DOI: 10.1007/s41635-019-00076-8

Noname manuscript No.
(will be inserted by the editor)

External Monitoring Changes in Vehicle Hardware Profiles: Enhancing Automotive Cyber-Security

Constantinos Patsakis · Kleanthis Dellios · Jose Maria De Fuentes · Fran Casino · Agusti Solanas

Received: date / Accepted: date

Abstract As the vehicles are gradually transformed into the connected-vehicles, standard features of the past (i.e. immobilizer, keyless entry, self-diagnostics) were neglected to be software updated and hardware upgraded so they do not "align" with the cyber-security demands of the new ICT era (IoT, Industry 4.0, IPv6, sensor technology) we have stepped into, therefore introducing critical legacy IT security issues. Stepping beyond the era of common auto-theft and "chop-shops", the new wave of attackers have cyber-skills to exploit these vulnerabilities and steal the vehicle or manipulate it. Recent evolution in ICT offered automotive industry vital tools for vehicle safety, functionality and up to 2010, theft prevention. However, the same technologies are the ones that make vehicles prone to cyber-attacks. To counter such attacks, this work proposes a unified solution that logs all hardware profile changes of a vehicle in a blockchain, to manage control and allow only authenticated changes, subject to user, time, geospatial and contextual constraints exploiting several blockchain features. Testing of the proposed solution omens the prevention of numerous commons attacks, while additionally provides forensics capabilities and significantly enhancing the security architecture of the vehicle (respecting the original IT architectural design of automotive manufacturers).

Constantinos Patsakis and Kleanthis Dellios and Fran Casino
Department of Informatics, University Piraeus, Piraeus, Greece
E-mail: kpatsak@unipi.gr

Jose Maria De Fuentes
University Carlos III of Madrid, Avenida de la Universidad, 30, Leganes, Spain

Agusti Solanas
Smart Health Research Group. Department of Computer Engineering and Mathematics. Rovira i Virgili University. Catalonia. Spain

1 Introduction

Undoubtedly, the automotive industry is one of the most critical sectors of the worldwide economy. According to the International Organization of Motor Vehicle Manufacturers; known as the "Organisation Internationale des Constructeurs d'Automobiles" (OICA) [59], 2012 was the first year that the automotive production exceeded the barrier of 80 million cars. Apart from the vast increase in the production rate, which introduces many changes in the market, the automotive industry is going through a radical transformation over the past few years. Vehicle control has been shifting from mechanical to electronic [18]. Therefore, modern vehicles can be considered as an operating platform within a collaborative environment [13], simultaneously running tasks and functions of many embedded subsystems, interacting inside a networked system [53].

While this transformation is gradually becoming more obvious, the automotive industry has not fully succeeded in providing the necessary security mechanisms, many of which are applied for decades in common information systems. This might be attributed to the market pressure towards embedding more features in vehicles, while the production needs the implementation of these features as soon as possible to put their products in the market before other competitors. Nevertheless, this policy might endanger the safety of passengers under certain circumstances. Moreover, unlike common information systems that can often receive both software and hardware updates, doing the same in vehicles is not always an option. Therefore, modern vehicles can be considered even more vulnerable to cyber attacks.

Added features can improve the driving experience and comfort of drivers, but they also extend the attack layer of the vehicle. Attacks can be launched by connecting devices to the vehicle's communication ports or even wirelessly. Under certain circumstances, the attackers can gain full access to the vehicle; therefore, we do not only have to consider the case of a car being stolen, but also the cases where the goal is to harm the passengers or even the transportation infrastructure in large-scale automated attacks [28].

1.1 Main contribution of the article

Despite the security measures that are taken by manufacturers and the wide range of after-market products that attempt to provide additional security to the vehicles, statistics and everyday experience prove that vehicles are still stolen. Moreover, there are several reports of automated attacks that can be launched against vehicles, e.g. in situ [40,20] or even remotely [76,50,51]. Therefore, not only vehicles can be stolen, but drivers and passengers are exposed to other risks as well. The only authentication that current deployed vehicle architectures support is between the "key" (traditional or wireless, biometrics of the drivers, etc.) and the immobilizer. The devices or Electronic Control Units (ECU) do not provide any further authentication. Even AUTOSAR [8] the most widely used framework for developing vehicle hardware components has only recently started to support encryption algorithms, which are not mandatory.

A wide range of these new attacks stems from the fact that devices and ECUs can be easily connected to modern vehicles, thus providing the attackers with a broader attack layer. For instance, several sources indicate that for the vast amount of cars that are currently being stolen, the method is the penetration through the electronic system [68,78]. To address this problem, we propose a novel three-layered scheme to manage Electronic Control Modules (ECM) or ECUs and device installation and maintenance. According to our scheme, a vehicle has a unique profile for each of its legitimate users. For every hardware change, whether this is the installation of a new device or the installation of a new firmware, the immobilizer connects to an Internet service to update its profile. Note however that firmware updates may provide undocumented functionality to a module which actually means that it treats as a new device. The service checks whether this update should be allowed and triggers the appropriate alarms. This is achieved by the use of a private blockchain and a smart contract, which adds another security layer, enabling secure control version management. Therefore, our scheme provides an additional layer of security with minimal overhead, as the experimental results clearly indicate.

The dependence of our scheme on an Internet connection might be considered a considerable constraint, however, Internet is available almost anywhere in Europe and the US, and all the population centers in the rest of the world[1]. Therefore, our solution can easily cover the vast majority of places where an actual repair should be authorised.

The novelty of this work is not the introduction of new cryptographic primitives or authentication methods, but the introduction of a new framework which uses well-known methods in a very demanding industry sector such as the automotive industry. This work is extending [63,48] and can be considered as a step towards the unification of mechanics' databases in an online database that is used to provide an extra security layer. Therefore, the description of the framework does not provide details about the algorithms or the protocols that might be adopted, but rather the functionality that they should provide. We have implemented the proposal based on several well-known cryptographic primitives and protocols as well as the Ethereum blockchain. While the cryptographic primitives might not be optimal, they are currently industrial standards and well-known for the security that they offer. Moreover, they indicate the necessary processing and time overhead that are introduced by the proposed solution, which can be considered minimal.

1.2 Organization of this work

The rest of the article is organized as follows. In Section 2, first, we provide an overview of the related work focusing on the ICT side of modern vehi-

[1] `http://www.businessinsider.com/this-world-map-shows-every-device-connected-to-the-internet-2014-9`

cles, how it can be attacked and some countermeasures. Then, we discuss the use of Blockchain on vehicles. Section 3 introduces the proposed architecture, while Section 4 discusses several issues in terms of availability, security and privacy. Then, in Section 5 we illustrate the efficacy of our proposal through experimental results. Finally, the article concludes with some final remarks.

2 Related work

2.1 An ICT approach to the vehicle

Today's high-end automobiles contain over hundreds of embedded processors, sensors, multiple wired-buses and data acquisition/telemetry components, all optimized for robust functioning and safe driving whilst continuously interacting with the mechanical parts and components of the vehicle. This mechatronic nature is attributing the cyber-physical essence of the modern vehicle.

The automotive-platform is initiated after the immobilizer has been released, triggering the electronic ignition system that the motor-engine can be safely started. ECUs can read input signals from sensors (e.g. speed, motor temperature, fuel consumption and load data) but since many sensors' signals are used for multiple functions, it is a common practice for manufacturers to place them under the Engine Management System's (EMS) control. The latter is designed to ensure that the vehicle complies with emissions regulations and to provide improved performance. Furthermore, the sensors provide the input signals, and the actuators are the electromechanical devices (e.g. fuel injectors, ignition coils, ABS modulators) that use the output results of the input sensor readings. The integration of automotive controlled systems increased the need for embedding network interfaces to the automotive platform (both hard-wired and wireless) to satisfy the future requirements for bandwidth and performance needs and led to the introduction of a highly heterogeneous network system within the automobile. The *essential network protocols* for the *in-vehicle communication* include the Controller Area Network (CAN), the Local Interconnect Network (LIN), the Media Oriented Systems Transport (MOST) and the FlexRay Protocol.

