous driving computer or brake controller.

6.4. User Application. In the user application, the measurements of the vehicle are initiated, stored, and compared. On the main screen of the app, a list with all stored measurements from the vehicle is shown. The user can take a new measurement of the vehicle from there or compare two existing measurements to detect changes between them. In a typical use-case scenario, the owner of the car would take the first measurement before it gets rented out for the first time. This measurement is automatically stored in the app and used as the reference for all upcoming measurements. When the software in the vehicle changes for trusted reasons, i.e., after a software update, the user can take a new reference measurement afterwards and use that instead. Whenever the owner wants to check the status of his vehicle, he can take a measurement in the app and compare it with the reference measurement. When a new measurement is started, VeSIPreS will perform the following steps:

(1) First, the app establishes a connection with the vehicle. This can be done via different interfaces. For this implementation, a TCP/IP connection via a Wi-Fi connection is used. The app generates a random number (so-called nonce, N_G) which is then sent together with the measurement request back to the gateway. The nonce is used to quote the measurement and assure so the freshness of the returned measurement as it prevents replay attacks.

(2) At starting a new measurement, the gateway performs the measurement and sends an answer back to the user application. After the transmission is completed, the TCP/IP connection is closed.

(3) The app first verifies the signature of the received quote using the certificate of the signing key. This key is itself signed by the car manufacturer. The response from the gateway contains the TPM quote, boot log, and report of the measurement from the other ECUs. The boot log contains a measurement for each boot stage. The digest is calculated and compared with the PCR values to verify the authenticity of the report.

(4) If the log is authentic, the two values must match. The quote from the TPM contains all requested PCR values signed with the signing key of the gateway GK,

together with the nonce from the app (N_G). The app calculates the expected-value for the PCRs using the single measurements noted in the boot log. If they match, the boot log is trusted and gets, together with the report of the ECU status, stored on the smartphone.

For the verification of the boot sequence, it is not possible to just compare the generated digest stored in the PCR register with the reference one. Since the boot order of some components could eventually change or individual modules get updated, the entire PCR value would change and cannot be used to compare it with a previous recorded PCR digest. The verification is rather done by comparing the boot log with the previously taken reference boot log. With the individual measurements written in the boot log, the value that the PCR register should have can be calculated and then compared with the actual value of the register. If they match, it means that the boot log can be trusted and is not manipulated.

To check the status of the vehicle, stored measurements are compared among each other. In the considered scenario, the current measurement is compared with the reference measurement. A measurement consists of many different individual parameters. If a compared value is not equal to the reference value, it can be assumed that the software in the component which does not match has changed. If, however, the digest of the gateway does not correspond with previous measurements, it means that something in the executed code during boot time has changed. The gateway measurement comes from the measured boot and is a digest of the code that is booted, so if it varies, it means that there is another code running than during the reference measurement. Since the gateway is the central component that manages the measurements of the other ECUs, we can only trust the vehicle measurements if the measurement of the gateway is correct. That means that if the measurement of the gateway indicates manipulation of it, we cannot trust the other measurements as they are processed by the gateway (which is potentially manipulated).

6.5. Protocol. The protocol used for the data exchange is initiated by the user in the application. A detailed message flow diagram of this process is given in Figure 6:

(1) When the user starts a measurement, the user application sends a RequestMeasurement command together with a random number, the nonce (N_G) to the gateway. The communication between user application, which runs on a mobile device and gateway in the vehicle, is established via a TCP/TLS connection.

(2) After the gateway receives the command, it starts the measurement process. The gateway requests from the TPM a quote of the PCRs which were used during the boot sequence. This command includes N_G, which has afterwards to be included by the TPM in the signed answer.

(3) In parallel, a status request is sent to all ECUs in the vehicle. Depending on the security level of the ECUs, a nonce N_{En} is attached to the request or not. The nonce is added to the requests for ECUs with security level SM and SH to sign the report. It is additionally used by ECUs with SM level to quote the PCRs used for the measured boot.

(4) When an ECU receives the request, it will perform a self-check and create a report which contains system errors, measurements of predefined parameters, and measurements of important software components. An ECU with security stage SL will then directly send this report to the gateway. ECUs with security stages SM and SH sign the report including the received nonce first in the HSM, or TPM, and then send the signed message to the gateway. In stage SH, the ECUs send additionally a quote of the PCRs to the gateway. The PCRs are signed together with the received nonce by the cryptographic signing key. If an ECU does not respond in a certain time, it is assumed that it is either removed, compromised, or malfunctioning. Such an event will, in any case, be noted in the report.

(5) The gateway will then verify the signatures of the received reports. All the received reports are then combined with the boot log of the gateway, additional values, and the nonce N_G to a combined report which is signed by the TPM and sent back together with the quote from TPM to the user application.