Both the architectural design and the network protocols described above, realize the automotive platform of the modern vehicle [25] a mobile-system capable to utilize engineering, technological, mechanical and networking aspects under the context of a traditional ICT system. Thus, the automotive ICT stack consists of various engineering, technological, mechanical and networking modules. Its cyber-physical structure comprises the following layers: The *automobile physical layer*, which includes the infrastructure, i.e. the body chassis whose user is the operator/driver of the automobile or the passengers; the *automotive cyber layer* which comprises the infrastructure (hardware and the peripheral components), the network (communication and network interfaces), the services (software used) and the data (signals, frames and packets) exchanged among the components of the vehicle. Figure 1 illustrates the paradigm of the automotive cyber layer based on the backbone structure of the FlexRay protocol. The interested reader may refer to [75,39,15] for additional information related to automotive communication protocols.

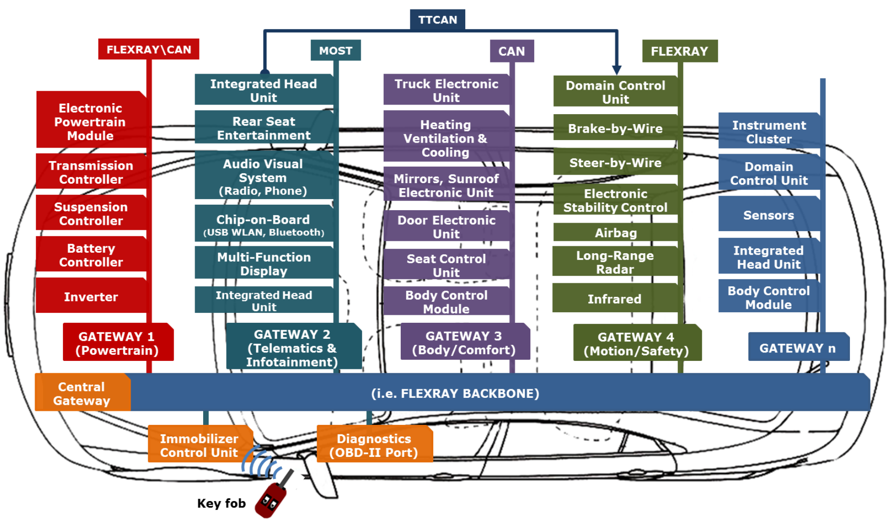


Fig. 1 Paradigm of the automotive network architecture.

2.2 Vehicle security

In many cases, the implementation of cryptographic primitives in the automotive industry has proven to be inefficient to to prevent automotive thefts [55, 36,11,27,14,76,32,33]. Moreover, researchers have proved that vehicles can be exploited in many ways as the attack surface is quite large [40,20,30,72,67,31]. As shown in the recent attack of Miller and Valasek [51], interdependencies allow, e.g. WiFi connections of the infotainment system to push updates to the core system and send arbitrary commands through the CAN, taking control the vehicle remotely, with a similar approach being exploited by Mahaffey and Rogers [47] also to gain full control.

Weimerskirch et al. [77] were among the first to propose an architecture for securing in-vehicle communications. In this architecture, all the underlying networks are connected to a gateway, which routes their encrypted messages. This scheme manages to protect the communication between heterogeneous networks; nevertheless, the traffic inside each network is still transparent to the connected ECUs.

Chavez et al. [19] foresaw the development of security threats for vehicles, and thus proposed the replacement of CAN, by an OSI-like architecture. Their proposal suggested the use of several encryption algorithms, however mutual authentication of ECUs or even non-repudiation were not covered.

Nilsson et al. in [54] introduced a novelty for in-vehicle communications, that of authenticating the components that communicate over CAN using Message Authentication Codes (MACs). Even if the use of strong encryption algorithms was proposed, mutual authentication of the ECUs fell behind again, and the model is vulnerable to replay attacks. MACs are also used in [73] and [74] where a protocol for the authentication of messages with multiple simultaneous destinations is proposed.

Groll and Ruland, in [35], categorize ECUs according to their trust level into discrete communication groups. Each group receives a symmetric key from a Key Distribution Center (KDC) and uses it for its communication. The vehicle manufacturer creates and signs an Access Control List (ACL) that is distributed to the ECUs that maintain a local copy of the ACL of the group to which they belong to. The KDC and ECU communicate through public key encryption algorithms, and the vehicle manufacturer signs the keys.

EVITA [29] is an FP7 project whose deliverables can be regarded as a major contribution to state of the art in vehicle security, as they provide a real-world implementation of an architecture, where all the traffic is encrypted through secure and efficient encryption algorithms. The developed architecture is designed to be compliant with the AUTOSAR framework, version 3.0. The infrastructure does not provide stream ciphers, while the full hardware security module is only used for the communication of the vehicle with V2X infrastructures. The architecture does not support certificates for in-vehicle communication, and as a result, several attacks are possible. Finally, the scheme allows only ECUs which have been signed by the vehicle manufacturer.

Oguma et al. [56] proposed a formally verified attestation-based architecture that: (a) enables only valid controllers to communicate, (b) processes separately or immediately discards all unauthorised messages, (c) encrypts and authenticates all communications and (d) does not allow a single attack to endanger the whole system. Since many attacks exploit the ODB-II port and how diagnostics are sent, Kleberger and Olovsson [38] propose the use of the recent ISO 13400 standard for Diagnostics over IP with some modifications to enable secure broadcasting.

A redesign of the immobilizers was proposed to address many of the new attacks that are being launched against the vehicles' ECUs [62]. To this end, the immobilizer becomes a Trusted Third Party, where all parts authenticate via certificates. The immobilizer decides whether the engine will ignite, based on the categorization of the ECUs and whether the modules have succeeded in authenticating. Finally, the immobilizer acts as a ticketing server, granting access to each module according to predefined ACLs.

To counter malware threats, Zhang, Antunes, and Aggarwal [80] proposed the use of a cloud-assisted framework to detect and discover malicious files and traffic in vehicles. On the other hand, Mansor et al. [49] based on the firmware updates process of the EVITA project studied the security issues of these protocols and extended it to provide a more secure process.

Furthermore, BMW recently introduced a novel system that sends vehicle's usage statistics over the Internet whenever it finds the proper resources (3G connected devices, open networks etc.), to arrange the next service appointment [12]. Other vehicle's subsystems, such as event data recorders (EDR), track the GPS location and can broadcast it over the Internet. This is very helpful in case of accidents and to look for stolen vehicles. Well-known such products are OnStar[58] and AcuraLink [7]. Many vehicles, for example from GM, come with OnStar already pre-installed. All the above indicate that the automotive industry is gradually adopting a "call back home" policy to its products, mainly to ensure passenger safety and vehicle security. A more private solution for vehicle EDRs has recently been introduced in [64], exploiting the properties of timed release encryption. Unfortunately, this fragmented landscape extends even further since existing security standards, methodologies, and tools don't explicitly cover the rapidly developed automotive and ITS environment or address its specific requirements, as indicated by Dellios et al. [24].

Sethumadhavan et al. [69] regard that even if a plugged component might be an actual part of the system, this does not mean that it is benign and there is no hardware backdoor installed. To mitigate such backdoors, they propose a set of pre-design and runtime techniques which, e.g. will block the triggers that would activate the backdoor.

Finally, Mansor et al. [48] introduced a mobile application named AutoLOG which keeps a secure log of all maintenance services in the cloud.

2.3 Blockchain and the vehicle

Blockchain is a distributed append-only time-stamped data structure where non-trusting members can interact with each other in a verifiable manner without the need of a trusted authority [52]. This technology is nowadays blurring the lines between the physical, digital, and biological spheres, collectively referred to as cyber-physical systems [65,16]. Its peer-to-peer structure facilitates any type of transaction (e.g. physical assets, money, intellectual property), without mediation. Yet, its security and trustworthiness are established by a decentralised, cloud-based, independent protocol.

A relatively recent aspect of the blockchain technology is the notion of smart contracts [71] (with a full Turing complete Language) which provide the ability to perform computations within the blockchain, thus operating as a decentralized virtual machine. In essence, smart contracts are actual programs written in specific programming languages, e.g. Script in Bitcoin or Solidity in Ethereum. In this regard, smart contracts can be considered agreements between mutually distrusting participants, which are automatically enforced by the consensus mechanism of the blockchain without relying on a trusted third party. Therefore, the emergence of smart contracts opens the door to a myriad of new application scenarios [16,82,21,23], since specific functions and configurable requirements can be defined.

The use of blockchain in the vehicular industry is not novel. Nevertheless, most of the works focus on secure Vehicular Ad-Hoc Network (VANET) data and trust management [57,70,46,45,79,42], service enhancement (such as location-based information) [37], and forensic traffic data analysis to improve security, with special regard of self-driving cars [17]. Moreover, several works on the use of blockchain to enable secure VANET communications (i.e. vehicle-to-infrastructure and vehicle-to-vehicle) can be found in the literature [44,9,81,34,43,60]. In the case of [26], authors describe the benefits and advantages of the use of blockchain in the locomotive industry over centralized models and discuss the potential of such applications. They provide a high-level example of a protocol that could be used to perform remote software updates. However, they only ensure data integrity of the software, and they do not discuss possible attack vectors, challenges nor countermeasures of vehicle control version. Other works related to version control can be found in the literature [10,66], however, the potential of blockchain in this regard has yet to be explored [16]. Therefore, to the best of our knowledge, this is the first work that introduces the use of blockchain and smart contracts to provide efficient and secure version control for vehicles.

3 The proposed solution

In what follows we introduce our proposed solution. Before presenting it, we first, provide the threat model and then we introduce the main actors of the scheme along with the assumptions.

As discussed in the related work section, attackers in many occasions are expected to penetrate the vehicle, e.g. by forging credentials, exploiting cryptographic primitives, or even by exploiting software vulnerabilities of the vehicles. In this context, the adversary is either trying to penetrate the vehicle or already entered the vehicle and tries to gain more control. The question is whether these actions can be monitored to trigger further actions in the cyber or the physical layer, or at least keep track of these actions to provide feedback for further investigation and to patch the exploited attack vector.

3.1 Threat model

Generally, a vehicle is left for many hours unattended and can be physically accessed by an arbitrary number of individuals. As already discussed, many of the attacks to steal a vehicle involve the installation of a device in the vehicle or the exploitation of an existing vulnerable device, which will be used to install new firmware to another connected device as a way to escalate the privileges of the attacker. For instance, an adversary initially breaks into the vehicle, e.g. by the window and plugs a device to the ODB II port to create a valid pair of keys to start the engine or installs a vulnerable device. Therefore, in our model, we assume that the adversary has physical access to the vehicle and he may perform several actions without interruptions. Moreover, we assume that the intruder will try to install a device in the vehicle; even if it is a vulnerable one, to gain further access to the vehicle or use an existing device to install a new firmware.

3.2 Main actors and desiderata

The primary goal of the proposed solution is to block when possible, or otherwise detect, attacks originated from plugged in devices or malicious firmware upgrades. With this aim, we propose that after every hardware update or whenever a new device is detected by the vehicle, a query should be executed on an external database before the engine is allowed to ignite. The proposed infrastructure is illustrated in Figure 2 and it is analyzed in the following paragraphs.

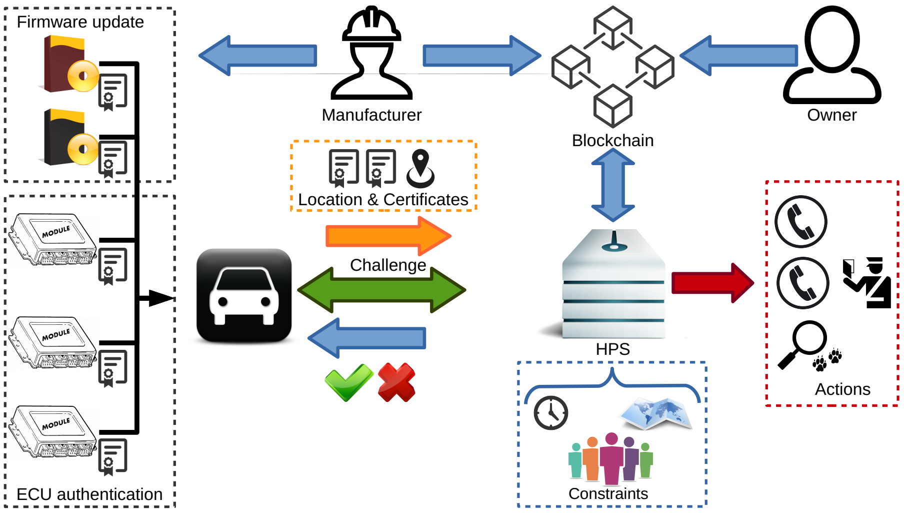


Fig. 2 The proposed infrastructure.

The framework proposed in [62] meets the needs of the vehicle architecture for in-vehicle communication, as the immobilizer module acts like a ticketing server that enables the communication between all ECUs and allows the engine ignition. Therefore, we assume that the vehicle that we refer to has this infrastructure in place. However, this model assumes that the modules are known to the immobilizer and they establish a connection proving that they have not been altered. Moreover, with the term **hardware profile**, we define a list of all the hardware parts, firmware and their corresponding certificates for a specific vehicle. We assume that each vehicle vendor creates a hardware profile for each vehicle once it is manufactured. This hardware profile is committed to a federated blockchain. This blockchain is in charge of maintaining the hardware state of each vehicle and its status regarding whether accepting updates or not.

The main actors in our scheme are:

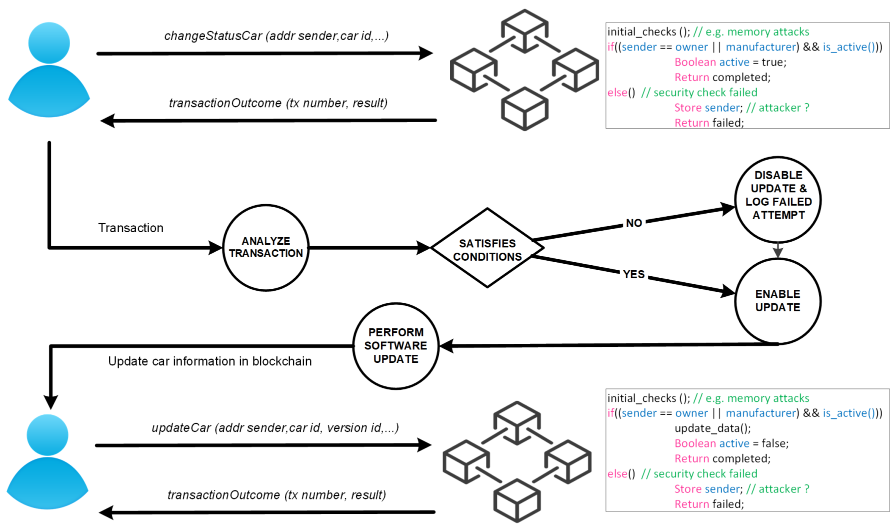
- **The User:** Whether this is the driver or someone who has rights to access the car, i.e. a passenger who is given the keys temporarily or a mechanic. A user has the credentials to access the vehicle and ignite the engine. His credentials can be something that he owns (keys), something that he knows (password) or something that he is (biometrics).
- **The vehicle:** Each vehicle is identified by its immobilizer unit. A vehicle can be accessed by many users, with different access levels for each of them. This means that not all users have the right to use all ECUs or that their connected devices might not be functional. Therefore, each user has his own hardware profile for the vehicle.
- **The Hardware Profile Server (HPS):** A trusted server with which immobilizers contact whenever they discover a newly connected device. The HPS is responsible for:
  - Retrieving user/vehicle hardware profiles.
  - Grant vehicle/user profile changes and alert the owner.
  - Checking hardware certificates.
  - Checking time and spatial constraints.
  - Sending notifications and alerts to the proper authorities (Legal, users, etc.).

We further assume that the owner of the vehicle has registered the vehicle to the HPS, hence the vehicle has its own credentials that can be used to communicate with the HPS. This communication involves mutual authentication from each side and can be facilitated by the use of a certificate from the HPS' side.