7. Proof-of-Concept Implementation

A VeSIPreS proof-of-concept prototype is implemented with the motivation to demonstrate the functionality of the solution, show the practical feasibility, and identify during the development and testing eventual design flaws in the solution. The implementation consists of two parts: the first part is the vehicle simulator. The vehicle simulator represents all sequences and tasks that happen inside the vehicle and emulates the gateway and ECUs. It necessarily needs a TPM to run. This service can either be provided by a TPM simulator running on the same host or by using a real TPM the host is equipped with it. For testing, the TPM simulator was used as it is more practical and flexible for different test scenarios. The architecture of VeSIPreS in a physical car consists of software running on the ECUs and software running on the central gateway. In the proof of concept, all the different components in the vehicle are integrated into one application which offers the same interface as a physical car would do.

The second part is the user app, which runs on a mobile device. The app connects to the vehicle simulator and initiates, manages, and evaluates the measurements. During the development of the software, the focus was to work with a platform that is widely used in the automotive industry and can be ported easily without significant code changes. The app is the same as also for real use outside the simulator. Figure 7 gives an overview of the vehicle simulator that communicates with the user application.

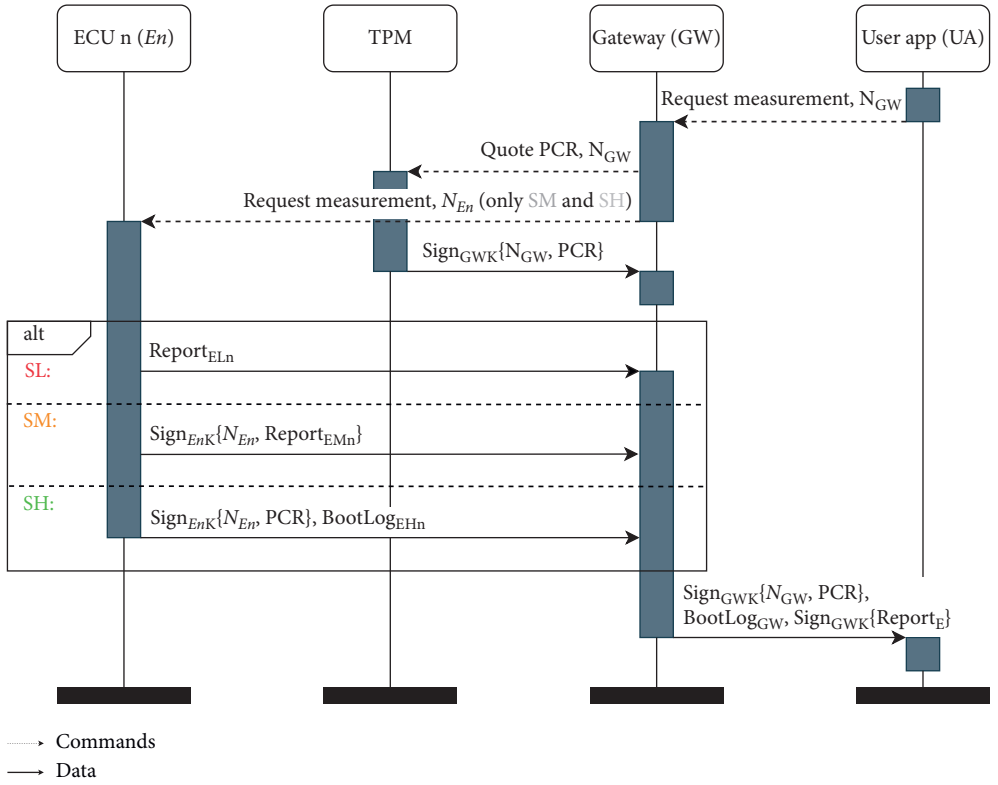

Figure 6: Message flow diagram for taking a measurement of the vehicle, triggered by the user application.

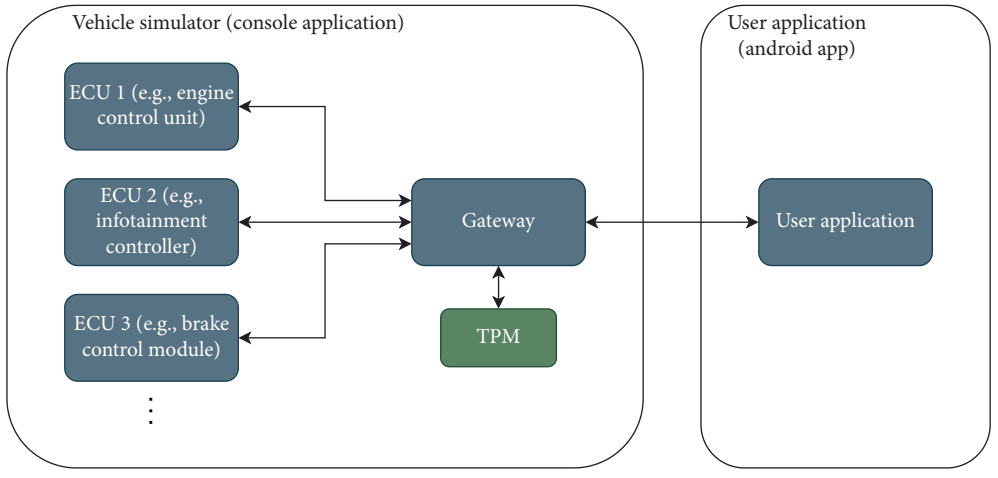

Figure 7: Model of the developed proof-of-concept prototype, consisting of the vehicle simulator with some ECUs and the central gateway and the user application.