3.3 The scheme

Our proposed scheme contains three layers to counter different attacks. In the upper layer, we deploy a blockchain mechanism to keep track of the hardware of a vehicle and the installed firmware, allowing the owner of the vehicle to manage when updates to the firmware or the hardware can be performed. The next layer, inside the vehicle, tries to determine whether the ECUs that are attached are legitimate and discover new changes. Finally, an additional layer is in charge of monitoring whether the requested updates meet specific requirements, e.g. geospatial, and then proceed to actions.

When the car is sold, the data about the car is registered into a federated permissioned blockchain, and a transaction from the manufacturer is made to the address of the owner of the vehicle, and some other data structures to enable vehicle version control. This procedure provides a firmware lock mechanism that enables only the manufacturer or the owner of the vehicle to set the status of the vehicle to "*updatable*". Thereafter, to allow an update to the car, the immobilizer checks the status of the vehicle in the blockchain to determine whether the update can be performed. This way, we allow for secure version control management and perform consent management on the update preferences of the user. An overview of how the blockchain mechanism works is illustrated in Figure 3. More details about the benefits of such an approach are discussed in Section 4.


changeStatusCar (addr sender,car id,...)
transactionOutcome (tx number, result)
Transaction
ANALYZE TRANSACTION
SATISFIES CONDITIONS
NO
YES
DISABLE UPDATE & LOG FAILED ATTEMPT
ENABLE UPDATE
PERFORM SOFTWARE UPDATE
Update car information in blockchain
updateCar (addr sender,car id, version id,...)
transactionOutcome (tx number, result)

```
initial_checks (); // e.g. memory attacks
if((sender == owner || manufacturer) && is_active()))
            Boolean active = true;
            Return completed;
else()  // security check failed
            Store sender; // attacker ?
            Return failed;
```

```
initial_checks (); // e.g. memory attacks
if((sender == owner || manufacturer) && is_active()))
            update_data();
            Boolean active = false;
            Return completed;
else()  // security check failed
            Store sender; // attacker ?
            Return failed;
```

Fig. 3 Setting the update status of a vehicle through blockchain.

As in [62] and [63] the user presents her credentials, which could be a traditional key, a wireless key or biometric data. A simple protocol for the latter two cases to manage the authentication is illustrated in Figure 4. The protocol is lightweight as it only uses private key encryption, provides mutual authentication, and does not disclose the key. Initially, the user sends her ID and a nonce (r_{User}) to the vehicle. The vehicle can now query the local database to check whether this user may access to the vehicle, if that is the case, it retrieves their mutual key $k(User, Vehicle)$ and encrypts the vehicle's identity $VehicleID$, the received nonce r_{User} and a new nonce it creates $r_{Vehicle}$. If the user is valid, then she would be able to recover $r_{Vehicle}$ and form the message $E_{k(User,Vehicle)}(r_{Vehicle}, r_{User}, UserID)$. On receiving this message, the vehicle checks that the values are valid and authenticates the user.

In case of many failed authentication attempts, the system assumes that it is under attack. Therefore, two-factor authentication is triggered, e.g. a message is sent to the user's phone and an alert is stored in the database whenever there is Internet availability. This may allow authorities to create a dynamic map of places where thieves try to steal vehicles, enable users to monitor how many attempts have been made to steal their vehicles, with which user credentials etc.

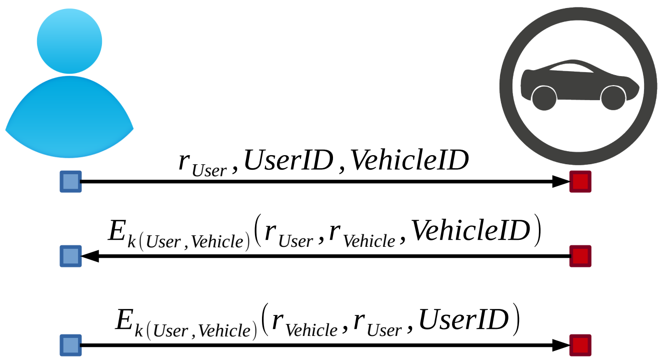

$r_{User}, UserID, VehicleID$
$E_{k(User,Vehicle)}(r_{User}, r_{Vehicle}, VehicleID)$
$E_{k(User,Vehicle)}(r_{Vehicle}, r_{User}, UserID)$

Fig. 4 User authentication

Once the user is authenticated, she has the right to access the vehicle, so the immobilizer broadcasts a "*hello*" message to all installed ECUs. All ECUs have to reply to this message; otherwise, their communication will be blocked afterwards by the immobilizer, and therefore they will not be functional. If any of the ECUs is new or it has a newly plugged device, it has to send the relevant certificate to the immobilizer. The immobilizer indicates the new hardware and its position to the user and requests a connection to the Internet to check whether the hardware changes are acceptable. Again, failure to present a valid certificate and prove its ownership results in the isolation of the module and blockage of its traffic.

If the user agrees to connect to the Internet, the immobilizer checks the update status of the vehicle. If the vehicle is set to updatable, it connects to the HPS with the user's credentials and its own certificate, and sends the certificates of the new ECUs to the server, along with the current geolocation of the vehicle. The HPS first checks the credentials of the user and the certificate of the immobilizer. To validate the latter, a challenge to which the immobilizer has to respond is sent. If the checks are successful, the HPS queries the blockchain to recover the profile of the user for the specific vehicle. Based on this profile, the HPS constructs a set of constraints, thus different users have different access levels to vehicle hardware updates, and are allowed to use specific devices. HPS then couples the aforementioned possible constraints with several time and geospatial constraints. The latter constraints are discussed in the following section. Their main role is to check the time and the position at which the hardware update is being made so that malicious attacks on unattended vehicles can be detected. If the constraints do not trigger any alarm, the HPS sends a challenge to each of the new devices to prove their origin. If the devices correctly answer these challenges, the hardware profile is approved by HPS, and the updated profile is sent for approval to the owner of the vehicle. Once she accepts the changes, the immobilizer receives the clearance to proceed with engine ignition. The clearance is given through an HPS-signed new hardware profile that is stored in the immobilizer.

The protocol for authenticating a new device to HPS is illustrated in Figure 5. Initially, the device sends a nonce $rDev$, its ID and the immobilizer's ID to the immobilizer encrypted using their shared key $K_{Dev,Immo}$. The latter can be easily generated from the public keys of their certificates using the typical Diffie-Hellman key exchange protocol. Then, the immobilizer will initiate a session with HPS to authenticate the user, send the current location and the certificate of the device. Therefore, the immobilizer sends a nonce $rImmo$, the user's credentials, the vehicle's ID and its ID, encrypted under their common key $K_{Immo,HPS}$. HPS checks the provided information, and if it is valid, it creates a hash of the previous values and a nonce $rHPS$. The immobilizer will return the received nonce, its identity, current location, the device's certificate and its nonce $rDev$. HPS extracts the device's public key PK_{Dev} from its certificate, and it sends it $rDev$, its identity HPS and a nonce r encrypted using PK_{Dev}. Finally, the device proves that its certificate is valid, by extracting r and using it as a key to encrypt the hash of $rDev$, its identity and HPS's identity. To facilitate the implementation of the last two steps, the traffic from the new device to HPS and vice versa is tunnelled through the Immobilizer.

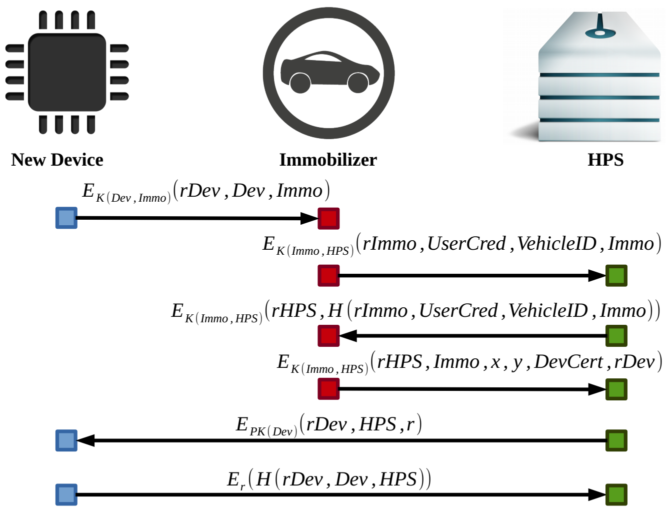

New Device
Immobilizer
HPS
$E_{K(Dev,Immo)}(rDev, Dev, Immo)$
$E_{K(Immo,HPS)}(rImmo, UserCred, VehicleID, Immo)$
$E_{K(Immo,HPS)}(rHPS, H(rImmo, UserCred, VehicleID, Immo))$
$E_{K(Immo,HPS)}(rHPS, Immo, x, y, DevCert, rDev)$
$E_{PK(Dev)}(rDev, HPS, r)$
$E_r(H(rDev, Dev, HPS))$

Fig. 5 The protocol between the Immobilizer and the HPS.

Since modern vehicles allow the installation of many peripherals, the case of peripherals such as phones, GPS receivers etc. being connected continuously and removed is very frequent. To disclose as little as possible information about users' location, on every new attempt to ignite the engine, the immobilizer checks the previously stored and signed hardware profiles to trace whether the current hardware profile previously existed. If it finds a match, it then checks the validity of the signature and decides whether to allow the engine's ignition. This last check is crucial to stop attackers from arbitrarily creating false hardware profiles to gain access to the vehicle.

4 Discussion

In this section, we discuss the main issues that emerge from the adoption of the proposed solution, with regard to availability, security and privacy, and highlight the advantages of our proposal compared to the current security standards in the vehicle industry.

4.1 Availability

One of the most important issues that stems from the proposed framework is what happens in the case when there is no Internet access or geolocation services available, and whether this would create significant availability issues.

In this case, the framework will not allow the device to operate. If the device is peripheral, then obviously this is not a significant issue, but if it is not, then the vehicle will not be allowed to ignite the engine. The reason why the latter does not introduce any important availability issues is that Internet access and GPS signal are available almost worldwide or, at least, where a vehicle should have maintenance services. We consider that the chances of needing to legitimately change, for instance, the steering wheel or the brake system somewhere where there is no Internet access, and the GPS signal is not reachable are so slim that they can be neglected. Such crucial maintenance work is typically done at a registered mechanic; if this is not the case, then the probabilities of dealing with a malicious act are very high.

Another important question is whether this approach deters people from working on their car themselves. If these modifications do not alter the hardware, then there is no problem. If someone is going to fine-tune the parameters of an ECU, there is no issue again. Finally, if someone is going to change an ECU, then he might trigger an alert, that can be easily tackled if he has notified the HPS service provider of his actions, so his profile will be successfully updated.

While one could argue that Internet connectivity can be easily reached within urban environments, this is hardly the case for rural areas. Therefore, our solution could be extended to provide "break the glass" policies to account for such cases. Such policies have already been adopted by major companies such as Google which provides emergency codes for accounts protected by 2-step authentication. In this scenario, where there is no available connection to the Internet, the owner could temporarily force device authentication using her biometrics. We argue that this authentication should not extend to more than a very limited time to deter over-usage of the feature and possible attacks.

Another important aspect is that there is no need for the HPS to be hardware-coded in the immobilizer, as long as he holds a valid certificate. This way users can have alternate HPS providers, their profiles can be transfered to others and are not bound to the manufacturer, thus introducing a new market.

In regard to the blockchain layer, the information about vehicles and transactions can be stored in IPFS[2], and retrieved using the hash function implemented in the smart contract (i.e. IPFS enables content-addressable storage) so that data tampering can be prevented. This enables efficient auditing as well as increases the trust of the system and its integrity. Moreover, the version traceability enables better management, in terms of malfunctioning detection (e.g. companies can send alerts to users who own vehicles with vulnerable software versions) as well as effective insurance claims processing. Another benefit of this approach is enhanced interoperability, since cross country data legislation, which may hinder the efficiency of update and managerial procedures, can be bypassed. This is of particular relevance in the case of critical infrastructures and public transportation since this approach enables real-time and secure permission checking. Moreover, the decentralized nature of the blockchain is resilient to attacks such as DDos, which could give attackers more control over critical infrastructures during a longer period, compared with the use of centralized data management solutions.

4.2 Security

The proposed scheme enhances the overall security of vehicles in several ways. In an attack case scenario, if there is no Internet connection or GPS signal, then the attack is blocked as any traffic from the newly plugged device is blocked so the device will not be functional or the engine will not be allowed to start. However, if both of them are available, then the device will have to prove that it is valid and does not forge any legitimate one. This enables quick detection of malicious devices that try to masquerade others. If they are legitimate and vulnerable, the next line of defense is the time and geospatial constraints. These constraints can trigger many alerts, as for example they will indicate the attempt of a thief to steal the car, as a message will be sent that there is an attempt to install, for example, a new brake system or an MP3 player at 3 am, or that someone is installing a new air-conditioning system in the middle of the road. The HSP will send a message to block the engine's ignition and, based on the constraints that are not met, it will proceed with other actions, such as calling the police, informing the legitimate user or even activating the eCall system [5].

One can argue that the proposed approach might not be able to prevent the attacker from taking control of the vehicle, as the reply from the HPS might be overruled by the attack. While this is true, in this case, several very

[2] `https://ipfs.io`

important pieces of information are collected. The most important is when and where the attack was made. Therefore, even if the software methods failed to provide the necessary security, physical methods can be applied, e.g. tracking by the police. A crucial aspect is that clear and precise attack patterns will be recorded, as the installation of a new device is showing the entry point of the attacker. Clearly, this can help manufacturers trace vulnerabilities to their source and have valid evidence, so that they do not have to base their research on assumptions and vague scenarios based on sparse data that have been collected from human witnesses or CCTV monitoring. Therefore, patching the vulnerabilities and introducing vehicle forensics becomes easier.

Additionally, the proposed solution enables manufacturers and agencies to track stolen vehicle parts. In many cases, stolen vehicles are sold in parts in the black market. Therefore, if a vehicle is stolen and sold in parts, if its profile is stored in the HPS, whenever someone installs one of its parts the act can be traced, and law enforcement agencies can unravel the story to find how the part reached the new vehicle.

Indeed, one could claim that this proposal can lead to a Denial of Service. An attacker could attach a device that does not have the required certificates making the vehicle non-functional. While theoretically, this is true, the risk is minimal. Firstly, to launch such an attack, the attacker should have physical access to the vehicle to such an extent as to change a primary module. This means that he would just have to remove it to launch his attack, not just change it, as this demands less effort. In any case, even if we assume that this could happen since the immobilizer notifies the user of the type of device that has been installed and its location, the user can trace it easily and proceed to its removal.

It should be highlighted, that the protocols do not transmit any key that would allow an attacker to intercept it. Second, the protocol is safe from replay attacks, a widespread issue in the automotive industry, especially in the past, as it guarantees the freshness of each instance. Finally, the protocol provides mutual authentication to both entities, avoiding man-in-the-middle attacks. More precisely, no sensitive information is disclosed, all entities need to be alive, and no packets can be injected. For the sake of simplicity, in the appendix, we have added the implementation of the two protocols using Scyther [22] to allow the reader to check the security claims automatically.

In addition to the aforementioned security enhancements, our smart contract implements several operations to enable secure version control management. Therefore, its contents can be retrieved and/or modified only by participants with specific roles (each function is implemented with concrete permissions, e.g. using the *require* clause of solidity and variables such as *msg:sender* to check account authenticity). Moreover, we added another security layer to store failed update attempts. Therefore, we can track suspicious addresses (i.e. blockchain accounts) or implement further security policies in such cases, to disable/take down malware campaigns.

4.3 Privacy

The location disclosure on a central database seems a very intrusive act. Nevertheless, the exposure that is introduced from the proposed scheme is the least possible, and it is made under very specific constraints to avoid the "Big Brother" effect.