7.1. Used Implementation Environment. The vehicle simulator is written in C++ to run as a console application on a Linux machine. C++ is one of the most used programming languages and together with Linux widely used in the automotive industry. It was furthermore beneficial as also the used API, the so-called TPM Software Stack (TSS), is written in C such that it allows an easy implementation of it in the vehicle simulator. That allowed rapid prototyping and a secure TPM implementation in the code. The user application was written in Kotlin and developed in Android Studio. The communication between the user application and the vehicle simulator is established via a TCP/IP connection whereby the vehicle acts as the server and waits for incoming connections from the app. The used communication protocol for the command and response message is implemented in a JSON standard format.

The complete source code can be accessed via GitHub at https://github.com/valaenthin/VeSIPreS_Gateway (vehicle simulator) and https://github.com/valaenthin/VeSIPreS_Client (user app).

7.2. PoC Initialisation. The gateway must know already prior to the attestation about which ECUs are in the vehicle network and which security level they are equipped with to prepare and address the command messages correctly. Based on the stored information, a measurement is requested from all ECUs individually, based on their cryptographic capabilities (security stage). Since the vehicle simulator represents all ECUs and the gateway in one application, it directly takes the measurements (taken from a file that represents the corresponding ECU) with no further encryption or signing with a nonce. This would, however, be needed for implementation in a real vehicle.

During manufacturing, the vehicle manufacturer initialises every ECU. Depending on whether the ECU is equipped with a HSM or TPM (security level SM and SH), it creates and initialises the key pair used for symmetric-key encryption. The public keys of all used ECUs are stored in the gateway, such that there is no need to transmit them over the vehicle network. During manufacturing, the certificate for the TPM in the gateway is created and signed by the OEM.

7.3. Vehicle Simulator. A modern vehicle consists of hundreds of different ECUs which handle the controlling tasks and are interlinked by various internal networks. These networks are interlinked by a gateway which provides a diagnosis interface and is connected to the telematics unit. The goal of the vehicle simulator is to provide a realistic replacement for a real vehicle to test the concepts and provide a counterpart for the user application which is the same as it would be in a real car. The simulator includes all functionalities that the gateway implements in a real car, including a simulated measured boot, cryptographic key management, request decoding, and report creation. In the simulated virtual environment, single ECUs are included in the simulator such that it is one app which simulates a gateway with an arbitrary number of ECUs connected to it. The software stack of the ECUs is written in an independent file for every single ECU so that it can be easily modified to test the reaction of the system in that case using a text editor. The same as for a real vehicle, the gateway needs also in the simulator a TPM. The IBM TPM 2.0 simulator is a convenient choice for this application. Figure 8 shows the vehicle simulator and TPM simulator console output after a measurement was performed.

As discussed in [25], it is inevitable to trust the gateway, which is the central element for the measurement process. To assure the trustworthiness of the gateway, a measured boot is performed. The vehicle simulator uses the software TPM to simulate a measured boot. The measured boot is embedded in the process and assumes the sequential starting and measurement taking of the individual components that emulate a measured boot. The single steps that are executed by the vehicle simulator in one iteration are as follows:

(1) Measure boot
(2) Wait for an incoming TCP connection from the user app
(3) Handle the received command from the user app
(4) Quote PCR from the measured boot
(5) Measure ECUs
(6) Send answer back to client containing the ECU measurements and PCR quote

7.4. Implementation of the User Application. The user application is a smartphone app (Figure 9) for Android operating systems that allows the user to manage all measurements taken from a vehicle. It allows to initiate a new measurement of the vehicle, store it, and afterwards compare two measurements (reference measurement with the suspected one) among each other. The application connects via a TCP socket to the vehicle equipped with VeSIPreS or the vehicle simulator running on a desktop PC.

The app always starts on the main screen, where all stored measurements are listed and can be managed. The user can on the main screen take a new measurement (by pressing "+"). Then, the app changes in the "Take Measurement" screen. There, it first connects to the gateway in an actual vehicle or the vehicle simulator. When the connection is successfully established, it generates a random nonce which is sent together with the command to take a measurement of the entire vehicle, to it. The vehicle will now perform the measurement and send an answer back to the app. The app waits during that time until it receives an answer or eventually times out. The received answer is now checked on plausibility. Thereby the signatures are checked, and if every test is passed, it gets stored for an eventual upcoming comparison. A measurement consists of the PCR value from the gateway together with the corresponding boot log, and a report containing all other digests from the ECUs.

The user returns back to the main screen where the new measurement will show up. He can now select a previously taken reference measurement and the last taken measurement. By pressing "Compare Measurements," the app will switch to the "Compare Measurement" screen. The individual components of the measurement are compared, and the results are displayed to the user. When the digest of an ECU is the same as it was at the reference measurement, it means that the software running on that ECU is still the same. If, however, the digests vary, it means that the software has changed, and the user gets an alert displayed in the app. When the measurement of the gateway itself (the PCR values) is different, it indicates a modification of the gateway firmware. In that case, the entire car cannot be trusted since the gateway must be trusted to process all the other measurements. Figure 10 gives an overview about the app by giving a screenshot for each screen that appears in the app.

7.5. Protocol Implementation. The communication between the user application and the vehicle or, in this case vehicle simulator, is done via a standard format for data object sharing (JSON). There are especially two different messages used in VeSIPreS. A request message is sent from the user application to the vehicle simulator and contains the request

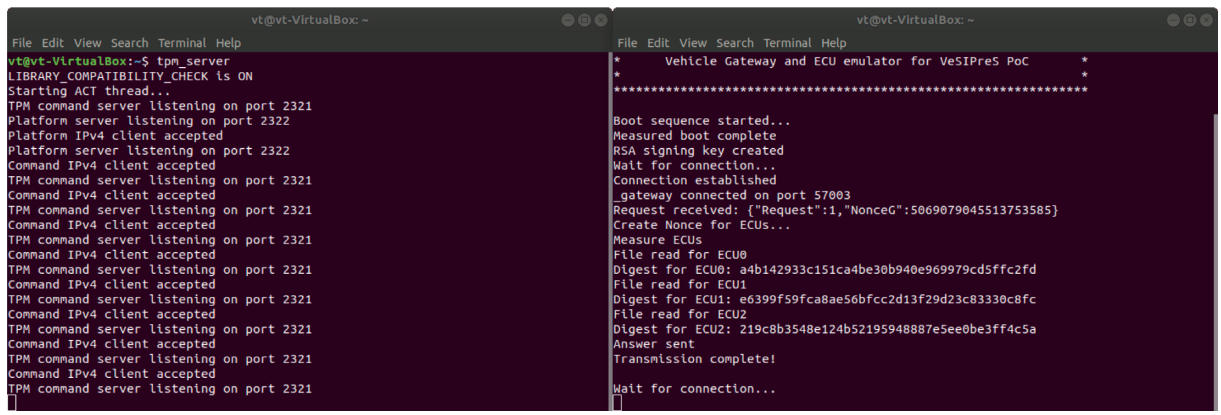

Figure 8: Screenshot of the vehicle simulation running on a desktop environment. The left console shows the running TPM simulator, and the right one shows the vehicle simulator.

Figure 9: Launcher icon of the user application that connects to the vehicle and receives the measurements. It then manages and compares the measurements.