As discussed in the description of the scheme, the disclosure of a vehicle's location is not made every time the engine is started. The HPS becomes aware of the vehicle's location only when there is a hardware update, or a new device is connected to a communication port. Therefore, location disclosure can only be triggered in two scenarios, when the user is going for maintenance services or when the vehicle is attacked. In the latter scenario, it is evident that users do not mind the disclosure as it is likely to enable the rescue of their asset. In the former, the only disclosure is that of their favorite registered mechanic, as the vehicle itself is not tracked. Contrary to the aforementioned policies e.g. of EDRs continuously sending the driver's location, the imposed privacy exposure is minimum, triggered only by the driver and with consent and acknowledgment. Additionally, users can select different providers and do not depend solely on the manufacturer.

As previously stated in Section 3.3, only the manufacturer or the owner of the vehicle can modify the status of her car in the blockchain or retrieve private information from it. Therefore, the permission of the manufacturer or the owner is required when a 3rd party (e.g. mechanic) has to perform updates. In this regard, blockchain-related operations do not disclose any information nor require additional data from any actors.

For more privacy, one could use the method of Palmieri et al. [61]. This way, the HPS could check whether the vehicle is within a permited area, without the disclosure of its actual location. Moreover, to hide user's and vehicle's ID and to provide even more security, one could use one-way chains to authenticate via one-time passwords [41].

5 Implementation

To evaluate the time and processing cost of the proposed solution, we have implemented the authentication protocols in Python 2.7 using the Flask framework. For the HPS, a web service with a RESTful interface that exchanges compressed and encrypted JSON objects has been developed. We have selected an SQLite database and used typical X.509 elliptic curve certificates generated through OpenSSL.

The computer where the HPS is running is equipped with an Intel® Core™ i7-2600 CPU at 3.40GHz and 16GB of RAM, running on a 64 bit Ubuntu GNU/Linux kernel 3.2.0-29. The client is equipped with an Intel® Core™ i5 CPU at 1.7GHz and 4GB of RAM, running on a 64 bit Ubuntu GNU/Linux kernel 3.2.0-29. The connection between the two is over the Internet, and they reside in two different countries.

Definitely, the implementation is not optimal, as Python is a scripting language and is not as efficient as others. Although we have not paid special attention to the performance nuances, the measurements from 1000 experiments indicate that the proposed solution is efficient and can be easily deployed with minor overhead, as on average, one full round of the proposed scheme takes around 0.023 seconds per thread. Obviously, the overhead is so small that it does not create any noticeable difference to the user.

To showcase the efficacy of the proposed blockchain architecture, we provide experiments using a local private blockchain. More concretely, we created an ethereum-based blockchain using `node`[3] and `ganache-cli`[4], and we used `truffle`[5] to implement and deploy a functional smart contract. In this regard, the deployment of the smart contract and the transaction times were in the order of milliseconds, enabling real-time control version management. The code is available in GitHub[6].

5.1 Feasibility in vehicular devices

To assess the suitability of the proposal in vehicular environments, it is necessary to consider state-of-the-art devices and networks. In order to prototype the system, HPS will be modelled as an Intel Xeon 2010 computer, whereas both immobilizer and new devices will be played by a CyCurV2X [6] embedded platform. Regarding the user side, an NVIDIA Tegra 2010 GPU processor will be considered to play the role of a smartphone. The choice of these devices is because there is a published benchmark of their computational cost for cryptographic operations (Table 1) [2].

Operation	CyCurV2X	NVIDIA Tegra 250	Intel Xeon E5-620
Symmetric encr/decr. (AES CTR 256)	0.23	2.90 e ^{6}	1.89 e ^{7}
RSA encryption	1.74	1.83 e ^{3}	1.06 e ^{4}
RSA decryption	1.32	0.055	0.002
RSA signing	7.156	0.055	0.002
RSA signature verification	27.114	0.001	9.15 e ^{5}

Table 1 Performance of cryptographic primitives on considered devices. All reported timings refer to the average time per byte and are in ms. RSA is considered to operate with 1024 long keys.

We assume that crypto operations are the most demanding ones in the proposed mechanism. Therefore, issues such as message preparation or reception are considered negligible.

[3] `https://nodejs.org/`
[4] `https://github.com/trufflesuite/ganache-cli`
[5] `http://truffleframework.com`
[6] `https://github.com/francasino/Vehicular_Control_Version`

Data field	**Value**	**Data field**	**Value**
Nonce	2	HPS ID	8
User ID	8	HPS certificate	256
Vehicle ID	8	Signature	56
Device ID	8	Latitude, Longitude	4
Device certificate	125	Hash	16
Immobilizer ID	8		

Table 2 Data sizes. Size in bytes.

		Time	
	Data	Transmission	Computation
User	30	0.114	0.000034
Vehicle	12	0.0457	20.954

Table 3 Cost of User authentication phase. Data in bytes, time in ms.

Considering the previous numbers, in what follows we analyse the cost per phase. The data sizes considered for these calculations are shown in Table 2. These choices are based on specifications for similar fields in vehicular standards such as SAE J2735 [3] and IEEE 1609.2 [4].

Tables 3 and 4 show the computation and transmission times for User and Device authentication, respectively. Regarding computation, the vehicle takes most of the time in both phases. Notably, it takes around 21 ms. to compute in the User authentication phase, and around 300 ms. in the Device authentication. In any case, it is noteworthy that these computation times are significantly small, thus supporting the feasibility of the proposal. Furthermore, new devices only need to carry 133 ms. of computation for their authentication. It must be noted that this calculation has been done considering that the computational resources for new devices are equivalent to those present in the ECU. Although it could not be realistic for some cases, given that embedded processors are becoming widespread it is a reasonable assumption.

On the other hand, transmission times are also affordable. It must be noted that three different communication technologies have been considered – Bluetooth, CAN and DSRC. Bluetooth is used for user-vehicle communication in the User authentication phase and has 2.1 Mbit/s as nominal bandwidth [1]. CAN enables communications between the immobilizer and new devices, with a nominal speed of 1 Mbit/s. The communication from the vehicle to HPS is performed through DSRC, with a bandwidth of 6 Mbps [4].

Considering the transmission times using the said technologies, all communications are performed in less than a millisecond.

Taking into account these calculations, both computation and transmission times vouch for the practical feasibility of the proposal.

		Time	
	Data	Transmission	Computation
New device	92	0.736	12.2995
Immob. $\rightarrow$ new device	12	0.096	0
Immob. $\rightarrow$ HPS	171	0.228	298.587
HPS	30	0.04	0.0029

Table 4 Cost of Authentication of a new device to HPS. Time in ms, communication in bytes.

6 Conclusions

As vehicles come equipped with more and more computerised features and enable users to attach more gadgets, attackers find new ways to tamper with them. The situation is getting rather serious because not only vehicles are often unattended, but the automotive industry is so focused on promoting new features on their products, that it does not consider information security a priority. Therefore, vehicles are not only insecure against acts of theft but in specific attack scenarios, passengers lives can also be endangered, for instance from cyber attacks.

In this context, to prevent and protect both the essence of a modern vehicle, its human operator and passengers, this work proposes the use of a security protocol that relies on the use of blockchain and an external database, which is contacted by the vehicle's immobilizer. In this regard, our system can report hardware changes and mitigate attacks originated from unauthenticated plugged in devices, as well as from unauthorised users. As discussed, the proposed solution introduces a minimal overhead, offering an additional layer of security. This layer enables the detection, blocking and tracing of several attacks.

Certainly, the adoption of our solution demands a number of changes in the design of the vehicle's architecture and its parts. Nevertheless, since vehicles are currently under tremendous transformations, it is time to rethink how to integrate into the vehicle well-known and long-lived security mechanisms, adequately adapted to meet the real-time requirements of vehicles.

References

1. Bluetooth specifications. `https://www.bluetooth.org/en-us/specification/adopted-specifications`.
2. eBACS: ECRYPT Benchmarking of Cryptographic Systems. `http://bench.cr.yp.to/computers.html`.
3. SAE J2735 Dedicated Short Range Communications (DSRC) Message Set Dictionary. *SAE J2735*, November 2009.
4. IEEE Standard for Wireless Access in Vehicular Environments Security Services for Applications and Management Messages. *IEEE Std 1609.2-2013 (Revision of IEEE Std 1609.2-2006)*, pages 1–289, April 2013.
5. eCall: Time saved = lives saved. `https://ec.europa.eu/digital-single-market/ecall-time-saved-lives-saved`, 2014.