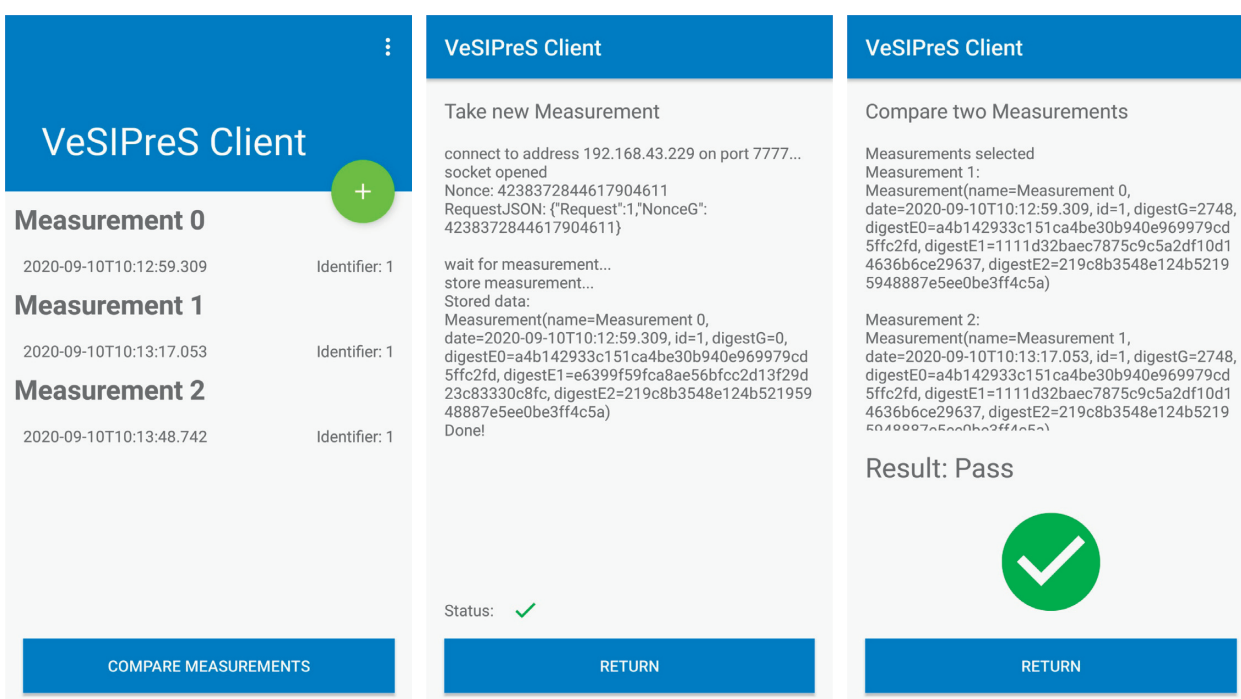

Figure 10: Screenshots from the user application with the main functions (from left to right: main screen with the possibilities to take a new measurement and compare two measurements, taking a new measurement, comparing the two measurements).

identifier that designates the type of request and a nonce used by the gateway. After the vehicle processed the command, it will answer with an answer message, containing the boot log, the PCR value(s), and a report for every ECU. The following list gives a detailed definition of the transmitted information in the request and answer message:

(i) Request message:

ID: identifier of the request type. ID 1 designates a full-vehicle measurement.
Nonce: random number used to quote the PCR register values.

(ii) Answer message:

(a) Boot log (structure is transmitted for every single entry in the log):

Description: identifier of the request type. ID 1 designates a full-vehicle measurement.
Digest: random number used to quote the PCR register values.
Digest length: length in bytes (or characters) of the digest.
PCR register: number of the register where the value is stored.

(b) PCR values:

Quote: quote of the PCR registers (PCR value and nonce signed together).

(c) ECU report (structure for each ECU):

Digest: string containing the object name, version, and other relevant information.

8. Security Analysis

This section gives an analysis of different attack scenarios on a vehicle and discusses how VeSIPreS can prevent such attacks.

8.1. Security Discussion. In [2, 29] a classification and systematic search for security flaws in the electrical architecture of vehicles is given. The terminology security can be divided into different categories. Information security consists of different parts, namely, confidentiality, integrity, availability, accountability, and assurance services. This leads to different attacks, such as man-in-the-middle attacks and replay attacks. VeSIPreS protects against man-in-the-middle attacks by encrypting the traffic. The used mechanisms are encrypting, signing, measured boot, and remote attestation.

There is to consider that the software of the vehicle may change on purpose. That happens, for example, when a software update is installed on the car. For this case, a specific procedure must be developed. Before a scheduled change is performed, a measurement is taken and verified that the vehicle is at that point (before the update) as intended. Then, the update is installed, and afterwards, a measurement is taken that replaces the reference measurement as the new one. During the update process, manipulations cannot be detected as they would after the update process be measured together with the proper software and stored as the new reference measurement. It is necessary to either perform updates only in a trusted environment (for example, under the care of a rental agent) or by implementing additional protection mechanisms.

Vehicles have a relatively long life cycle of well over 20 years. To ensure, at least to some extent, that the cars are still safe at the end of their lifecycle, the vehicle uses over-the-air updating to fix security vulnerabilities. Furthermore, the cryptographic mechanisms of TPMs can be updated remotely and offer so always state-of-the-art security mechanisms (crypto-agility).

8.2. Design Choices. In the RATS standard [25], they described two types of attestation environments which are used in this work. The vehicle is modelled as a composite device according to the definition where the single ECUs are interlinked by different networks and represent subentities. For attesting the software of the vehicle, a measurement of every single ECU in the car must be created. A lead attester, in the case of VeSIPreS the central gateway, collects all measurements and creates a report containing all measurements of the vehicle and a measurement of its own status. To trust the composite device, the lead attester (central gateway) takes the main role as the measurement of the vehicle can only be trusted if the lead attester is trusted. The central gateway (lead attester) is protected by layered attestation. The concept of layered attestation is applied only on a single ECU and measured boot therefore implemented. The gateway is equipped with a TPM that acts as the root of trust. The boot sequence of the gateway starts with a CRTM, a piece of software that is immutable and measures the next executed boot stage and passes then control to it. Starting with the first boot stage, each sequential stage measures the next stage before passing control to it. This guarantees that every executed code is measured and a representation in the form of a digest is forwarded to the TPM which extends the PCR register with the digests. Whenever the user initiates a measurement, the response contains a quote of the PCR registers so that the verifier can be sure that the gateway has not changed. If the gateway is not trusted, also all other measurements in the report cannot be trusted as the gateway could potentially modify them.

It must be assured that a tampered gateway cannot send a previously recorded measurement captured in a healthy state to the verifier and pretend so to be as intended. To assure that this will not happen, nonces are used. A nonce is a random or pseudorandom number created by the verifier. It ensures that the same message cannot be used multiple times in a replay attack. When the user app requests a measurement of the vehicle, it will include to the command a nonce. The gateway will forward the received nonce in a quote-command to the TPM which includes it in the response message and makes a signature of the message including the nonce. The generated signature is then verified by the user app using the public signing key of the gateway (provisioned by the TPM). As the private part of the signing

key is only known to the TPM and cannot be extracted, a manipulated gateway cannot imitate this message such that an eventual manipulation would be detected.