6. http://auto2015.bosch.com.cn/ebrochures2015/energizing_powertrain/etas/141126_produktblatt_cycurv2x_web.pdf, 2015.
7. AcuraLink. http://owners.acura.com/acuralink.
8. Automotive Open System Architecture. http://www.autosar.org/.
9. M. Awais Hassan, U. Habiba, U. Ghani, and M. Shoaib. A secure message-passing framework for inter-vehicular communication using blockchain. *International Journal of Distributed Sensor Networks*, 15(2), 2019.
10. J. Bell, T.D. Latoza, F. Baldmitsi, and A. Stavrou. Advancing open science with version control and blockchains. pages 13–14, 2017.
11. E. Biham, O. Dunkelman, S. Indesteege, N. Keller, and B. Preneel. How to steal cars - a practical attack on keeloq. In *CRYPTO 2007*, 2010.
12. BMW. BMW Teleservices. https://secure.bmw.com/com/en/owners/service/teleservices/2013/index.html.
13. Allan Bonnick. *Automotive computer controlled systems*. Routledge, 2001.
14. S. Bono, M. Green, A. Stubblefield, A. Juels, A. Rubin, and M. Szydlo. Security analysis of a cryptographically-enabled rfid device. In *USENIX Security*, 2007.
15. R. Bosch. *Bosch Automotive Electrics and Automotive Electronics: Systems and Components, Networking and Hybrid Drive*. Springer Vieweg, 2013.
16. Fran Casino, Thomas K. Dasaklis, and Constantinos Patsakis. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and Informatics*, 36:55 – 81, 2019.
17. M. Cebe, E. Erdin, K. Akkaya, H. Aksu, and S. Uluagac. Block4forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles. *IEEE Communications Magazine*, 56(10):50–57, 2018.
18. Robert N Charette. This car runs on code. *Spectrum*, 46(3), 2009.
19. M. L. Chavez, C. H. Rosete, and F. R. Henriguez. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *Proceedings of the 15th International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 166–170, 2005.
20. S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, Savage S., K. Koscher, A. Czeskis, F. Roesner, and T. Kohn. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security*, 2011.
21. K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.
22. C.J.F. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc.*, volume 5123/2008 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.
23. T. K. Dasaklis, F. Casino, and C. Patsakis. Blockchain meets smart health: Towards next generation healthcare services. In *2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–8, July 2018.
24. Kleanthis Dellios, Dimitrios Papanikas, and Despina Polemi. Information security compliance over the intelligent transport systems: Is it possible? *Security & Privacy*, 13(3):770–772, 2015. (in press).
25. Kleanthis Dellios, Constantinos Patsakis, and Despina Polemi. Automobile 2.0: Reformulating the automotive platform as an it system. *IT Professional*, 18(5):48–56, 2016.
26. A. Dorri, M. Steger, S.S. Kanhere, and R. Jurdak. Blockchain: A distributed solution to automotive security and privacy. *IEEE Communications Magazine*, 55(12):119–125, 2017.
27. T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, Salmasizadeh M., and M. Shalmani. Physical cryptanalysis of keeloq code hopping applications. http://eprint.iacr.org/2008/058, 2008.
28. ENISA. Cyber security and resilience of smart cars. https://www.enisa.europa.eu/publications/cyber-security-and-resilience-of-smart-cars, 2017.
29. Evita FP7 project. http://evita-project.org.
30. Earlence Fernandes, Bruno Crispo, and Marco Conti. Fm 99.9, radio virus: Exploiting fm radio broadcasts for malware deployment. *Information Forensics and Security, IEEE Transactions on*, 8(6):1027–1037, 2013.

31. Ian Foster, Andrew Prudhomme, Karl Koscher, and Stefan Savage. Fast and vulnerable: a story of telematic failures. In *Proceedings of Workshop On Offensive Technologies (WOOT), Washington, DC*, 2015.
32. Flavio D Garcia, Gerhard de Koning Gans, Roel Verdult, and Milosch Meriac. Dismantling iclass and iclass elite. In *European Symposium on Research in Computer Security*, pages 697–715. Springer, 2012.
33. Flavio D. Garcia, David Oswald, Timo Kasper, and Pierre Pavlidès. Lock it and still lose it - on the (in)security of automotive remote keyless entry systems. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.* USENIX Association, 2016.
34. A. Gkogkidis, N. Giachoudis, G. Spathoulas, and I. Anagnostopoulos. Implementing a blockchain infrastructure on top of vehicular ad hoc networks. *Advances in Intelligent Systems and Computing*, 879:764–771, 2019.
35. A. Groll and C. Ruland. Secure and authentic communication on existing in-vehicle networks. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 1093–1097, 2009.
36. S. Indesteege, N. Keller, O. Dunkelman, E. Biham, and B. Preneel. A practical attack on keeloq. In *EUROCRYPT'08 Proceedings of the theory and applications of cryptographic techniques*, pages 1–18, 2008.
37. D. Kevin and B. David. Hacit2: A privacy preserving, region based and blockchain application for dynamic navigation and forensics in vanet. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 258:225–236, 2019.
38. Pierre Kleberger and Tomas Olovsson. Securing vehicle diagnostics in repair shops. In *Computer Safety, Reliability, and Security*, pages 93–108. Springer, 2014.
39. Timo Kosch, Schroth Christoph, Strassberger Markus, and Bechler Marc. *Automotive Inter-networking.* Wiley Press, 2012.
40. 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 *IEEE Symposium on Security and Privacy, Oakland, CA*, pages 447–462, 2010.
41. Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.
42. A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C.P.A. Ogah, and Z. Sun. Blockchain-based dynamic key management for heterogeneous intelligent transportation systems. *IEEE Internet of Things Journal*, 4(6):1832–1843, 2017.
43. L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang. Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19(7):2204–2220, 2018.
44. Y.-N. Liu, S.-Z. Lv, M. Xie, Z.-B. Chen, and P. Wang. Dynamic anonymous identity authentication (daia) scheme for vanet. *International Journal of Communication Systems*, 32(5), 2019.
45. Z. Lu, W. Liu, Q. Wang, G. Qu, and Z. Liu. A privacy-preserving trust model based on blockchain for vanets. *IEEE Access*, 6:45655–45664, 2018.
46. Z. Lu, Q. Wang, G. Qu, and Z. Liu. Bars: A blockchain-based anonymous reputation system for trust management in vanets. pages 98–103, 2018.
47. Kevin Mahaffey. Hacking a tesla model s: What we found and what we learned. `https://blog.lookout.com/hacking-a-tesla`, 2015.
48. Hafizah Mansor, Konstantinos Markantonakis, Raja Akram, Keith Mayes, and Iakovos Gurulian. *Log Your Car: Reliable Maintenance Services Record*, volume 10143 of *Lecture Notes in Computer Science*, pages 484–504. Springer, 2017.
49. Hafizah Mansor, Konstantinos Markantonakis, Raja Naeem Akram, and Keith Mayes. Don't brick your car: Firmware confidentiality and rollback for vehicles. In *Availability, Reliability and Security (ARES), 2015 10th International Conference on*, pages 139–148. IEEE, 2015.
50. Charlie Miller and Chris Valasek. A survey of remote automotive attack surfaces, 8 2014. Blackhat USA 2014, Las Vegas, NV, USA.

51. Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle, 8 2015. Blackhat USA 2015, Las Vegas, NV, USA.
52. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
53. N. Naver and F. Simonot-Lion. *Automotive Embedded Systems Handbook: Industrial Information Technology.* CRC Press, 2009.
54. D. K. Nilsson, U. E. Larson, and E. Jonsson. Efficient in-vehicle delayed data authentication based on compound message authentication codes. In *Proceedings of the 68th IEEE Vehicular Technology Conference (VTC 2008-Fall)*, pages 1–5, 2008.
55. K. Nohl. Car immobilizer security. In *ESCAR*, 2010.
56. H. Oguma, A. Yoshioka, M. Nishikawa, R. Shigetomi, A. Otsuka, and H. Imai. New attestation based security architecture for in-vehicle communication. In *IEEE GLOBECOM*, pages 1909–1914, 2008.
57. H. Onishi. A survey: Engineering challenges to implement vanet security. 2018.
58. OnStar. `https://www.onstar.com`.
59. Organisation Internationale des Constructeurs d'Automobiles. `http://oica.net/`.
60. V. Ortega, F. Bouchmal, and J.F. Monserrat. Trusted 5g vehicular networks: Blockchains and content-centric networking. *IEEE Vehicular Technology Magazine*, 13(2):121–127, 2018.
61. Paolo Palmieri, Luca Calderoni, and Dario Maio. Spatial bloom filters: Enabling privacy in location-aware applications. In *Information Security and Cryptology*, pages 16–36. Springer International Publishing, 2014.
62. Constantinos Patsakis and Kleanthis Dellios. Securing in-vehicle communication and redefining the role of automotive immobilizer. In *SECRYPT*, pages 221–226, 2012.
63. Constantinos Patsakis, Kleanthis Dellios, and Mélanie Bouroche. Towards a distributed secure in-vehicle communication architecture for modern vehicles. *Computers & Security*, 40:60 – 74, 2014.
64. Constantinos Patsakis and Agusti Solanas. Privacy-aware event data recorders: Cryptography meets the automotive industry again. *IEEE Communications Magazine*, 51(12), 2013.
65. Kefa Rabah. Convergence of AI, IoT, Big Data and Blockchain: A Review. *The Lake Institute Journal*, 1(1):1–18, 2018.
66. C. Riegger, T. Vinçon, and I. Petrov. Efficient data and indexing structure for blockchains in enterprise systems. pages 173–182, 2018.
67. I. Rouf, R. Miller, R. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *in Proceedings of the 19th USENIX Security Symposium, Washington DC*, page 21, 2010.
68. Henry Samuel. Three quarters of cars stolen in france 'electronically hacked'. `http://www.telegraph.co.uk/news/worldnews/europe/france/11964140/Three-quarters-of-cars-stolen-in-France-electronically-hacked.html`, 2015.
69. Simha Sethumadhavan, Adam Waksman, Matthew Suozzo, Yipeng Huang, and Julianna Eum. Trustworthy hardware from untrusted components. *Communications of the ACM*, 58(9):60–71, 2015.
70. R. Sharma and S. Chakraborty. B2vdm: Blockchain based vehicular data management. pages 2337–2343, 2018.
71. Nick Szabo. The idea of smart contracts, 1997.
72. András Szijj and Levente Buttyán. Hacking cars in the style of stuxnet, 10 2015. Hacktivity 2015, Las Vegas, NV, USA.
73. C. Szilagyi and P. Koopman. A flexible approach to embedded network multicast authentication. In *2nd Workshop on Embedded Systems Security (WESS)*, pages 165–174, 2008.
74. C. Szilagyi and P. Koopman. Flexible multicast authentication for time-triggered embedded control network applications. In *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*, pages 165–174. IEEE, 2009.
75. S Tuohy, M Glavin, C Hughes, E Jones, M Trivedi, and L Kilmartin. Intra-vehicle networks: A review. *Intelligent Transportation Systems, IEEE Transactions*, 16(2), 2015.

76. Roel Verdult, Flavio D Garcia, and Barıs Ege. Dismantling megamos crypto: Wirelessly lockpicking a vehicle immobilizer. In *22nd USENIX Security Symposium (USENIX Security 2013). USENIX Association*, 2013.
77. A. Weimerskirch, C. Paar, and M. Wolf. *Secure In-Vehicle Communication*. Springer-Verlag, 2006.
78. Victoria Woollaston. Forget carjacking, the next big threat is car-hacking: Thousands of vehicles are being stolen using cheap gadgets bought online. `http://www.dailymail.co.uk/sciencetech/article-2623275/Forget-carjacking-big-threat-car-HACKING-Thousands-vehicles-stolen-using-cheap-gadgets-bought-online.html`, 2014.
79. Z. Yang, K. Yang, L. Lei, K. Zheng, and V.C.M. Leung. Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet of Things Journal*, 2018.
80. Tao Zhang, Helder Antunes, and Suhas Aggarwal. Defending connected vehicles against malware: Challenges and a solution framework. *Internet of Things Journal, IEEE*, 1(1):10–21, 2014.
81. X. Zhang, R. Li, and B. Cui. A security architecture of vanet based on blockchain and mobile edge computing. pages 258–259, 2019.
82. J. Leon Zhao, Shaokun Fan, and Jiaqi Yan. Overview of business innovations and research opportunities in blockchain and introduction to the special issue. *Financial Innovation*, 2(1):28, Dec 2016.

A Appendix

A.1 User authentication

```
usertype SessionKey;
protocol auth(A,B){

        role A{
                fresh ra: Nonce;
                var rb: Nonce;
                send_1(A,B, ra,A);
                recv_2(B,A,{ra,rb,B}k(A,B));
                send_3(A,B,{rb,ra,A}k(A,B));
                claim_A1(A,Alive);
                claim_A2(A,Niagree);
                claim_A3(A,Nisynch);
        }

        role B{
                fresh rb: Nonce;
                var ra: Nonce;
                fresh kab: SessionKey;
                recv_1(A,B,ra,A);
                send_2(B,A,{ra,rb,B}k(A,B));
                recv_3(A,B,{rb,ra,A}k(A,B));
                claim_B1(B,Alive);
                claim_B2(B,Niagree);
                claim_B3(B,Nisynch);
        }
}
```

A.2 Device authentication

```
protocol auth3(Immo,HPS,Dev){
        hashfunction H;

        role Dev{
                fresh rDev;
                var r;
                send_1(Dev,Immo, {rDev,Dev,Immo}k(Dev,Immo));
                recv_!5(HPS,Dev,{r,rDev,HPS}pk(Dev));
                send_!6(Dev,HPS,{H(rDev,Dev,HPS)}k(r));

                claim_D1(Dev,Alive);
                claim_D2(Dev,Niagree);
                claim_D3(Dev,Nisynch);
        }

        role Immo{
                var rb: Nonce;

                fresh rImmo,UserCred,VehicleID,DevCert,x,y,rDev:Nonce;
                var rHPS,rDev:Nonce;

                recv_1(Dev,Immo, {rDev,Dev,Immo}k(Dev,Immo));

                send_2(Immo,HPS, {rImmo,UserCred,VehicleID,Immo}k(Immo,HPS));
                recv_3(HPS,Immo,{rHPS,H(rImmo,UserCred,VehicleID,Immo)}k(Immo,HPS));
                send_4(Immo,HPS,{rHPS,Immo,x,y,DevCert,rDev}k(Immo,HPS));

                claim_A1(Immo,Alive);
                claim_A2(Immo,Niagree);
                claim_A3(Immo,Nisynch);
                claim_A4(Immo,Secret,x);
                claim_A5(Immo,Secret,y);
                claim_A6(Immo,Secret,UserCred);
                claim_A7(Immo,Secret,VehicleID);
        }

        role HPS{
                fresh rb: Nonce;
                fresh kab: SessionKey;

                var rImmo,UserCred,VehicleID,DevCert,x,y,rDev:Nonce;
                fresh rHPS,r:Nonce;

                recv_2(Immo,HPS, {rImmo,UserCred,VehicleID,Immo}k(Immo,HPS));
                send_3(HPS,Immo,{rHPS,H(rImmo,UserCred,VehicleID,Immo)}k(Immo,HPS));
                recv_4(Immo,HPS,{rHPS,Immo,x,y,DevCert,rDev}k(Immo,HPS));

                send_!5(HPS,Dev,{r,rDev,HPS}pk(Dev));
                recv_!6(Dev,HPS,{H(rDev,Dev,HPS)}k(r));

                claim_B1(HPS,Alive);
                claim_B2(HPS,Niagree);
                claim_B3(HPS,Nisynch);
        }
}
```