The other ECUs in the vehicle do not necessarily need such strong security like the gateway because the only way to access them to change their software is through the gateway. The gateway hosts all interfaces which can be used to install new software on an arbitrary ECU in the vehicle, such as the OBD-II port and the telematics unit. The gateway can implement a firewall and log system that can deny and log all attempts to modify the firmware of an ECU. To manipulate the software, the attacker would have to first modify the gateway to penetrate through it to install the software on an ECU. The implemented mechanisms successfully protect against attacks through the classical updating channels. They will, however, not protect against physical attacks. An attacker could, for example, attach a programming device to the internal network that connects the gateway with the ECU and bypass so the protection provided by the gateway. Although manipulation of the ECUs or networks in the vehicle is not the scope of this work, a possible extension is proposed in the form of three security stages SL, SM, and SH. ECUs with security stage SL have no cryptographic coprocessor or other cryptographic feasibilities. SM is ECUs which are equipped with an HSM or similar module that allows them to encrypt data. ECUs from these categories can encrypt critical traffic between the gateway and the ECU such that a man-in-the-middle attack on the internal bus will fail. Since encrypting the entire traffic from the ECU requires a lot of performance only sensitive data like critical measurements, firmware update or the measurement of the software are encrypted. Time-critical commands cannot be encrypted due to the induced delay. Highly sensitive ECUs should be protected by level SH. These ECUs are equipped with a TPM that allows them to perform a measured boot, like the gateway. The measurement which is taken by the ECU also contains a quote of the PCR register. The gateway produces therefore for every ECU with level SH an individual nonce which is included in the signed answer from the TPM in the ECU.

The communication between the gateway and the user application is passed through a secure and encrypted channel, for example, Transport Layer Security (TLS), to prevent attacks between these two entities.

For the ECUs to evaluate their health status, they will take in security level SL and SM a measurement of the code they run. This measurement is generated by a part of the code that could theoretically also be manipulated such that it always returns the measurement value of a known software image. However, to manipulate the component in the code that generates the measurement, the firmware must be updated, which is protected by the gateway.

A benefit of the direct communication of the user application with the gateway is that no internals of the car must be shared, and the app does not have to be adapted explicitly on a specific car model. The gateway takes the task of checking the measurements of the single ECUs and compares them with the stored reference measurements. The measurements are provided by the OEM and together with the corresponding version number stored in the gateway. The gateway can then send an attestation to the user application that the ECU has not tampered and since in the same moment also a measurement of the gateway is transmitted, it can be assured that the attestation is trusted if the gateway is secure.

8.3. Limitations. The proposed solution offers comprehensive protection against software tampering attacks. This includes the installing of new software or the modification of existing parameters. However, it protects only to a limited extend against physical tampering. If, for example, an attacker uses a man-in-the-middle attack at the internal network (CAN bus) between the gateway and the ECU, he can update the ECU without going through the gateway or modify the communication between the gateway and ECU during a measurement such that the gateway gets trusted measurements although it is compromised. This can either be achieved by installing a device in the network which always replaces the answers with trusted ones, or by updating the software of the ECU such that it always replies a trusted measurement. This is possible since the measurement of its own software stack is performed by the ECU itself. Although such a scenario is physical tampering and requires already a vast technical understanding, it is possible to protect against it by protecting the ECUs with cryptographic functions. The security stage SH can successfully prevent such attacks and security stage SM adds significant protection to prevent such attacks.

9. Future Work

The shown solution can effectively detect software modifications performed by an attacker which has physical access to the vehicle or are caused by a malfunction in the ECU that corrupted the software. For use in a real-world environment, the proposed protocol must be systematically checked for common design flaws. This is done by using already established tools such as ProVerif, Tamarin, or AVISPA. The protocol will be modelled in these tools and afterwards checked against predefined parameters. Besides the theoretical proof, it is also crucial to verify the proposed solution in a real environment in a vehicle. Although a proof-of-concept prototype consisting of the vehicle simulator and user application was developed during this work, practical testing is inevitable for further testing.

9.1. Possible Extensions in the Future. The current version of VeSIPreS is only intended for one vehicle that is measured by the app on one specific smartphone that belongs to the user. In a car-sharing scenario, however, this is infeasible. There are many cars and many users who want to use them and verify the vehicles before they use them. To solve this, a cloud-based central storage for the vehicles reference measurements is proposed. All the reference measurements are stored on a server. If the user wants to verify the software status of a car, he takes the measurement as usual. As

reference measurement, he does not use a previous measurement stored on the phone, but rather download the reference measurement from the car-sharing provider. This extension brings more flexibility in the system and can be implemented without significant changes.

A requirement is that the reference measurements are securely stored in the cloud. If they are not trusted, the entire attestation cannot be trusted. Keeping the references on a central cloud server is a potential weakness as if an attacker can compromise the stored reference values, he can also change the software in the vehicles without noticing by the app. To prevent this, blockchain technology [30] can be used to secure this weak link. Blockchains allow a decentralized storage of reference measurements where the suppliers from the single components in the vehicle can add the reference measurement for their components software directly without a central entity that collects and stores the measurements.

A further possible extension includes the use of physical unclonable functions (PUFs). This allows to secure additional sensors and actuators which cannot be fitted with a TPM against tampering. Implementing these extra safety mechanisms allows to verify an even more comprehensive part of the vehicle and protects against some physical tampering attacks. VeSIPreS provides a framework to add these additional security mechanisms.

9.2. Perspectives Involving OEMs and Mobility Services. The proposed solution needs, besides the implementation of additional software for the measurements in the software of the gateway and ECUs, no further support by the manufacturer or mobility provider. However, there can be two significant additional possibilities with a collaboration of the OEM:

(1) Instead of taking a measurement before the car is rented out as the reference measurement, a reference can be provided by the manufacturer. The manufacturer will therefore provide for different versions and the different ECUs in the vehicle the digest they should produce if everything is intended. This allows not only to measure the integrity over a certain period but rather over the whole lifetime. Additionally, measurements can then be taken by the user and compared with the reference provided by the OEM. This allows that the owner has not to take a reference measurement. This would further increase the use case as it is then possible for a user without trust in the previous user or owner to check the status. An example will be if a private person buys a second-hand car from another person. He can check the software of the vehicle with the reference from the OEM and can so be sure that the software has not changed.

(2) Whenever it comes to a manipulation or tampering attack, the OEM can get a detailed report about the occurrence. After a manipulation is detected, the gateway collects all relevant data, eventually anonymizes them, and sends the bundle encrypted to the car manufacturer. The OEM can then evaluate the occurrence and identify potential security flaws in the vehicle. If necessary, he can then react to it by releasing an update with fixed security vulnerabilities.

The gateway includes nonvolatile storage which holds all certificates and is set up by the OEM. The stored certificates contain the public key of the ECUs that can communicate protected (security level SM and SH).

10. Conclusion

This work looked at different attacks on vehicles in the past and identified the software stack of a car as a huge attack surface. OEMs make a big effort by securing the processes and close newly found vulnerabilities. Still, there is yet no actual possibility to check the software stack of a car to be sure that there is no manipulated software running on an ECU in the vehicle. Especially regarding the trend towards shared mobility, where multiple users use a car, and the fact that computers take an increasingly more critical role in the vehicle, a mechanism to check the software running on the car gets essential. An attacker has physical access to the car, which allows him to use many more attack vectors than he would have without physical access. Especially, the diagnosis interface (OBD-II) allows the installation of modified software on the ECUs. By using a TPM as a root of trust for cryptographic operations, we proposed a solution to take at any time a reliable measurement of all software stacks in the vehicle. In the electronic architecture of a car, the central gateway is the module that connects different ECUs and interfaces among each other. The OBD-II diagnosis interface is directly connected to the gateway, which forwards the commands. Implementing a TPM in the gateway allows it secure boot and remote attestation of the gateway. A mobile app allows the user to take a measurement of the entire software stack of the car, store it, and compare it with other measurements if they are still the same. The app connects via a secured connection (can be local Wi-Fi or cloud-based) to the gateway, which is trusted due to the measured boot. The gateway will then perform the measurements of all ECUs in the car and send the result back to the app. The implementation of a proof of concept based on a virtual vehicle (vehicle simulator) running on a desktop PC and the user application showed the practical feasibility and effectiveness of the solution.

Data Availability

No datasets were used. The source code can be accessed via GitHub at https://github.com/valaenthin/VeSIPreS_Gateway (vehicle simulator) and https://github.com/valaenthin/VeSIPreS_Client (user app).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work of Valaenthin Tratter was partially funded by the European Union's Erasmus SMP Program. This work has received funding from the European Union's Horizon 2020 Research and Innovation Programme through the nIoVe Project (https://www.niove.eu/) under grant agreement No 833742.

References

[1] G. Laporte, F. Meunier, and R. Wolfler Calvo, "Shared mobility systems: an updated survey," *Annals of Operations Research*, vol. 271, no. 1, pp. 105–126, 2018.

[2] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, p. 148, 2019.

[3] J. Petit and S. E. Shladover, "Potential cyberattacks on automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 546–556, 2014.

[4] C. Riggs, C.-E. Rigaud, R. Beard, T. Douglas, and K. Elish, "A survey on connected vehicles vulnerabilities and countermeasures," *Journal of Traffic and Logistics Engineering*, vol. 6, no. 1, 2018.

[5] K. Koscher, A. Czeskis, F. Roesner et al., "Experimental security analysis of a modern automobile," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pp. 447–462, IEEE, Oakland, CA, USA, May 2010.

[6] A. Greenberg, "Hackers reveal nasty new car attacks–with me behind the wheel (video)," 2013, https://www.forbes.com/sites/andygreenberg/2013/07/24/hackers-reveal-nasty-new-car-attacks-with-me-behind-the-wheel-video/#2370c801228c.

[7] S. Checkoway, D. McCoy, B. Kantor et al., "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the USENIX Security Symposium*, vol. 4, pp. 447–462, San Francisco, CA, USA, August 2011.

[8] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *Black Hat USA*, vol. 2014, p. 94, 2014.

[9] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, p. 91, 2015.

[10] Fiat Chrysler recalls 1.4 million cars after Jeep hack, 2015, https://www.bbc.com/news/technology-33650491.

[11] A. Greenberg, "This bluetooth attack can steal a Tesla model X in minutes," 2020, https://www.wired.com/story/tesla-model-x-hack-bluetooth/.

[12] M. Rogers and K. Mahaffey, "How to hack a Tesla model S," *DEFCON*, vol. 23, pp. 37–116, 2015.

[13] Z. Cai, A. Wang, W. Zhang, M. Gruffke, and H. Schweppe, "0-days & mitigations: roadways to exploit and secure connected BMW cars," *Black Hat USA*, vol. 2019, p. 39, 2019.

[14] L. Wouters, E. Marin, T. Ashur, B. Gierlichs, and B. Preneel, "Fast, furious and insecure: passive keyless entry and start systems in modern supercars," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 3, pp. 66–85, 2019, https://tches.iacr.org/index.php/TCHES/article/view/8289.

[15] S. Kamkar, "Drive it like you hacked it: new attacks and tools to wirelessly steal cars," *DEFCON*, vol. 23, 2015.

[16] R. Petri, M. Springer, D. Zelle, I. McDonald, A. Fuchs, and C. Krauß, "Evaluation of lightweight TPMs for automotive software updates over the air," in *Proceedings of the 4th ESCAR USA*, Detroit Metropolitan, MI, USA, June 2016.

[17] S. McMahan, "Volkswagen deploys infineon TPM 2.0 for vehicle communication security," 2019.

[18] P. Kleberger and T. Olovsson, "Protecting vehicles against unauthorised diagnostics sessions using trusted third parties," in *Proceedings of the International Conference on Computer Safety, Reliability, and Security*, pp. 70–81, Springer, Toulouse, France, September 2013.

[19] G. Kornaros, D. Bakoyiannis, O. Tomoutzoglou, M. Coppola, and G. Gherardi, "Trustnet: ensuring normal-world and trusted-world can-bus networking," in *Proceedings of the 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–6, IEEE, Beijing, China, October 2019.

[20] S. Halder, A. Ghosal, and M. Conti, "Secure OTA software updates in connected vehicles: a survey," 2019, https://arxiv.org/abs/1904.00685.

[21] M. S. Idrees, H. Schweppe, Y. Roudier, M. Wolf, D. Scheuermann, and O. Henniger, "Secure automotive onboard protocols: a case of over-the-air firmware updates," in *Proceedings of the International Workshop on Communication Technologies for Vehicles*, pp. 224–238, Springer, Oberpfaffenhofen, Germany, March 2011.

[22] A. Fuchs, C. Krauß, and J. Repp, "Advanced remote firmware upgrades using TPM 2.0," in *Proceedings of the IFIP International Conference on ICT Systems Security and Privacy Protection*, pp. 276–289, Springer, Gent, Belgium, May 2016.

[23] J. Howden, L. Maglaras, and M. A. Ferrag, "The security aspects of automotive over-the-air updates," *International Journal of Cyber Warfare and Terrorism*, vol. 10, no. 2, pp. 64–81, 2020.

[24] Y. Zhou, X. Wu, and P. Wang, "Secure software updates for intelligent connected vehicles," *Electrical Engineering and Computer Science (EECS)*, vol. 3, pp. 109–112, 2019.

[25] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote attestation procedures architecture: internet-draft draft-ietf-rats-architecture-07, internet engineering task force," In press, 2020.

[26] P. Kleberger and T. Olovsson, "Securing vehicle diagnostics in repair shops," in *Proceedings of the International Conference on Computer Safety, Reliability, and Security*, pp. 93–108, Springer, Delft, Netherlands, September 2014.

[27] A. Nasser, *Securing safety critical automotive systems*, Ph.D. thesis, University of Michigan-Dearborn, Dearborn, MI, USa, 2019.

[28] G. Kim and I. Y. Jung, "Integrity assurance of OTA software update in smart vehicles," *International Journal on Smart Sensing & Intelligent Systems*, vol. 12, no. 1, 2019.

[29] M. Ring, J. Dürrwang, F. Sommer, and R. Kriesten, "Survey on vehicular attacks-building a vulnerability database," in *Proceedings of the 2015 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 208–212, IEEE, Yokohama, Japan, November 2015.

[30] J. Kang, R. Yu, X. Huang et al., "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4660–4670, 2018.